Automate Your Business Processes

Salesforce, Summer ’25

Last updated: May 16, 2025

CONTENTS

Automate Your Business Processes 1

Get Started 2
Browser Requirements 4
Choose Which Salesforce Flow Feature to Use 5
Automation Lightning App 7
Automate Tasks with Flows 16
Automate Complex Processes with Orchestrations 662
Suggest Options to Users with Recommendation Strategies 749
Automated Actions 811
Approval Processes 827
Legacy Salesforce Flow Features 882

Index 1023

AUTOMATE YOUR BUSINESS PROCESSES WITH SALESFORCE FLOW

Say goodbye to manual tasks and hello to streamlined operations with Salesforce Flow. By using automation to reduce the number of manual processes and repetitive tasks, you decrease turnaround time, improve accuracy and user productivity, and reduce costs. Salesforce Flow offers no-code, point-and-click tools like Flow Builder and Flow Orchestration, which help you create guided visual experiences, behind-the-scenes automations, and approval automations.

This video explores some ways you can use Flow Builder to create powerful automations without code.

Watch a video

View this video in a separate tab.

Get Started with Automation Learning Map

Whether you're new to automation or a seasoned expert, when you're building an automation project, you have a business problem to solve. And after you solve it, you must measure success and maintain the solution. If you're looking to create automations in your Salesforce org, this learning map focused on Salesforce Flow products helps you get started with information, resources, and product and feature content.

Browser Requirements

Our automation tools support these web browsers.

Choose Which Salesforce Flow Feature to Use

Salesforce provides a robust set of automation features to help you save time and resources. Use Flow Builder to automate most of your organization's repetitive business processes. More features can provide further automation functionality, including approval processes, Flow Orchestration, Einstein Next Best Action, and Apex.

Automation Lightning App

Monitor and manage your existing flows and orchestrations from the Automation Lightning app.

Automate Tasks with Flows

Automation can help your company reduce repetitive tasks, increase speed and accuracy, and do more with less. With Flow Builder, create simple or complex automations, forms, and guided user experiences with little to no code. Then, add what you create to the Salesforce app, experience, or portal where you need it.

Automate Complex Processes with Orchestrations

As your company grows, so does the complexity of your workflows. Processes often require input from multiple users in multiple departments across multiple time zones. This increased complexity results in an increased amount of time spent waiting for each person to complete their task in the proper order. Flow Orchestration helps you streamline this process with orchestrations: multi-step processes that interact with multiple users and systems.

Suggest Options to Users with Recommendation Strategies

Display the right recommendations to the right people at the right time with Einstein Next Best Action. Create and display offers and actions for your users that are tailored to meet your unique criteria. Develop a strategy that applies your business logic to refine those recommendations. Your strategy distills your recommendations into a few key suggestions, like a repair, a discount, or an add-on service. Display the final recommendations in your Lightning app or Experience Builder site.

Automated Actions

An automated action is a reusable component that performs some sort of action behind the scenes—like updating a field or sending an email. After you create an automated action, add it to a process, milestone, or other automated process.

Approval Processes

It's likely that you're familiar with process automation in the form of workflow rules. Approval processes take automation one step further, letting you specify a sequence of steps that are required to approve a record.

Legacy Salesforce Flow Features

Legacy Salesforce Flow features include Process Builder and Workflow Rules.

Get Started with Automation Learning Map

Whether you're new to automation or a seasoned expert, when you're building an automation project, you have a business problem to solve. And after you solve it, you must measure success and maintain the solution. If you're looking to create automations in your Salesforce org, this learning map focused on Salesforce Flow products helps you get started with information, resources, and product and feature content.

Each section focuses on a category of work, with related tasks listed for reference. Within each section, there are links to content that help you complete the tasks required to build out your automation solution. The goal is to aggregate information in one place so that you can get back to it quickly when you need guidance.

Assess and Plan	**Design**	**Build**
• Identify key business problems with an opportunity for automation. • Assess business requirements for automation. • Evaluate automation feasibility and value. • Define functional requirements related to business goals. • Define a high-level execution plan. *Playbook*: 3 Ways to Innovate Fast with Enterprise-Scale Automation Choose Which Salesforce Flow Feature to Use *Trailhead*: Flow Builder Concepts *Trailhead*: Build Flows with Flow Builder *Trailhead*: Build a Simple Flow *Trailhead*: Flow Fundamentals Superbadge Unit	• Design the automation solution. • Secure the automation. • Create high-level documentation for the automation. • Share the details with your stakeholders. • Ensure that infrastructure and platform resources are set up and allocated. *Trailhead*: Flow Interactions Superbadge Unit Flow Best Practices Flow Types Security Considerations for Flows Limit User Access to Execute Flows *Salesforce Architects*: Salesforce Diagramming Framework *Salesforce Architects*: Template Gallery—Find Salesforce data models and pre-built artifacts.	• Build the steps for the automation workflow. • Establish connectivity to data and systems. • Validate automation functionality. • Set up monitoring for automation. *Trailblazer Community*: Salesforce Flow Automation—Open discussion group for Salesforce Flow users. *Trailhead*: Quick Start: Einstein Copilot *Trailhead*: Use External Services with a Flow *Trailhead*: Quick Start: Create HTTP Callouts with Flow Builder HTTP Callout *Trailhead*: MuleSoft Composer Flow *Trailhead*: Build a Discount Approval Process *Trailhead*: Build a Discount Calculator

Trailhead: Flow Optimization Superbadge Unit

Trailhead: Approve Records with Approval Processes

Flow Limits and Considerations

Salesforce Admins Blog: Process Automation—A collection of videos and articles centered around process automation with Salesforce Flow products.

Salesforce Architects: Automated—Determine the best way to solve your automation challenges.

Salesforce Admins Blog: Refine Your Process Automation Skills as a Salesforce Admin

Salesforce Architects: Intentional—Create solutions that deliver business value immediately and over time.

Salesforce Architects: Resilient—Improve the ability of your solutions to quickly recover from a problem or failure.

Flow Concepts

Trailhead: Flow Orchestration Basics

What's the Difference Between a Flow and an Orchestration?

Flow Orchestration Concepts

Flow Orchestration Entitlements

Approval Processes

Salesforce Architects: Building Forms

Salesforce Architects: Designing Your Record-Triggered Automation

Salesforce Admins Blog: The Ultimate Guide to Flow Best Practices and Standards

Salesforce Admins YouTube Channel: Automate This! Series—A series of videos focused on automation solutions.

Trailhead: Build an Email Flow for Program Management Module (PMM)

Build a Flow

Flow Reference

Build an Orchestration

Test

- Conduct end-to-end testing.
- Debug issues.

Salesforce Architects: Resilient Testing Strategy

Trailhead: Distribute and Implement Flows

Trailhead: Flow Debugging Superbadge Unit

Trailhead: Flow Error Handling Superbadge Unit

Testing Your Flow

Deploy

- Obtain pre-deployment approvals.
- Activate and deploy your automation.
- Make an automation available for use and reuse.

Trailhead: Screen Flow Distribution Superbadge Unit

Trailhead: Flow Data Collections Optimization Superbadge Unit

Distribute a Flow

Manage and Maintain

- Monitor the automation.
- View analytics data.
- Identify and troubleshoot issues.
- Manage users and permissions.
- Develop a change management strategy.
- Implement continuous improvements.

Trailhead: Flow Administration Superbadge Unit

Troubleshoot Flow Errors *Video*: Troubleshooting Flow Errors Using the Debug Button Debug a Flow in Flow Builder Troubleshoot Orchestrations	Flow Distribution Concepts Activate or Deactivate a Flow Version Deploy an Orchestration Activate an Approval Process Prepare Your Org for Approvals	*Trailhead*: Flow Management Specialist Superbadge Flow Interviews Monitor Flows and Processes *Trailhead*: Orchestrate Complex Processes with Flow Orchestration Approval History Reports

Browser Requirements

Our automation tools support these web browsers.

Supported Browsers

- Google Chrome™ (latest stable version)
- Mozilla® Firefox® (latest stable version)
- Microsoft® Edge (latest stable version)
- Apple® Safari® (latest stable version)

Microsoft® Internet Explorer® isn’t supported.

Note: The browser vendor defines the latest for their own browser. Check with your browser vendor to determine the latest version available.

Screen Resolution

The minimum screen resolution required is 1024 x 768. We recommend using a width of at least 1200.

Choose Which Salesforce Flow Feature to Use

Salesforce provides a robust set of automation features to help you save time and resources. Use Flow Builder to automate most of your organization's repetitive business processes. More features can provide further automation functionality, including approval processes, Flow Orchestration, Einstein Next Best Action, and Apex.

Note: Use Flow Builder instead of Process Builder and workflow rules. Flows have all the capabilities of Process Builder processes and workflow rules, and much more, providing a single no-code automation home.

To determine your automation needs and which feature to use, ask these questions.

1. When do you want the automation to start running?
2. What do you want to happen after the automation starts?

For example, when you add a contact, you want the contact to receive a welcome email.

1. Start: When a contact is created.
2. Automation: Send an email to the contact.

What Can Start or Trigger a Flow?

FLOW STARTS OR TRIGGERS	FLOW TYPE TO USE	FOR EXAMPLE, YOU WANT SOMETHING TO HAPPEN…
When a record is created	Record-Triggered	When a new case is created.
When a record is updated	Record-Triggered	When a lead's status field is changed.
When a record is created or updated	Record-Triggered	When an account is created or the account priority field is changed.
When a record is deleted	Record-Triggered	When a contact is deleted.
After a certain amount of time	Record-Triggered	A week after a quote is created. *Add a scheduled path to the record-triggered flow.*
At a specified time and frequency	Schedule-Triggered	Every Saturday at midnight.
When a user clicks a button on a form	Screen	When a customer enters contact information into a flow screen and clicks the Next button.
When a user clicks a quick action button	Screen	When an employee clicks Request PTO on their employee record. *Opens a form to complete.*
When a user clicks a custom button or link	Autolaunched	When a user clicks a Complete Sale button after closing an opportunity. *Starts background automations, such*

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Flow Builder and Einstein Next Best Action are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approval Processes and Flow Orchestration are available in **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Apex is available in **Enterprise**, **Performance**, **Unlimited**, **Developer**, and **Database.com** Editions

FLOW STARTS OR TRIGGERS	FLOW TYPE TO USE	FOR EXAMPLE, YOU WANT SOMETHING TO HAPPEN…
		as updating records and emailing stakeholders.
When called by another flow	Autolaunched or Screen	When a flow executes another flow within the same running instance to reduce repetition within the main flow.
When called by Apex code	Autolaunched	When an Apex class is triggered by a change to an opportunity's stage, which triggers an autolaunched flow.
When a platform event message is received	Platform Event–Triggered	When an integrated printer is out of ink, it publishes a platform event message.

What Can a Flow Automate?

AUTOMATION	Example
Create records	Create an account.
Update records	Change a contact's address.
Delete records	Delete a user's permission.
Send an email	Send an introductory email to a lead.
Collect input from external users with an online form	Let new customers add themselves to your contacts using an online form.
Collect input from internal users with a form placed on a Lightning page or launched by a button	Create an account, a contact, and a case quickly with one simple form placed directly on support reps' Home pages.
Send a custom notification	Notify managers when significant opportunities are won.
Send a survey	Send a customer satisfaction survey after an opportunity is closed.
Submit a record for approval	Require that managers approve discounts.
Run another flow in the context of the current flow	Run another flow that creates a contact within your account-creation flow.
Access external systems	Call actions registered by External Services to insert details from an external banking system into a Salesforce record.
Call a custom invocable action	Execute a custom Apex method that uses the InvocableMethod annotation. You decide what the action does.
Send outbound messages	Initiate the reimbursement process for an approved expense report by sending a message to an external HR system.

When Do I Use Another Tool or Feature?

WHEN YOU WANT TO...	USE...	FOR EXAMPLE, YOU WANT...
Approve records through multiple levels in your organization.	Approval Processes	Two levels of management to review and approve employee PTO request records.
Suggest offers and actions to users that are tailored to meet your unique business criteria.	Einstein Next Best Action	To prompt agents to offer service contracts to customers who don't have them.
Coordinate multiple flows, and assign them to multiple teams or individuals.	Flow Orchestration	An automated hiring process that involves HR, Finance, and the hiring team.
Perform an operation for more records than scheduled-triggered flows allow.	Batch Apex	To update all your 600,000 contacts at once.
Perform CPU-intensive operations.	Apex	To calculate a highly complex discount rate.

SEE ALSO:

Per-Transaction Flow Limits

Salesforce Architects : Record-Triggered Automation

Automation Lightning App

Monitor and manage your existing flows and orchestrations from the Automation Lightning app.

To make this app available to anyone with the permission to see it, select the **Enable the Automation Lightning App** process automation setting in Setup.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Automation Lightning App: Home

The Home tab includes list views of your most recently modified flows and flows that are configured with errors. Search for flows using a keyword in the flow label or description. Additionally, find direct links to helpful resources.

Automation Lightning App: Flows

The Flows tab includes a number of standard list views or any list views that you create. Search for keywords in flow labels or descriptions. Filter or sort flows by fields like type, progress status, last modified date, last modifying user, and associated record fields. If you have the appropriate permissions, create flows in Flow Builder right from the Flows tab.

Automation Lightning App: Orchestrations

The Orchestrations tab includes a number of standard orchestration list views and any list views that you create. Search for keywords in orchestration labels or descriptions. Filter or sort orchestrations by fields like trigger type, status, or last modified date. View and manage related orchestration runs for a selected orchestration.

Connections Tab

The **Connections** tab includes a list view of connections. If you're a Salesforce admin, you see all the connections for your org.

Automation Lightning App: Orchestrations

The Orchestrations tab includes a number of standard orchestration list views and any list views that you create. Search for keywords in orchestration labels or descriptions. Filter or sort orchestrations by fields like trigger type, status, or last modified date. View and manage related orchestration runs for a selected orchestration.

Automation Lightning App: Home

The Home tab includes list views of your most recently modified flows and flows that are configured with errors. Search for flows using a keyword in the flow label or description. Additionally, find direct links to helpful resources.

See list views of your most recently modified flows and any flows that are configured with errors. Navigate to the Trailblazer Community and to relevant learning material on Trailhead.

In the Recently Modified list view, filter, search for, sort, and monitor your latest flows, see the details of an existing flow, or create a flow or open an existing flow in Flow Builder.

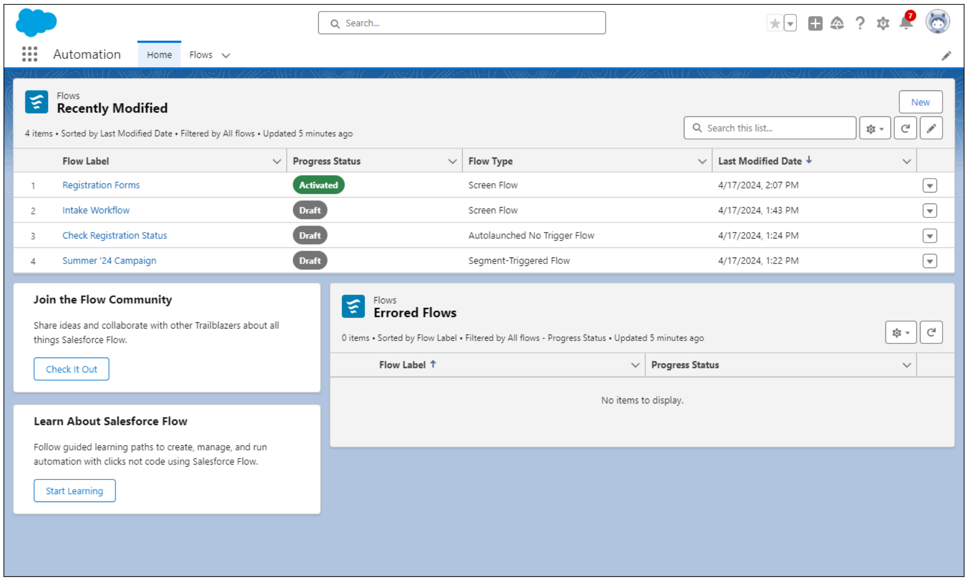
In the Errored Flows list view, organize and monitor segment- and form-triggered flows configured with errors, see the details of a segment- or form-triggered flow, or create a flow or open an existing flow in Flow Builder.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Use the Action menu in any list view to open the active or latest version of a flow, change the flow owner, or delete a flow.



Flows
Recently Modified

4 items • Sorted by Last Modified Date • Filtered by All flows • Updated 5 minutes ago

	Flow Label	Progress Status	Flow Type	Last Modified Date
1	Registration Forms	Activated	Screen Flow	4/17/2024, 2:07 PM
2	Intake Workflow	Draft	Screen Flow	4/17/2024, 1:43 PM
3	Check Registration Status	Draft	Autolaunched No Trigger Flow	4/17/2024, 1:24 PM
4	Summer '24 Campaign	Draft	Segment-Triggered Flow	4/17/2024, 1:22 PM

Join the Flow Community

Share ideas and collaborate with other Trailblazers about all things Salesforce Flow.

Check It Out

Learn About Salesforce Flow

Follow guided learning paths to create, manage, and run automation with clicks not code using Salesforce Flow.

Start Learning

Flows
Errored Flows

0 items • Sorted by Flow Label • Filtered by All flows - Progress Status • Updated 5 minutes ago

Flow Label	Progress Status

No items to display.

Automation Lightning App: Flows

The Flows tab includes a number of standard list views or any list views that you create. Search for keywords in flow labels or descriptions. Filter or sort flows by fields like type, progress status, last modified date, last modifying user, and associated record fields. If you have the appropriate permissions, create flows in Flow Builder right from the Flows tab.

In the Flows tab, see a list view of all flows, recently viewed flows, segment- and form-triggered flows configured with errors, or recently modified flows. Organize and monitor your flows, see the details of an existing flow, or, if you have the appropriate permissions, create or open a flow in Flow Builder. To create a flow in Flow Builder, click **New**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Column Descriptions

The standard list views include these columns. If you create a custom list view in the Flows tab, you can specify which columns appear.

Column	Description
Flow Label	User-friendly label given to the latest version of the flow by the flow creator or editor. To open the record Details page for the flow, click the label. On the Details page, you can edit flow details including the Category and Subcategory.
Progress Status	Status of the flow version. Segment-triggered and form-triggered flow versions can have any progress statuses except Paused. Other flow versions can have only the Activated, Draft, and Paused statuses. **Activated** The flow version is active and doesn't have a scheduled trigger. Corresponds to the Active flow status. **Canceled** The flow version is deactivated and all flow interviews are forcibly stopped. **Completed** The flow version is deactivated and all flow interviews reached completion. **Draft** The flow version hasn't been activated yet. Corresponds to the Inactive flow status. **Error** A configuration issue prevents the flow version from running. **Finishing** The flow version is deactivated and some flow interviews have not yet run. **Paused** The flow version is paused. Corresponds to the Paused flow status. **Preparing** Salesforce is performing backend operations to set up the flow. This status is typically completed in a few moments. **Scheduled** The flow version is active and has a scheduled trigger.
Flow Type on page 24	Type of the flow, for example, screen flow.
Last Modified Date	Date and time that the flow was last updated. If you update your org, the Last Modified date is the date of the update until you make subsequent changes the flow.
Active Version	User-friendly label given to the active version of the flow by the flow creator or editor. The active version can be different from the latest version. To open the record detail page for the flow, click the label.
Description	The flow's description, which is provided by the flow's creator or editor.
Action menu	Dropdown menu that you can use to: • Open the active version of a flow • Open the latest version of a flow • Change the flow owner • Delete a flow

Automation Lightning App: Orchestrations

The Orchestrations tab includes a number of standard orchestration list views and any list views that you create. Search for keywords in orchestration labels or descriptions. Filter or sort orchestrations by fields like trigger type, status, or last modified date. View and manage related orchestration runs for a selected orchestration.

Each orchestration has an associated status.

Orchestration Status	Description
Active	The current version of the orchestration has been activated.
Draft	The current version of the orchestration hasn't been activated and contains errors.
Inactive	The current version of the orchestration hasn't been activated or has been deactivated.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

View Active Orchestration Version Details

View details for an orchestration's active version.

View All Orchestration Versions

View details about any version of an orchestration.

View Orchestration Runs Related to an Orchestration

View orchestration run records that are related to an orchestration. You can also view the orchestration stage run, orchestration step run, and orchestration run log records related to a selected orchestration run.

Manage Orchestration Runs

Cancel, debug, suspend, and resume orchestration runs from the Automation Lightning app.

Connections Tab

The **Connections** tab includes a list view of connections. If you're a Salesforce admin, you see all the connections for your org.

A connection enables you to connect to external systems. A connection has the access and authorization configuration. It also has the metadata of the specific API or object, such as input and output fields from the connected external system. You can use and reference these fields in flows for your integration and automation needs. You can use the same connection multiple times in the same flow or in different flows that you create.

Creating a connection also creates the named credentials and external credentials behind the scenes so that you don't have to create the same named credentials and external credentials on multiple screens in Setup.

Automation Lightning App: Create a Connection

TBD

Automation Lightning App: Test a Connection

Test a connection in the Automation Lightning app.

Automation Lightning App: Edit a Connection

Edit an existing connection to a third-party system.

Automation Lightning App: Delete a Connection
Delete a connection to an external service.

Automation Lightning App: Create a Connection

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create connections:
- Manage Integration Connections

TBD

TBD

TBD

1. In the Automation Lightning app, select the **Connections** tab.
2. Click **New Connection**.

Example: TBD

You prob don't need this...

Automation Lightning App: Test a Connection

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To test connections:
- Manage Integration Connections

Test a connection in the Automation Lightning app.

1. In the Automation Lightning app, select the **Connections** tab.
2. Select the connection.
3. In the **Connections Details** section, click **Test**.

   If the connection test is successful, a success message appears. If the connection test isn’t successful, a failure message appears. The **Status** and **Last Tested On** fields are updated automatically, so you always know the status, date, and time of your most recent test.

Automation Lightning App: Edit a Connection

Edit an existing connection to a third-party system.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To edit connections:

- Manage Integration Connections

1. In the Automation Lightning app, select the **Connections** tab.
2. In the **Connection Details** section, click **Edit**.
3. Read the pop-up and click **Edit**.

   Important: If you edit a connection used in a flow and it can't connect to the external system, errors can occur in the flow. Connection edits can't be undone, take effect immediately, and impact all flows using the connection.

4. Edit the fields.
   Disabled fields can't be edited, which includes **Connection API Type** and **Authorization Type**
   If you edit any field in the **Authentication Details** section, the value in the **Password** field clears and must be entered again.
5. Click **Save**.
   If there are errors when you save, the edits won't save and you must address the errors before saving again. If everything saves without errors, a success message appears. You're redirected to the **Connection Details** section, where a connection test automatically runs. If the connection test is successful, another success message appears. If the connection test isn't successful, a failure message appears. The **Status** and **Last Tested On** fields are updated automatically, so you always know the status, date, and time of your most recent test.

Automation Lightning App: Delete a Connection

Delete a connection to an external service.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To delete a connection:

- Manage Integration Connections

When you delete a connection, you remove it from all flows that use it. So, before you delete a connection, make sure it isn't used in any flows. If you delete a connection that's being used by a flow, you can get errors. You can't undo a connection deletion.

1. Select the connection to delete.
2. In the connection details page, click **Delete**.
3. After reading the popup, click **Delete**.

   If your connection deletion is successful, a success message appears and you're redirected to the Connections tab. If the deletion fails, a popup message appears.

Automation Lightning App: Orchestrations

The Orchestrations tab includes a number of standard orchestration list views and any list views that you create. Search for keywords in orchestration labels or descriptions. Filter or sort orchestrations by fields like trigger type, status, or last modified date. View and manage related orchestration runs for a selected orchestration.

Each orchestration has an associated status.

Orchestration Status	Description
Active	The current version of the orchestration has been activated.
Draft	The current version of the orchestration hasn't been activated and contains errors.
Inactive	The current version of the orchestration hasn't been activated or has been deactivated.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

View Active Orchestration Version Details

View details for an orchestration’s active version.

View All Orchestration Versions

View details about any version of an orchestration.

View Orchestration Runs Related to an Orchestration

View orchestration run records that are related to an orchestration. You can also view the orchestration stage run, orchestration step run, and orchestration run log records related to a selected orchestration run.

Manage Orchestration Runs

Cancel, debug, suspend, and resume orchestration runs from the Automation Lightning app.

View Active Orchestration Version Details

View details for an orchestration's active version.

To make this app available to anyone with the permission to use it, the **Enable the Automation Lightning App** process automation setting must be selected.

To view information about an orchestration's active version, you must have at least one active orchestration.

1. In the Automation Lightning app, select the **Orchestrations** tab.
2. Click a hyperlinked active version name.
   Details for the active orchestration version are displayed.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To view the Orchestrations tab in the Automation Lightning app

- View Orchestration in Automation App

View All Orchestration Versions

View details about any version of an orchestration.

To make this app available to anyone with the permission to use it, the **Enable the Automation Lightning App** process automation setting must be selected.

To view details about a version of an orchestration, you must have at least one orchestration.

1. In the Automation Lightning app, select the **Orchestrations** tab.
2. Click the hyperlinked orchestration label.
3. Click the **Versions** tab and then click a hyperlinked orchestration version name.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To view the Orchestrations tab in the Automation Lightning app

- View Orchestration in Automation App

View Orchestration Runs Related to an Orchestration

View orchestration run records that are related to an orchestration. You can also view the orchestration stage run, orchestration step run, and orchestration run log records related to a selected orchestration run.

To make this app available to anyone with the permission to use it, the **Enable the Automation Lightning App** process automation setting must be selected.

To view orchestration runs associated with an orchestration, you must have an active orchestration that's been run at least one time.

1. In the Automation Lightning app, select the **Orchestrations** tab.
2. Click a hyperlinked orchestration label.
   Details for the selected orchestration are displayed with a list of orchestration runs related to the selected orchestration.
3. Click a hyperlinked orchestration run name.
   Details for the selected orchestration run are displayed.
4. Click the **Related** tab.
   A list of orchestration stage runs, orchestration step runs, and orchestration run log entries related to the selected orchestration run are displayed.
5. To open the full list of any displayed items, click **View All** for that section.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To view the Orchestrations tab in the Automation Lightning app

- View Orchestration in Automation App

Manage Orchestration Runs

Cancel, debug, suspend, and resume orchestration runs from the Automation Lightning app.

User Permissions Needed

To view the Orchestrations tab in the Automation Lightning app	• View Orchestration in Automation App OR • Orchestration Process Manager permission set
To cancel, debug, resume, or suspend an orchestration run	• Manage Orchestration Runs OR • Manage Orchestration Runs and Work Items

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

To make this app available to anyone with the permission to use it, the **Enable the Automation Lightning App** process automation setting must be selected.

The available management options depend on the status and age of the orchestration run.

- Cancel, debug, and pause an in-progress orchestration run.
- Debug an orchestration run that failed within the last 14 days.
- Cancel, debug, and resume a suspended orchestration run.
- Resume an orchestration run that failed within the past 14 days because of an error in an action called by a step.

1. To manage an orchestration run from the Runs tab of an Orchestration:
   1. In the Automation Lightning app, select the **Orchestrations** tab.
   2. Click a hyperlinked orchestration label.
      Details for the selected orchestration are displayed with a list of orchestration runs related to the selected orchestration.
   3. Select the desired action from the dropdown of the orchestration run to manage.
2. To manage an orchestration run from the Details tab of an Orchestration Run:
   1. In the Automation Lightning app, select the **Orchestrations** tab.
   2. Click a hyperlinked orchestration label.
      Details for the selected orchestration are displayed with a list of orchestration runs related to the selected orchestration.
   3. Click a hyperlinked orchestration run name.
      Buttons for available management actions appear on the Orchestration Runs title bar.
   4. Click the button for the desired action.

Automate Tasks with Flows

Automation can help your company reduce repetitive tasks, increase speed and accuracy, and do more with less. With Flow Builder, create simple or complex automations, forms, and guided user experiences with little to no code. Then, add what you create to the Salesforce app, experience, or portal where you need it.

Flow Builder Tour

Get to know the Flow Builder user interface.

Plan for Success in Flow Builder

Before you begin building and distributing flows, take the time to plan your approach and incorporate good Flow Builder methods so you minimize errors and rework.

Flow Concepts

If you're new to flows, in need of a review, or curious, dive in and learn what a flow is made of and how it's different from workflow rules.

Build a Flow

After you model the process that you want to automate, design and build the flow in Flow Builder.

Testing Your Flow

Before you activate a data cloud-triggered flow or a record-triggered flow, test it to quickly verify its expected results and identify flow run-time failures.

Distribute a Flow

After you've designed and tested your flow, it's time to put it to work! Flows can be executed in several ways, depending on who the flow is designed for. Internal users, external users, or systems can run a flow, or a flow can be deployed for another organization.

Flow Interviews

A *flow interview* is a running instance of a flow. A *flow* is an application built by your administrator that asks you for inputs and does something in Salesforce based on those inputs.

Monitor Flows and Processes

Monitor your org's usage of flows and processes. See a list of paused interviews and scheduled actions from processes. Control who can view and how they view monitoring information for flows and processes.

Troubleshoot Flow Errors

If a flow interview fails, Salesforce sends an email to either the admin who last modified the associated flow or the Apex exception email recipients. The email includes the error message, details about each flow element that the interview executed, and a link to view the failed flow interview in Flow Builder.

Flow Limits and Considerations

When designing, managing, and running flows, consider the permissions, use limits, and data issues.

Flow Reference

Bookmark this page for quick access to information about flow elements, resources, events, and more.

SEE ALSO:

Choose Which Salesforce Flow Feature to Use

Build Blocks of Flows

Browser Requirements

Let Einstein Help You Build Flows (Beta)

Flow Builder Tour

Get to know the Flow Builder user interface.

EDITIONS

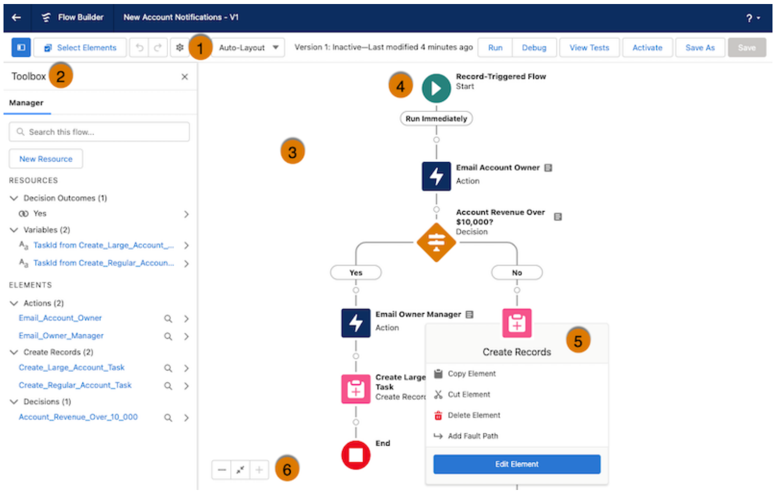
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow



Button Bar (1): Manage a flow as you build it.

- To copy multiple elements in auto-layout, click **Select Elements**, then click + on each element that you want to select. To copy all selected elements to the clipboard, click 📋 . To paste the elements you copied, click ⊕, and then select **Paste Elements**. The Paste Elements option displays the number of elements you copied. For example, Paste 3 Elements.
- To duplicate an element in free-form, select the element and click ⧉ .
- To undo a change to your flow, click ↶ . To redo a change, click ↷ .
- To show or hide the Toolbox in auto-layout, click ◧ .
- To access the flow version's properties, such as context and API version, click ⚙ .
- Choose between **Auto-Layout** or **Free-Form**. Auto-Layout is the default for all new flows.
- If the flow has warnings or errors, the Show Warnings icon (⚠) or the Show Errors icon (⊘) appears. To see details, click the icon.
- To run the most recent save of the flow version that's open, click **Run**. If the flow version contains Subflow elements, the active version of its referenced flow is executed. If the referenced flow has no active version, then the Subflow element runs the latest version of its referenced flow.
- Next to the Run button, you can see the version's active or inactive status and when it was last saved.
- To test the flow that's open, click **View Tests**, and then click **Create**. Set test parameters and assertions.
- To debug the most recent save of the flow version that's open, click **Debug**. Define values for input variables, roll back changes so debugging doesn't impact data, view debug details about how the flow runs, and use those details to troubleshoot errors.
- To make the current flow version available to your users, click **Activate**. Only one version of each flow can be active at a time.
- To save your flow, click **Save**, or to save it as a new version or a new flow, click **Save As**.

Toolbox (2): View and add resources.

- To create resources, or to view a list of all elements and resources and their details, use the Manager tab.
- To view resource and element details, such as type, outputs, usage, and incoming go to connections, click > for the resource or element.
- To add elements in free-form, use the Elements tab.

Canvas (3): Build a flow on the canvas.

As you add elements to the canvas, you can see a diagram of your flow.

Start Element (4): The Start element represents the start of a flow.

The flow executes the elements in order following the Start element. In Record-Triggered flows and Schedule-Triggered flows, configure the Start element to indicate when you want the flow to start running.

Elements (5): Elements are the building blocks of a flow.

- To add elements in auto-layout, click ⊕ where you want the element and select an element from the menu, such as Create Records or Update Records.
- To see an element's description in an element menu in auto-layout, hover over ⓘ.
- To add elements in free-form, in the Elements tab in the Toolbox, drag an element onto the canvas and connect it to the rest of the flow.
- To edit or see options for an element, click the element.
- To see an element's user-provided description on the canvas in auto-layout, hover over ▤ next to the element.

Zoom Button Bar (6): Use this button bar to zoom in and out of a flow.

Flow Builder Keyboard Shortcuts
Use these handy keyboard shortcuts for macOS and Windows to quickly navigate flows.

SEE ALSO:

Flow Elements
Flow Resources
Browser Requirements
Flow Builder Keyboard Shortcuts

Flow Builder Keyboard Shortcuts

Use these handy keyboard shortcuts for macOS and Windows to quickly navigate flows.

Use these handy keyboard shortcuts for macOS and Windows to quickly navigate flows.

Action	Layout	macOS	Windows
Zoom in	Both	Cmd+Option++	Ctrl+Alt+=
Zoom out	Both	Cmd+Option+-	Ctrl+Alt+-
Zoom to fit	Both	Cmd+Option+1	Ctrl+Alt+1
Zoom to view	Both	Cmd+Option+0	Ctrl+Alt+0

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Action	Layout	macOS	Windows
Select multiple elements on the canvas	Free-Form	Shift+Click	Shift+Click
Delete elements on the canvas	Free-Form	Delete	Backspace
View the description for an element or resource that's in a menu or on the canvas	Auto-Layout	Cmd+i	Ctrl+i
Switch panel focus	Both	F6	F6
Switch focus between tips and toolbox	Both	g and then d	g and then d
View available keyboard shortcuts	Free-Form	Cmd+/	Ctrl+/

Plan for Success in Flow Builder

Before you begin building and distributing flows, take the time to plan your approach and incorporate good Flow Builder methods so you minimize errors and rework.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Plan out your flow before you start building.

Write or draw out all the details of your business process. That way, you have a clear idea of what information you need, where you're getting that information from, and what logic and actions to perform. Doing so makes building the corresponding flow easier.

Build your flows in a test environment—like a sandbox or Developer Edition org.

The last thing you want to do is accidentally change records in your company's production org. Build your flows in a separate environment. That way, you can enter fake data and test various versions of your flow without changing or deleting production data.

Never hard-code Salesforce IDs.

IDs are org-specific, so don't hard-code new or existing IDs. Instead, let Salesforce create the IDs, and pass them into variables when the flow starts. You can do so, for example, by using merge fields in URL parameters or by using a Get Records element.

Wait until the end of the flow to edit the database.

Have you heard about flow limits? Because flows operate under Apex governor limits, the sky isn't the limit. To avoid hitting those limits, we recommend bunching all your database changes together at the end of the flow, whether those changes create, update, or delete records. And avoid making edits in a loop path. Also, for smooth finish behavior, don't create records before the first screen in the flow. You don't want any CRUD operations running unintentionally.

Avoid creating records before the first screen in the flow.

You ensure a smooth finish behavior. Besides, you don't want any CRUD operations from running unintentionally.

Control when running users can navigate backward.

If the flow commits changes to the database or performs actions between two screens, don't let users navigate from the later screen to the previous screen. Otherwise, the flow can make duplicate changes to the database.

Provide an error handler.

Sad to say, but sometimes a flow doesn't perform an operation that you configured it to do. Perhaps the flow is missing crucial information, or the running user doesn't have the required permissions. By default, the flow shows an error message to the user and emails the admin who created the flow. However, you can control that behavior. See Customize What Happens When a Flow Fails for more information and recommendations.

Save early and often.

Sometimes Flow Builder falls victim to unexpected problems, like losing Internet access. Salesforce doesn't save your changes automatically, so it's up to you to save your work. Save as often as possible, so that you don't accidentally lose a few hours' worth of work.

Restrict data access via profiles and permission sets.

Make sure that only the right flow users can access data at the right time. To restrict access to enabled profiles or permission sets, edit the access for a flow from the Flows list view.

Control data access with the Subflow element.

When a flow must respect user permissions for some actions and ignore user permissions for other actions, you can use the Subflow element. To respect user permissions for actions, you create a flow that runs in user context. To ignore user permissions such as object permissions and field-level access for other actions, you create a separate flow that runs in system context. To put it all together, you can use the Subflow element in the flow that runs in user context to launch the flow that runs in system context.

Test as many permutations of your flow as you possibly can.

As with all customizations in Salesforce, it's important to test your work, especially if your flow uses branching or other complex logic. Make sure that you test as many possibilities as you can think of before you distribute the flow to your users.

Avoid accessing external objects after DML operations in a transaction.

If Salesforce creates, updates, or deletes data in your org and then accesses external data in the same transaction, an error occurs. In your flow, we recommend using a separate transaction to access data in an external system. To do so, end the prior transaction by adding a screen or local action to a screen flow or a Wait element to an autolaunched flow. If you use a Wait element, don't use a record-based resume time.

For example, a screen flow creates a contact and then displays a confirmation screen. Next, the flow updates the contact in the external system. The flow doesn't fail because it uses a separate transaction to access the external data.

Avoid mixing Apex Data Manipulation Language (DML) operations on setup objects, other Salesforce objects, and external objects in a transaction.

A single transaction can't mix Apex DML operations on objects (such as Account), setup objects (such as User Role), and external objects. For example, if Salesforce updates an account and a user role in a single transaction, an error occurs.

The best approach is to use a separate transaction. To end a transaction, add a screen or local action to a screen flow or a Wait element to an autolaunched flow. If you use a Wait element, don't use a record-based resume time.

Avoid custom CSS whenever possible.

Use custom CSS sparingly and avoid targeting DOM elements in components that you don't own. Changes to a component's internal DOM structure are likely to break hard-coded CSS selectors. Salesforce can change the internal implementation of Salesforce components at any time. Additionally, Salesforce Customer Support can't help resolve issues with custom CSS.

SEE ALSO:

Data Safety When Running Screen and Autolaunched Flows in System Context

Flow Concepts

If you're new to flows, in need of a review, or curious, dive in and learn what a flow is made of and how it's different from workflow rules.

Build Blocks of Flows

Use combinations of elements, connectors, and resources to build flows.

Flow Types

A flow or flow version's type determines which elements and resources you can add to a flow and how you can distribute the flow. For example, you distribute a screen flow on a website to gather information from visitors, or you distribute an autolaunched flow using a custom button that users click to run the flow in the background. Similarly, you can add a Custom Error element only in a record-triggered flow, whereas you can add a Choice resource type only in a screen flow.

Triggers for Autolaunched Flows

In the Start element of an autolaunched flow, you can specify a trigger that launches the flow. The flow trigger can be a schedule or the new and changed records of a specified object. Without a trigger, you must set up other things to launch the flow, such as custom buttons, processes, Apex classes, or even Einstein Bots.

Template-Triggered Prompt Flows

Provide data to prompt templates in Prompt Builder with template-triggered prompt flows.

Advanced Flow Concepts

After you understand the basics, you're ready for a closer look at what you can do after a flow finishes. Learn what happens when a flow interacts with the Salesforce database, and how flows perform similar operations at the same time. Get to know how a flow test saves you time and improves reliability.

SEE ALSO:

Trailhead: Flow Basics

Trailhead: Build a Simple Flow

Trailhead: Flow Builder

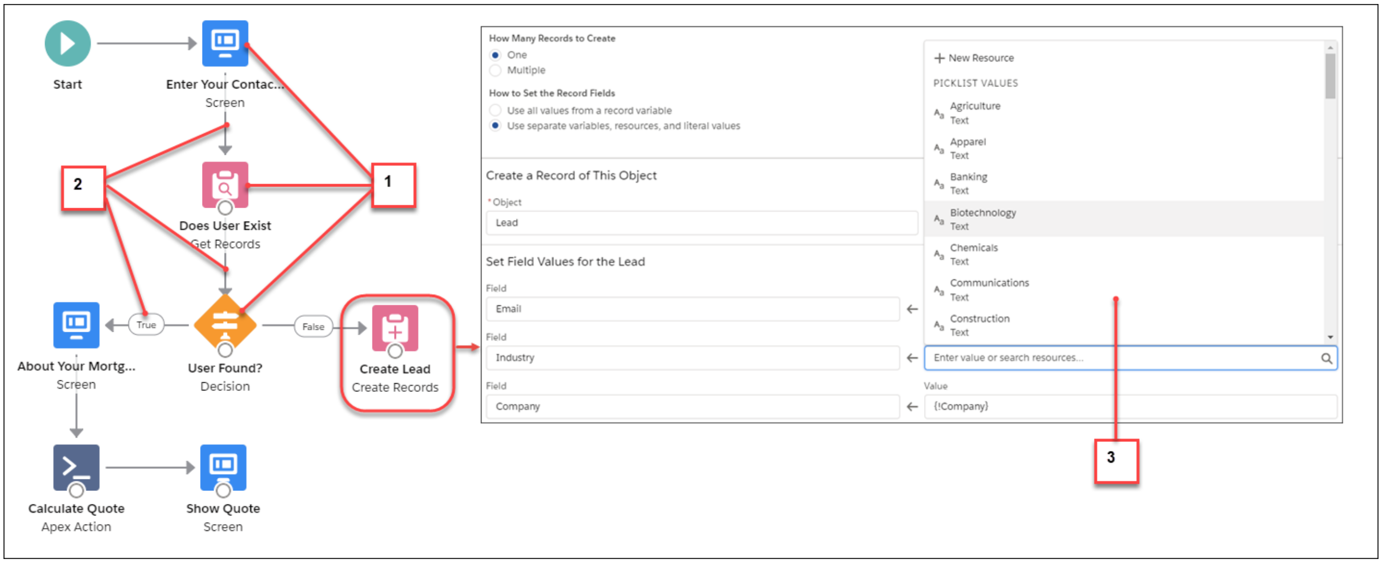
Build Blocks of Flows

Use combinations of elements, connectors, and resources to build flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



How Many Records to Create

● One

○ Multiple

How to Set the Record Fields

○ Use all values from a record variable

● Use separate variables, resources, and literal values

Create a Record of This Object

* Object: Lead

Set Field Values for the Lead

Field: Email ←

Field: Industry ← Enter value or search resources...

Field: Company ← Value: {!Company}

+ New Resource

PICKLIST VALUES

Agriculture Text

Apparel Text

Banking Text

Biotechnology Text

Chemicals Text

Communications Text

Construction Text

- Each element (1) represents an action that the flow can execute. Examples include reading or writing Salesforce data, displaying information to and collecting data from flow users, executing logic, or manipulating data.
- Each connector (2) defines an available path that the flow can take at run time.
- Each resource (3) represents a value that you can reference throughout the flow.

SEE ALSO:

Flow Elements

Flow Resources

Flow Connectors

Flow Types

A flow or flow version's type determines which elements and resources you can add to a flow and how you can distribute the flow. For example, you distribute a screen flow on a website to gather information from visitors, or you distribute an autolaunched flow using a custom button that users click to run the flow in the background. Similarly, you can add a Custom Error element only in a record-triggered flow, whereas you can add a Choice resource type only in a screen flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Standard Flow Types

These flow types are supported in Flow Builder.

Type	Description	Available Distribution Methods	Supported in Translation Workbench
Action Cadence Autolaunched Flow	Launches after a sales step is completed. This type of flow runs in the background without user interaction.	Sales Cloud	✓
Action Cadence Step Screen Flow	Launches from a cadence step. This type of flow collects or displays information and requires user interaction.	Sales Cloud	✓
Autolaunched No Trigger Flow	Launches when invoked by a supported distribution method. This autolaunched flow runs in the background and doesn't require user interaction. This flow type doesn't support screens, local actions, choices, or choice sets.	• Processes • Custom Apex classes • REST API • Flow Orchestration • Web tabs • Custom buttons or custom links • Visualforce pages	✓
Autolaunched No Trigger Orchestration	Launches when invoked by Apex, REST API, custom buttons, or custom links. With an orchestration, you can create a multi-step, multi-user process.	Flow Orchestration	✓
Automation Event-Triggered Flow	Launches when an automation event such as a form submission is received. This type of flow runs in the background without user interaction.	Starter or Marketing Cloud Growth • Automations Lightning app • Marketing 360 Lightning app on the Flows tab	✗

Type	Description	Available Distribution Methods	Supported in Translation Workbench
Cart Async Autolaunched Flow	Launches when invoked by a cart change, such as an add to cart. This type of flow runs in the background without user interaction.	Commerce Cloud	✓
Checkout Screen Flow	Create a screen flow that implements a Commerce Cloud checkout process. This type of flow collects or displays information and requires user interaction.	Use this Experience Builder component to add this flow to your store. • Checkout	✓
CMS Workflow Orchestration Autolaunched Flow	Launches when invoked from the Workflows component in a CMS workspace. With this type of orchestration, you can create a multi-step, multi-user process to create, edit, organize, and manage digital content from a centralized location. Includes the `mContentVariantId` and `mContentSpaceId` input variables. This type of flow runs in the background without user interaction. For more details, see CMS Workflows and Approvals.	Salesforce CMS in the Digital Experiences app	✓
Contact Request Screen Flow	Launches when invoked by a customer. With it, a customer can enter contact details into a self-service form. This type of flow collects or displays information and requires user interaction.	Use one of these Experience Builder components to add this flow. • Contact Request Button & Flow—Launch the flow in a window. • Flow—Embed the flow directly on the page.	✓
Data Capture Flow (Beta)	Launches when invoked by a user in the Field Service mobile app. Create, update, and pre-fill records based on data from related records in the flow. This type of flow requires user interaction.	Field Service mobile app	✓
Evaluation Autolaunched Flow	Launches when invoked by an orchestration to evaluate custom criteria for a stage or step. To indicate that the criteria are met, set the output variable `isOrchestrationConditionMet` to True, which discards all other output variables. This type of flow runs in the background without user interaction.	• Orchestration stage custom exit condition • Orchestration interactive step custom entry or exit condition • Orchestration background step custom entry condition	✓
Event-Driven Flow	Launches when an event is received. This type of flow runs in the background without user interaction. Requires org and user permissions.	Starter, Pro Suite, or Marketing Cloud Growth • Automations Lightning app	✓

Type	Description	Available Distribution Methods	Supported in Translation Workbench
	Requires Flows: Lightning App and Flow Builder: Citizen flows org permissions and View Flows and Create or Edit Flows user permissions.	• Marketing 360 Lightning app on the Flows tab	
Field Service Mobile Flow	Launches when invoked by a user in the Field Service mobile app. This type of flow collects or displays information and requires user interaction.	Field Service mobile app	✓
Field Service Web Screen Flow	This flow type launches from the Field Service app. With it, users can schedule, modify, or cancel an appointment in a web browser. This type of flow requires user interaction.	Embedded Appointment Booking	✓
Form-Triggered Flow	Launches when a web visitor submits a marketing form. This type of flow runs in the background without user interaction. Requires Flows: Lightning App and Flow Builder: Citizen flows org permissions and View Flows, Create or Edit Flows, and Create or Modify Form-Triggered Flows user permissions.	Marketing Cloud Growth • Automations Lightning app • Marketing 360 Lightning app on the Flows tab	✗
Indicator Result Screen Flow	Launches a screen flow when initiated by a user to calculate and create indicator results for a selected indicator performance period.	Outcome Management	✓
Individual-Object Linking Screen Flow	Launches when invoked by an agent to link a record like a case or messaging session to a contact, lead, person account, or employee. This type of flow collects or displays information and requires user interaction.	Service Cloud	✓
Loyalty Management Autolaunched Flow	Launches when invoked by a loyalty program process, and contains Assignment, Decision, and Action elements. The Action element in this type of flow supports Apex and quick actions. This type of flow runs in the background without user interaction.	Loyalty Management app	✓
Mortgage Lending Screen Flow	Launches when invoked by a user and allows them to provide Financial Service Cloud mortgage application details. This type of flow requires user interaction.	Financial Service Cloud	✓
Platform Event Triggered Flow	Launches when a platform event occurs. This type of flow runs in the background without user interaction.	Platform Events in Setup	✓

Type	Description	Available Distribution Methods	Supported in Translation Workbench
Prompt Template Capability-Triggered Flow	Launches from a prompt template. Adds prompt instructions to the associated prompt template. This type of flow adds content to the associated prompt template and runs in the background without user interaction.	Prompt Builder	✓
Recommendation Strategy Autolaunched Flow	Builds a personalized list of recommendations for users. When a user responds to a recommendation, that recommendation launches its assigned flow. Used by Einstein Next Best Action.	• Einstein Next Best Action component on Lightning pages • Suggested Actions component on Experience Cloud pages • Visualforce pages	✓
Record-Triggered After Save Flow	Launches and runs in the background after a record change is saved or a record is deleted.	A record-triggered after save flow runs only when a record is saved, can make changes only to records related to the triggering record, and runs in the background without user interaction.	✓
Record-Triggered After Save Orchestration	Launches when a record is created or updated. With an orchestration, you can create a multi-step, multi-user process. This type of flow runs in the background without user interaction. See Orchestration Types for Flow Orchestration availability.	• Autolaunched orchestrations – Custom Apex classes – Custom buttons or custom links • Record-triggered orchestrations run only when a record is created or updated.	✓
Record-Triggered Before Delete Flow	Launches when a record is deleted. This type of flow runs in the background without user interaction.	A record-triggered before delete flow runs when a record is flagged for deletion and runs in the background without user interaction.	✓
Record-Triggered Before Save Flow	Launches after a record is created or updated, but hasn't been saved. Only these elements are supported: Assignment, Decision, Get Records, and Loop.	A record-triggered before save flow runs only when a record is created or updated, can make changes only to the triggering record, and runs in the background without user interaction.	✓
Schedule-Triggered Flow	Launches and runs in the background at a specified time and frequency for each record in a batch. This flow type doesn’t support user interaction, screens, local actions, choices, or choice sets.	A schedule-triggered flow runs only at the scheduled time and frequency.	✓
Screen Flow	Guides users through a business process that is launched from a supported distribution method. This flow type requires user interaction because it includes screens, local	• Flow actions • Lightning pages • Experience Builder pages	✓

Type	Description	Available Distribution Methods	Supported in Translation Workbench
	actions, steps, choices, or dynamic choices. Screen flows don't support Wait elements.	• Custom Aura components • Custom Lightning web components • Custom buttons or custom links • Flow Orchestration • Web tabs • Direct flow URLs • Visualforce pages • Lightning Out • Embedded Service deployments	
Segment-Triggered Flow	Launches when activated or when scheduled for qualified Data Cloud segment members. This type of flow runs in the background without user interaction. Requires org and user permissions. Requires Flows: Lightning App and Flow Builder: Citizen flows org permissions and View Flows, Create or Edit Flows, and Create or Modify Segment-Triggered Flows user permissions.	Starter, Pro Suite, or Marketing Cloud Growth • Automations Lightning app • Marketing 360 Lightning app on the Flows tab	✗
User Provisioning Screen Flow	Launches when invoked by a user. With it, users can create a user account and link it to a third-party service or app. This type of flow requires user interaction. For example, use this flow type to customize the user provisioning configuration for a connected app to link Salesforce users with their Google Apps accounts.	A user provisioning flow can only be implemented by associating it with a connected app when running the User Provisioning Wizard.	✗

Other Flow Types

Not all flow types are supported in Flow Builder. Some flow types are used only in other parts of Salesforce, so they're not listed in the Flows page in Setup. However, the list of paused flow interviews can include these types.

Type	Description	Supported in Translation Workbench
Customer Lifecycle Record-Triggered After Save Flow	Launches after a customer lifecycle map is saved. This flow type changes records related to the triggering record. This type of flow runs in the background without user interaction.	✓

Type	Description	Supported in Translation Workbench
Data Cloud Data Change Flow	Launches when a record from a Data Cloud data model object or a calculated insight object is changed and meets the specified conditions. This type of flow runs in the background without user interaction.	✓
Digital Form Screen Flow	Launches from app extensions. This type of flow collects or displays information and requires user interaction.	✓
Employee Service Catalog Item Screen Flow	Launches when invoked by a user. With it, users can browse and order items from the Employee Service Catalog. This type of flow collects or displays information and requires user interaction.	✗
Process Builder Autolaunched Process	Launched when invoked from a Process Builder process. This type of flow runs in the background without user interaction.	✗
Process Builder Custom Event Process	A Process Builder process that launches when a custom event message is received. This type of process runs in the background without user interaction.	✗
Process Builder Workflow	Launches when a record is created or updated. This type of process runs in the background without user interaction.	✗
Survey Enrich Autolaunched Flow	Launches in the context of a survey response and can't execute without a corresponding survey. Use it to conditionally map a response to a record, create records, or send notifications. This type of flow runs in the background without user interaction.	✗
Transaction Security Autolaunched Flow	Used by Condition Builder to declaratively build customized security policies to protect data. This type of flow runs in the background without user interaction.	✗
Managed Content Autolaunched Flow	Launches when invoked by the ManagedContentRelease translator. This type of flow runs in the background without user interaction.	✓
Routing Autolaunched Flow	Launches when a customer initiates a chat, voice, or messaging conversation and routes the work item to a queue, skill, agent, or bot. This type of flow runs in the background without user interaction.	✗

Type	Description	Supported in Translation Workbench
Scheduler Appointments Screen Flow	Launches when invoked by a user. With it, users can schedule appointments in Salesforce Scheduler. This type of flow collects or displays information and requires user interaction.	✓

SEE ALSO:

Flow Version Properties

User Provisioning for Connected Apps

Triggers for Autolaunched Flows

Triggers for Autolaunched Flows

In the Start element of an autolaunched flow, you can specify a trigger that launches the flow. The flow trigger can be a schedule or the new and changed records of a specified object. Without a trigger, you must set up other things to launch the flow, such as custom buttons, processes, Apex classes, or even Einstein Bots.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Manage Record-Triggered Flows

See, manage, reorder, and filter your record-triggered flows with Flow Trigger Explorer. Flow Trigger Explorer shows all the flows that are associated with a specified object and that run when a record is created, updated, or deleted. You can navigate between a flow in Flow Builder and other flows that run under the same circumstances.

Schedule Triggers for Flows That Run for Batches of Records

A schedule-triggered flow starts at the specified time and frequency for a batch of records. Configure the schedule trigger in the Start element of your autolaunched flow.

Record Triggers for Flows That Make Before-Save Updates

Creating or updating a record can trigger an autolaunched flow to make additional updates to that record before it's saved to the database. A record-triggered flow can update a Salesforce record 10 times faster than a record-change process. Configure the record trigger in the Start element of your autolaunched flow.

Define the Run Order of Record-Triggered Flows for an Object

Specify a trigger order value to determine the run order of before-save or after-save flows for the same object.

Scheduled Paths

For record-triggered flows, you can use scheduled paths to run part of a flow at a dynamically scheduled time after a triggering event.

Trigger Flows with Data Cloud Data

A Data Cloud-triggered flow starts when conditions are met in a data model object (DMO) or calculated insight object (CIO). Select the Data Cloud object and the conditions in the Start element of your autolaunched flow.

SEE ALSO:

Apex Developer Guide: Triggers and Order of Execution

Manage Record-Triggered Flows

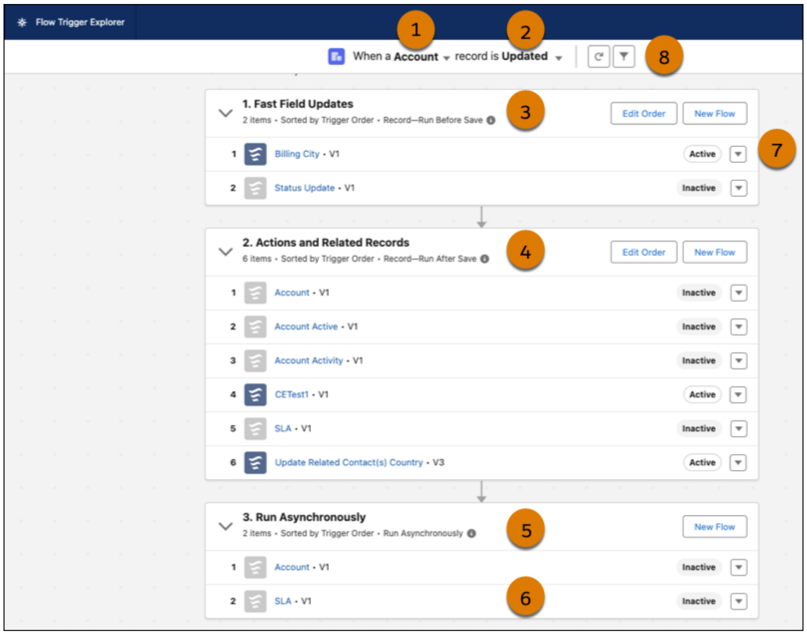
See, manage, reorder, and filter your record-triggered flows with Flow Trigger Explorer. Flow Trigger Explorer shows all the flows that are associated with a specified object and that run when a record is created, updated, or deleted. You can navigate between a flow in Flow Builder and other flows that run under the same circumstances.

In Flow Trigger Explorer, select an object (1) and a trigger (2) to see flows that run when a record for that object is created, updated, or deleted. View all the before-save (3) and after-save flows (4) that run for the same object and trigger. Check the asynchronous paths (5) that are associated with the flows that run for the same object and trigger. To open the flow in Flow Builder in a new tab, click the flow label (6). In the dropdown list, select Flow Details and Versions (7) to see details and activate or deactivate versions of the flow. To open the filter panel, click the filter icon (8). You can filter by status, package state, or process type.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



When a **Account** record is **Updated**

1. Fast Field Updates
2 items • Sorted by Trigger Order • Record—Run Before Save

1	Billing City • V1	Active
2	Status Update • V1	Inactive

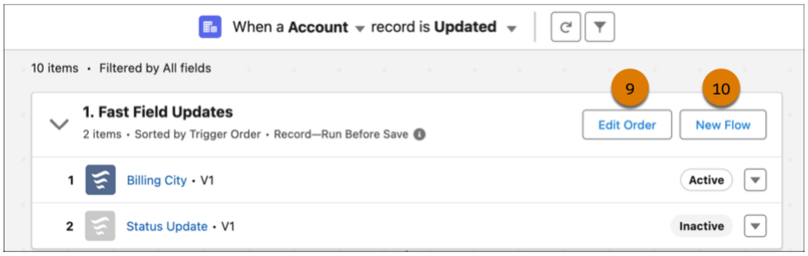
2. Actions and Related Records
6 items • Sorted by Trigger Order • Record—Run After Save

1	Account • V1	Inactive
2	Account Active • V1	Inactive
3	Account Activity • V1	Inactive
4	CETest1 • V1	Active
5	SLA • V1	Inactive
6	Update Related Contact(s) Country • V3	Active

3. Run Asynchronously
2 items • Sorted by Trigger Order • Run Asynchronously

1	Account • V1	Inactive
2	SLA • V1	Inactive

To change the run order, click **Edit Order** (9) and drag the flows to the order you want. Reordering changes the flow's Trigger Order value. The Trigger Order value determines the flow run sequence for an object's flows with the same trigger type. You can see when a Standard flow executes in relation to your other record-triggered flows, but you can't reorder it. To create a flow with the same object and record trigger type, click **New Flow** (10).



When a **Account** record is **Updated**

10 items • Filtered by All fields

1. Fast Field Updates
2 items • Sorted by Trigger Order • Record—Run Before Save

1	Billing City • V1	Active
2	Status Update • V1	Inactive

Limitations

You can see when a Standard flow executes in relation to your other record-triggered flows, but you can't reorder it.

Record-triggered flows created with a flow template don't appear in Flow Trigger Explorer.

SEE ALSO:

Guidelines for Defining the Run Order of Record-Triggered Flows for an Object

Schedule Triggers for Flows That Run for Batches of Records

A schedule-triggered flow starts at the specified time and frequency for a batch of records. Configure the schedule trigger in the Start element of your autolaunched flow.

Schedule an autolaunched flow to start at a specific time and set it to run one time, daily, or weekly. To have the scheduled flow run for a batch of records, specify the object and the filter conditions that each record must meet.

A flow interview runs for each record in the batch and stores all of the record's field values in the `$Record` global variable. As you build the scheduled flow, reference the `$Record` global variable to access the record's field values.

If you configure an Update Records element to use the ID and all field values from the $Record global variable, enable `Filter inaccessible fields from flow requests` in your org's process automation settings. Otherwise, the flow fails because the Update Records element tries to set the values for system fields and other read-only fields.

You can monitor scheduled flows from the Scheduled Jobs page in Setup.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Schedule-Triggered Flow Considerations

Considerations for Troubleshooting Flows

Record Triggers for Flows That Make Before-Save Updates

Creating or updating a record can trigger an autolaunched flow to make additional updates to that record before it's saved to the database. A record-triggered flow can update a Salesforce record 10 times faster than a record-change process. Configure the record trigger in the Start element of your autolaunched flow.

Perhaps you're familiar with Process Builder and using a record-change process to make additional updates to each record that triggers the process. Before-save updates in flows accomplish that same goal, but much more quickly because each record doesn't get saved to the database again. Avoiding that extra save procedure means skipping another round of assignment rules, auto-response rules, workflow rules, and other customizations that take time to execute.

Perhaps you're familiar with Apex triggers. A flow that makes before-save updates is similar to a `before` trigger. The previously described record-change process is similar to an `after` trigger. In a save procedure, before-save updates in flows are executed immediately before Apex `before` triggers.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Because of their speed, we recommend that you use before-save updates in flows to update fields on new or changed records. However, sometimes you must use a record-change process or an Apex `after` trigger to:

- Access field values that are set only after the record is saved, such as the Last Modified Date field or the ID of the new record.
- Create or update related records.
- Perform actions other than updating the record that launches the flow.

Flows that make before-save updates are typically simpler to build than other types of flows.

- The `$Record` global variable contains the values from the record that triggers the flow to run. As a result, there's no need to add a Get Records element to obtain the record data nor create flow variables to store the record data.
- When the flow changes the values in the `$Record` global variable, Salesforce automatically applies those new values to the record. So there's no need to add an Update Records element to save the new values to the database.
- Only these elements are supported: Assignment, Decision, Get Records, and Loop. These elements let you obtain data from other Salesforce records, and use them to decide whether to update the triggering record's fields and to what values.

SEE ALSO:

Record-Triggered Flow Considerations

Apex Developer Guide : Triggers and Order of Execution

Define the Run Order of Record-Triggered Flows for an Object

Specify a trigger order value to determine the run order of before-save or after-save flows for the same object.

Declaratively configure a flow to run before or after another flow. Order your flows to help ensure consistent results without creating overly complex flows, letting you divide automation by teams or logical owners.

- When you save a before- or after-save record-triggered flow, specify a trigger order value from 1 to 2,000. If a flow is already saved, specify a trigger order value in the flow's version properties.

Guidelines for Defining the Run Order of Record-Triggered Flows for an Object

Learn guidelines for configuring a record-triggered flow to run before or after another record-triggered flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Guidelines for Defining the Run Order of Record-Triggered Flows for an Object

Learn guidelines for configuring a record-triggered flow to run before or after another record-triggered flow.

- You can define a trigger order value only for before-save or after-save flows on an object. Trigger order affects only the specified object's flows with the same trigger type.
- Trigger order values always respect order of execution rules. For example, you can make an after-save flow run before other after-save flows, but you can't make an after-save flow run before a before-save flow or an Apex trigger, even if the trigger order value is lower.
- Flows with the same trigger (before- or after-save) on the same object with trigger order values from 1 to 1,000 run in ascending order (1, 2, 3, and so on). Multiple flows with the same trigger order value run in alphabetical order based on the flows' API names.
- Flows without trigger order values run next and in the order of their activation dates. Flows created in Winter '22 and earlier run in this order, unless you define a trigger order value for them.
- Flows with trigger order values from 1,001 to 2,000 run next and in ascending order. Multiple flows with the same trigger order value run in alphabetical order based on the flows' API names.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- When you order a large number of flows, a best practice is to evenly distribute the trigger order values, for example, 10, 20, 30, or 100, 200, 300. Then later, you can easily slot another flow in between, for example, between 10 and 20. This practice can help you avoid changing the trigger order values for pre-existing flows.
- Activating, deactivating, or changing the order for one flow can cause the order for other flows to automatically update. Ordering record-triggered flows has no direct effect on any associated scheduled or asynchronous paths.

SEE ALSO:

Manage Record-Triggered Flows

Apex Developer Guide: Triggers and Order of Execution

Scheduled Paths

For record-triggered flows, you can use scheduled paths to run part of a flow at a dynamically scheduled time after a triggering event.

Add a Scheduled Path to Your Flow

You can create scheduled paths for record-triggered flows that are optimized for actions and related records. You can base the scheduled time on when the record is created or updated or on a field value in the record.

Scheduled paths run in system context, so they have permission to access and modify all data. But the running user associated with the flow's actions is the user who originally changed the record.

Before you create a scheduled path, define your org's Default Workflow User. This setting tells Salesforce which user to default to if the user who triggered the flow is inactive. It's possible that the default user is already set in your production org, but it's a good idea to confirm it. You set the Default Workflow User on the Process Automation Settings page in Setup.

EDITIONS

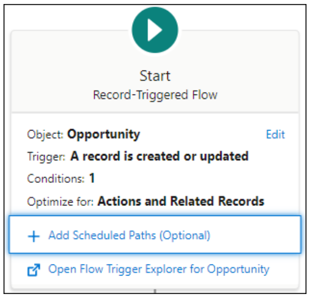
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow


Start
Record-Triggered Flow
Edit
+ Add Scheduled Paths (Optional)
Open Flow Trigger Explorer for Opportunity

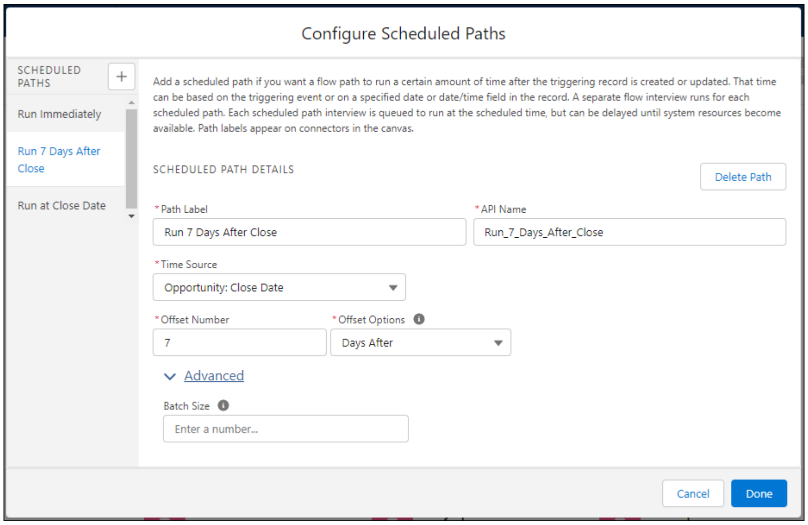
Object: **Opportunity**

Trigger: **A record is created or updated**

Conditions: **1**

Optimize for: **Actions and Related Records**

1. To add a scheduled path, click **Add Scheduled Paths (Optional)** in a flow's Start element. To add a scheduled path if the flow already contains scheduled paths, or to edit an existing scheduled path, click **Edit**.



Add a scheduled path if you want a flow path to run a certain amount of time after the triggering record is created or updated. That time can be based on the triggering event or on a specified date or date/time field in the record. A separate flow interview runs for each scheduled path. Each scheduled path interview is queued to run at the scheduled time, but can be delayed until system resources become available. Path labels appear on connectors in the canvas.

*Path Label: Run 7 Days After Close

*API Name: Run_7_Days_After_Close

*Time Source: Opportunity: Close Date

*Offset Number: 7

*Offset Options: Days After

Batch Size: Enter a number...

2. Enter a Path Label. The API Name is generated automatically.
3. Specify a Time Source for the scheduled path. That time can be based on the triggering event or on a specified date or date/time field in the record.
4. To run the scheduled flow path at a specified time before or after the selected Time Source, enter an Offset Number and select an Offset Option. Otherwise, enter 0 for the Offset Number.
5. Enter a Batch Size. The default and maximum value is 200, and the minimum is 1. The batch size is the number of records that a path can process at the same time. For example, if you specify a batch size of two and you have seven records scheduled in the same time interval, they run in four batches. Use this field to avoid hitting Apex governor limits.

View pending scheduled paths on the Time-Based Workflow Setup page. From Setup, in the Quick Find box, enter `Time-Based Workflow`, and then select **Time-Based Workflow**.

Set Batches in Scheduled Paths

Scheduled Paths can execute in batches when there's numerous paths set to execute at the same time.

All records that meet the conditions and are scheduled to be processed at the same time are grouped into batches, up to the batch size that you set. If your paths are set to execute 10 minutes apart, they aren't batched.

The batch size is the number of records that a path can process at the same time. The default and maximum value is 200, and the minimum is 1. For example, if you specify a batch size of two and seven records to be processed at the same time, Flow Builder groups them into four batches. You can set the batch size value in Advanced features. Otherwise, the default of 200 is used. Specifying a batch size for your path can help you to avoid hitting the per-transaction Apex limit.

Asynchronous limits apply to scheduled paths, making it easier to complete complex operations without hitting limits.

Use the When Record is Created or Updated Trigger

Let's look at an example where a path is scheduled to run when a record is created or updated.

When a scheduled path is configured to run with Time Source set to **When Record is Created or Updated**, the condition requirements for the flow must include one of these options.

- **Trigger the Flow When** is set to **A record is created**.
- **When to Run the Flow for Updated Records** is set to **Only when the record is updated to meet the conditions**.
- Use the **ISCHANGED** operator in the condition.

When you select the **Only when the record is updated to meet the conditions** option, the scheduled path is scheduled only when the previous version of the record didn't meet the conditions and now the updated version of the record does.

For example, configure a flow trigger on Account with **Trigger the Flow when** is set to **A record is created or updated** and a condition set to **Industry Equals Agriculture**. Set **When to Run the Flow for Updated Records** to **Only when the record is updated to meet the conditions**.

When does the scheduled path run?

Created or Updated Record	Does the Path Run?
Create an Account record with Industry equals Agriculture	Yes, because the new record meets the criteria.
Create an Account record with Industry equals Finance	No, because the new record doesn't meet the criteria.
Update an existing Account record to change the Billing City. Industry equals Agriculture.	No, because the original value of Industry was already Agriculture. Any already scheduled paths remain scheduled.
Update an existing Account record to change Industry from Finance to Agriculture.	Yes, because the updated record meets the criteria.
Update an existing Account record to change Industry from Agriculture to Finance.	No, and any already scheduled paths are canceled. It's important to remember that every time there's an update, the criteria are evaluated again to see if it's still TRUE.

Use Conditions with OR

Let's look at an example that uses the OR condition with scheduled paths.

Set a trigger on the Account object with **Trigger the Flow when** set to **A record is created or updated** and an **OR** condition set to:

- Industry equals Agriculture OR BillingState equals CA
- **When to Run the Flow for Updated Records** is set to **Only when the record is updated to meet the conditions**.

When the record is updated, does the scheduled path run?

Example:

Original Record	Updated Record	Does the Path Run?
Industry equals Agriculture Billing State equals NV	Industry equals Agriculture BillingCity equals CA	No. The original record met the condition. That BillingCity changes has no effect. The condition is met before and after.
Industry equals Finance Billing State equals NV	Industry equals Agriculture BillingCity equals NV	Yes. The original record didn't meet the condition, but the updated record does.
Industry equals Finance Billing State equals CA	Industry equals Finance BillingCity equals NV	No, and any scheduled paths are canceled because the updated record no longer meets the condition.

Limitations and Considerations

Understand the limitations and considerations when adding scheduled paths to your record-triggered flows.

Use this table as a guide when deciding between adding a scheduled path to a record-triggered flow or creating a scheduled-triggered flow.

Consideration	Scheduled Paths	Schedule-Triggered Flows
Flow Type	Record-Triggered	Schedule-Triggered
Trigger	A record is created, updated, or deleted	A specific date and time
What Time	A specified amount of time after the trigger or A specified amount of time before or after a date on the triggering record	A specific date and time
Frequency	Runs once for each time it's triggered	Can run once, daily, or weekly

Scheduled paths are only available on these types of record-triggered flows.

Trigger	No Condition	A Condition Exists
	Every time a record is updated and meets condition requirements	Only when a record is updated to meet the condition requirements
A record is created	X	X
A record is updated		X
A record is created or updated		X
A record is deleted		X

- The maximum scheduled-path interviews per 24 hours are 250,000, or the number of user licenses in your org multiplied by 200, whichever is greater. One interview is created for each executed scheduled path on a record-triggered flow. Paths that run immediately don't count toward this limit.
- If the flow can be triggered by record updates with Time Source set to be based on a field value such as `dueDate`, consider this behavior. If Time Source is set to a future time and condition requirements are met, the scheduled path executes at that new time regardless of whether the scheduled path already executed. You can use this record trigger configuration to schedule recurring actions on a record by updating the record's date field.
- If a record-triggered flow with scheduled paths is deactivated, any pending automations on the scheduled paths are canceled. The pending automations stay on the time-based workflow list until the time that they're scheduled to run. At that time, the flow is checked for activation status, and if the flow is deactivated, those pending automations are canceled.
- Scheduled paths and asynchronous paths are batched differently. In a scheduled path, all the records that meet the conditions and are scheduled to be processed in the same minute are grouped up to the batch size into one batch.
- Synchronous Apex transactions invoked by an asynchronous flow contribute to synchronous per-transaction Apex limits. Asynchronous flows include scheduled flows and flows with scheduled or asynchronous paths.

- If a scheduled path interview fails one time, the error email is sent and retried 15 minutes later. The path is retried a maximum of 5 times. So, the scheduled path is retried after 15 minutes, after 30 minutes, and then again in 60, 120, and 240 minutes. This retry count doesn't appear in the Time-Based Workflow page, but when a retry is scheduled, the Scheduled Date column shows the time of the next attempt.
- When executed in bulk, if some flow interviews are successful and others fail, the transaction is rolled back and the successful flow interviews are retried immediately. The maximum retries in this scenario is 2.
- If a path is scheduled but doesn't execute at the specified time, it can be because the execution of the path failed, potentially due to a problem in the flow itself. You receive an email informing you of the error. Or it can be because your org exceeded the rolling 24-hour limit. The flow is rescheduled and tried again.
- Scheduled path information is available in these debug log events.

Debug Log Event	Description
FLOW_SCHEDULED_PATH_QUEUED	An event is logged when a scheduled path is added to the queue after a record is created or updated.
FLOW_VALUE_ASSIGNMENT	An event is logged when a scheduled path runs.

Considerations

When a scheduled path is configured with **Every time a record is updated** and **Meets the condition requirements** and the execution time is field-based, for example, execute 1 hour before Opportunity.CloseDate, the flow behaves exactly as if the scheduled path is configured as **Only when a record is updated to meet the condition requirements**. It doesn't matter whether the corresponding entry conditions are configured for **Every time a record is updated** or **Only when a record is updated to meet the condition requirements**.

- An existing scheduled path schedule remains scheduled as long as the condition is TRUE and the date/time specified by the Time Source and Offset hasn't been reached yet.
- Changes to the time-source field cause the scheduled path to be scheduled at the new date, provided the date is in the future. This scenario occurs regardless of whether the scheduled path is currently scheduled or not.

SEE ALSO:

Record-Triggered Flow Considerations

Trailhead: Record-Triggered Flows

Trigger Flows with Data Cloud Data

A Data Cloud-triggered flow starts when conditions are met in a data model object (DMO) or calculated insight object (CIO). Select the Data Cloud object and the conditions in the Start element of your autolaunched flow.

To create a Data Cloud-triggered flow, from Setup, in the Quick Find box, enter `Flow`, and then select **Flows**. Click **New Flow**, and select **Start From Scratch**, then click **Data Cloud-Triggered Flow**, and click **Create**.

Note: If you select a DMO, the flow triggers only if the DMO is mapped as part of a data stream. External DMOs aren’t supported.

Depending on the volume of data changes, the complexity of your flow, and your data stream refresh schedule, you can encounter delays on when the Data Cloud-triggered flow is fired and processed. You can view a data stream’s refresh history on its record page.

SEE ALSO:

About Salesforce Data Cloud

Automate Flows in Data Cloud

Data Stream Schedule in Data Cloud

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view, save, and activate Data Cloud-triggered flows:

- Data Cloud Marketing Admin

  OR

  All of these permissions:

  - Manage Flow
  - View Setup
  - View All Data
  - Data Cloud User
  - Create, read, update, delete, and View All on Data Action Targets, Data Action Target Definitions, Data Actions, Data Action Definitions, Data Action Target Relations
  - Read on Data Model Object, Data Model Field, and Calculated Insights Object

Template-Triggered Prompt Flows

Provide data to prompt templates in Prompt Builder with template-triggered prompt flows.

Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

The template-triggered prompt flow performs dynamic logic and actions to generate output text that's merged into the associated prompt template's resolution.

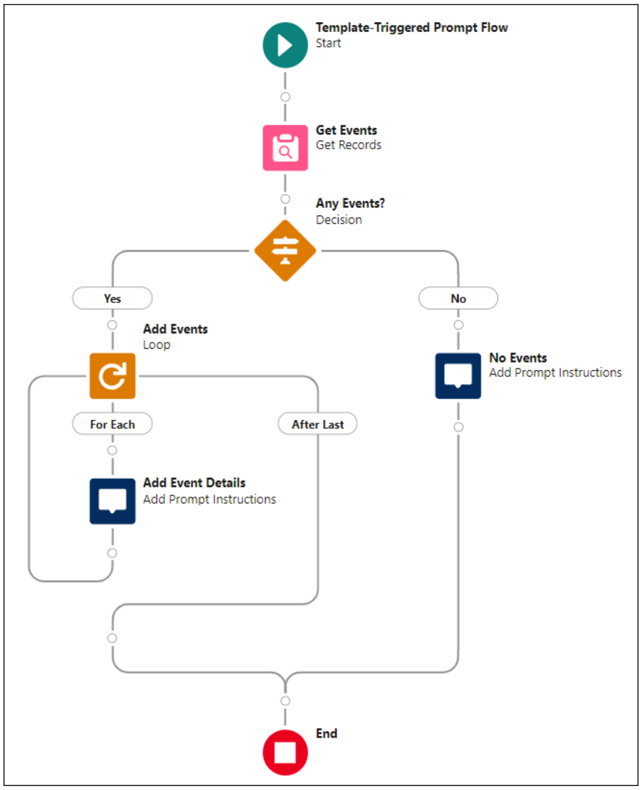
EDITIONS

Available in: Lightning Experience

Available in: **Unlimited+** Edition

Available for an additional cost in: **Enterprise** and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on.

Example: You can create a prompt template that explains how to write an invitation email for company events near a contact's location. The prompt template can trigger a flow that generates a list of company events near the contact. Depending on whether there are matching events in your Salesforce data, the prompt template adds either no events or the list of company events in the prompt template's resolution.



SEE ALSO:

- Example of Field Generation Template-Triggered Prompt Flow
- Example of Sales Email Template-Triggered Prompt Flow
- Example of Reusable Template-Triggered Prompt Flow
- *Salesforce Help*: Prompt Builder

Example of Field Generation Template-Triggered Prompt Flow

Let's create an example flow that integrates with a prompt template for generating a summary of records.

Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

Let's say you want to provide up-to-the-minute summaries of customer cases for a given customer to your company's sales team before they make customer calls. You want to create a field generation prompt template that integrates a template-triggered prompt flow. The flow obtains case data for an LLM to summarize.

Before creating your field generation prompt template, you create a flow that retrieves a list of open customer cases. For this example, you create the Get Open Cases for Account flow.

EDITIONS

Available in: Lightning Experience

Available in: **Unlimited+** Edition

Available for an additional cost in: **Enterprise** and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on.

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

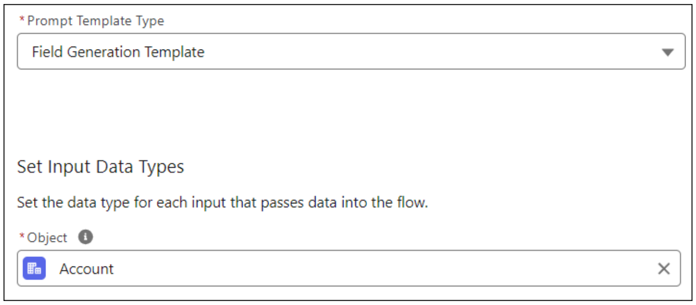
- Manage Flow

To create and manage prompt templates in Prompt Builder:

- Prompt Template Manager permission set

1. Turn on Einstein.
   a. From Setup, in the Quick Find box, enter `Einstein Setup`, and then click **Einstein Setup**.
   b. Turn on Einstein.
2. Create the flow.
   a. From Setup, in the Quick Find box, enter `Flows`, and then click **Flows**.
   b. Click **New Flow**.
   c. From Flow Builder, select **Start from Scratch**, and then click **Next**.
   d. Select **Template-Triggered Prompt Flow**, and then click **Create**.
3. Configure the flow.
   a. Select **Automatic Inputs**.
   b. For Prompt Template Type, select **Field Generation Template**.

      Each prompt template type is associated with its prompt template type in Prompt Builder.
   c. For Object, select **Account**.



*Prompt Template Type

Field Generation Template

Set Input Data Types

Set the data type for each input that passes data into the flow.

*Object

Account

4. Add the Get Records element and configure it.
   a. For Label, enter `Get Open Cases`.
   b. For Object, select **Case**.
   c. For Condition Requirements, select **All Conditions Are Met (AND)**.
   d. For Field, select **AccountId**.

e. For Operator, select **Equals**.

f. For Value, select **$Input** then **RelatedEntity (Account)** then **Id (Account ID)**.

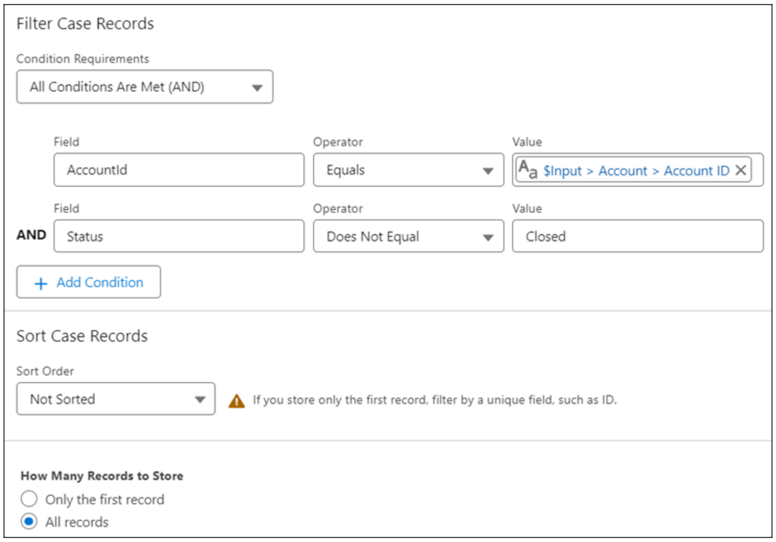
g. Click **Add Condition**.

h. For Field, select **Status**.

i. For Operator, select **Does Not Equal**.

j. For Value, select **Closed**.

k. For How Many Records to Store, select **All Records**.



Filter Case Records

Condition Requirements: All Conditions Are Met (AND)

	Field	Operator	Value
	AccountId	Equals	$Input > Account > Account ID
AND	Status	Does Not Equal	Closed

Sort Case Records

Sort Order: Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

○ Only the first record

● All records

5. Add the Decision element and configure it.

a. For Label, enter `Any Cases?`

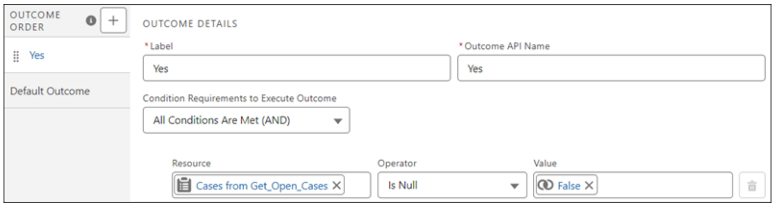
b. On New Outcome, for Label, enter `Yes`.

c. For Condition Requirements to Execute Outcome, select **All Conditions Are Met (AND)**.

d. For Resource, select **Cases from Get_Open_Cases**.

e. For Operator, select **Is Null**.

f. For Value, select **$GlobalConstant.False**.



OUTCOME DETAILS

*Label: Yes

*Outcome API Name: Yes

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

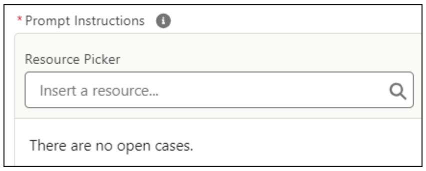
Resource	Operator	Value
Cases from Get_Open_Cases	Is Null	False

g. Click **Default Outcome**.

h. For Label, delete the text, and enter `No`.

6. On the No outcome path, add the Add Prompt Instructions element.

a. For Label, enter: `Add Note About No Open Cases`.
b. For Prompt Instructions, enter: `There are no open cases.`.


* Prompt Instructions

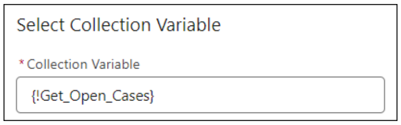
Resource Picker

Insert a resource...

There are no open cases.

If the flow takes this outcome path, the flow stores this text in the $Output global variable. When the flow finishes, the flow passes the text to the field generation prompt template.

7. On the Yes outcome path, add the Loop element.
   a. For Label, enter `Get Each Case`.
   b. For Collection Variable, select **Cases from Get_Open_Cases**.

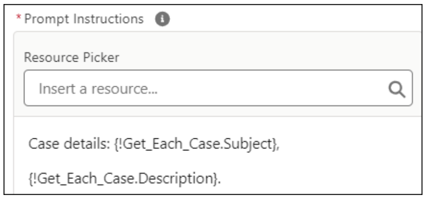

Select Collection Variable

* Collection Variable

{!Get_Open_Cases}

8. On the For Each path, add the Add Prompt Instructions element.
   a. For Label, enter: `Add Case Details to Prompt Template`.
   b. For Prompt Instructions, enter:

   `Case details: {!Get_Each_Case.Subject}, {!Get_Each_Case.Description}.`


* Prompt Instructions

Resource Picker

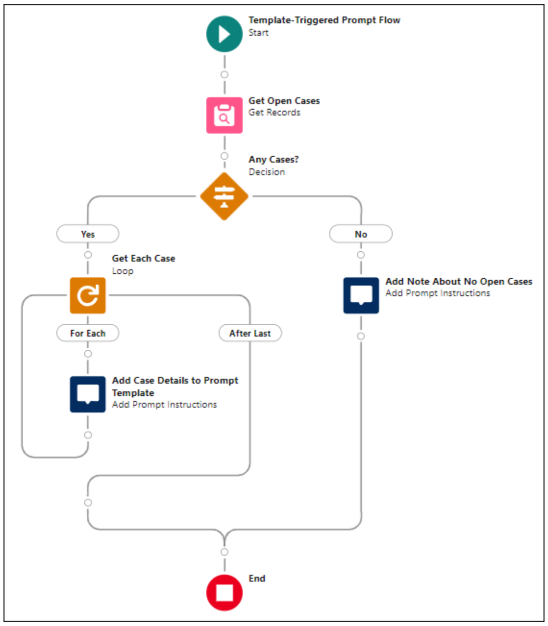
Insert a resource...

Case details: {!Get_Each_Case.Subject},

{!Get_Each_Case.Description}.

If the flow takes this outcome path, the merge fields are replaced with Salesforce data such as the subject and description text for a case. The flow stores the prompt instructions text in the $Output global variable. Each time the flow executes the Add Prompt Instructions element, the text is appended to existing text in the $Output global variable. It can contain details about multiple cases. When the flow finishes, the flow passes the text that's stored in the $Output global variable to the field generation prompt template.

9. Click **Save**.
   a. For Flow Label, enter: `Get Open Cases for Account`.
   b. Click **Save**, and then click **Activate**.



Now that you completed the flow, you create a field generation prompt template in Prompt Builder. You add the flow to the prompt template. When you preview the prompt template in Prompt Builder, it triggers the flow to run, and the flow sends its prompt instructions to the prompt template.

SEE ALSO:

- Flow Element: Add Prompt Instructions
- Flow Resource: Global Variables
- Transform Data in a Flow

Example of Sales Email Template-Triggered Prompt Flow

Let's create an example flow that integrates with a prompt template for generating a list of events.

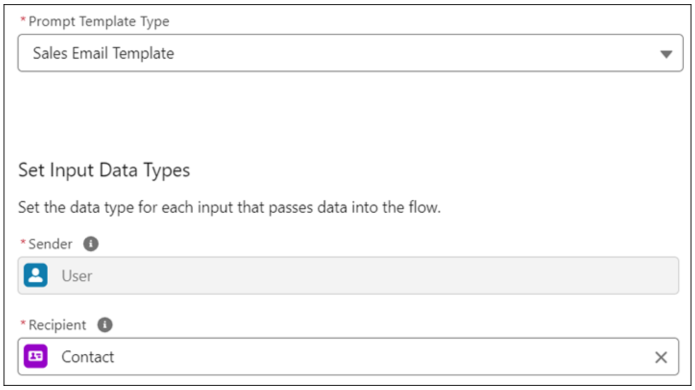
Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

Let's say you're a Salesforce admin, and your sales team requests an easier way to send emails to contacts about nearby Marketing events. You create an email prompt template that launches a prompt template flow to get Marketing events in the same city and state as your contact. And you draft an email for contacts about nearby Marketing events. We use a flow to get a list of events and their locations for the LLM.

Before creating your sales email prompt template, you create a flow that retrieves a list of events. For this example, you create the Get Marketing Events flow.

1. Turn on Einstein.
   a. From Setup, in the Quick Find box, enter `Einstein Setup`, and then click **Einstein Setup**.
   b. Turn on Einstein.
2. Create the flow.
   a. From Setup, in the Quick Find box, enter `Flows`, and then click **Flows**.
   b. Click **New Flow**.
   c. From Flow Builder, select **Start from Scratch**, and then click **Next**.
   d. Select **Template-Triggered Prompt Flow**, and then click **Create**.
3. Configure the flow.
   a. Select **Automatic Inputs**.
   b. For Prompt Template Type, select **Sales Email Template**.

      Each prompt template type is associated with its prompt template type in Prompt Builder.
   c. For Recipient, select **Contact**.



* Prompt Template Type: Sales Email Template

Set Input Data Types

Set the data type for each input that passes data into the flow.

* Sender: User

* Recipient: Contact

4. Add the Get Records element and configure it.
   a. For Label, enter `Get Events`.
   b. For Object, select **Event**.

EDITIONS

Available in: Lightning Experience

Available in: **Unlimited+** Edition

Available for an additional cost in: **Enterprise** and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on.

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and manage prompt templates in Prompt Builder:

- Prompt Template Manager permission set

c. For Condition Requirements, select **All Conditions Are Met (AND)**.

d. For Field, select **IsArchived**.

e. For Operator, select **Equals**.

f. For Value, select **$GlobalConstant.False**.

g. Click **Add Condition**.

h. For Field, select **Location**.

i. For Operator, select **Contains**.

j. For Value, select **$Input** then **Recipient (Contact)** then **MailingCity**.

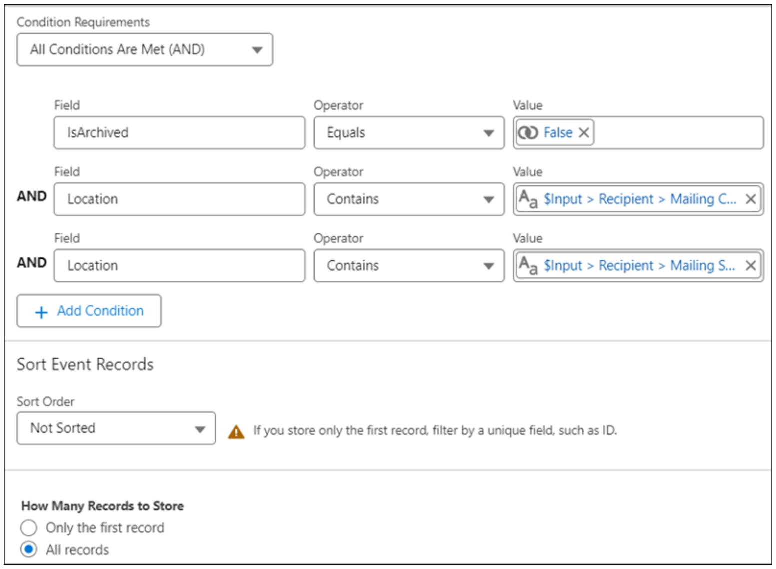
k. Click **Add Condition**.

l. For Field, select **Location**.

m. For Operator, select **Contains**.

n. For Value, select **$Input** then **Recipient (Contact)** then **MailingState**.

o. For How Many Records to Store, select **All Records**.



Condition Requirements

All Conditions Are Met (AND)

	Field	Operator	Value
	IsArchived	Equals	False
AND	Location	Contains	$Input > Recipient > Mailing C...
AND	Location	Contains	$Input > Recipient > Mailing S...

Sort Event Records

Sort Order

Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

○ Only the first record

● All records

5. Add the Decision element and configure it.

a. For Label, enter *`Any Events?`*

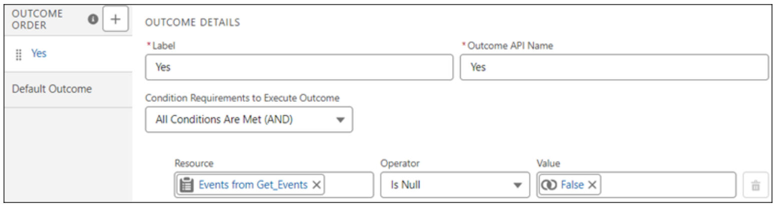
b. On New Outcome, for Label, enter *`Yes`*.

c. For Condition Requirements to Execute Outcome, select **All Conditions Are Met (AND)**.

d. For Resource, select **Events from Get_Events**.

e. For Operator, select **Is Null**.

f. For Value, select **$GlobalConstant.False**.



OUTCOME ORDER

Yes

Default Outcome

OUTCOME DETAILS

*Label: Yes

*Outcome API Name: Yes

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource: Events from Get_Events

Operator: Is Null

Value: False

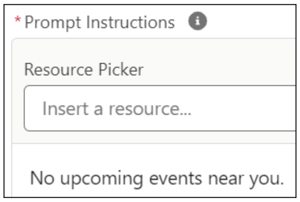
g. Click **Default Outcome**.

h. For Label, delete the text, and enter `No`.

6. On the No outcome path, add the Add Prompt Instructions element.

a. For Label, enter: `Add Note About No Events`.

b. For Prompt Instructions, enter: `No upcoming events near you.`.



*Prompt Instructions

Resource Picker

Insert a resource...

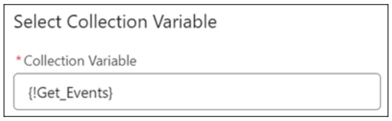
No upcoming events near you.

If the flow takes this outcome path, the flow stores this text in the $Output global variable. When the flow finishes, the flow passes the text to the sales email prompt template.

7. On the Yes outcome path, add the Loop element.

a. For Label, enter `Add Events`.

b. For Collection Variable, select **Events from Get_Events**.



Select Collection Variable

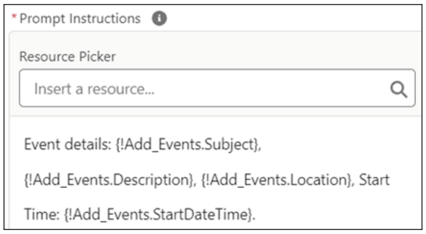
*Collection Variable: {!Get_Events}

8. On the For Each path, add the Add Prompt Instructions element.

a. For Label, enter: `Add Event Details`.

b. For Prompt Instructions, enter:

```
Event details: {!Add_Events.Subject}, {!Add_Events.Description},
{!Add_Events.Location}, Start Time: {!Add_Events.StartDateTime}.
```



*Prompt Instructions

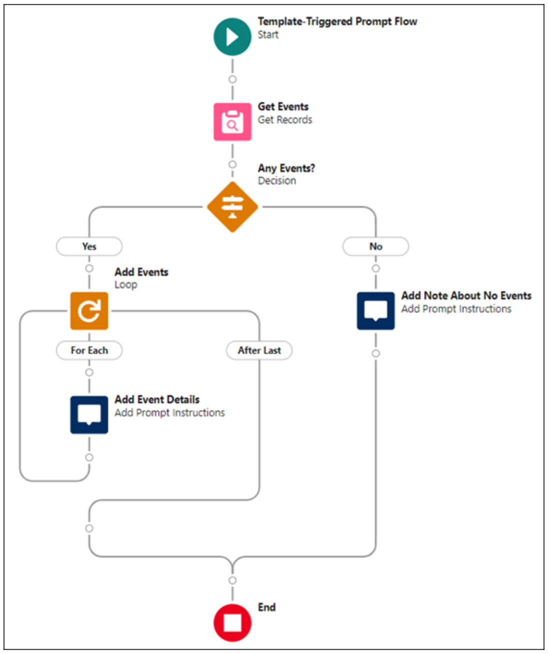
Resource Picker

Insert a resource...

Event details: {!Add_Events.Subject}, {!Add_Events.Description}, {!Add_Events.Location}, Start Time: {!Add_Events.StartDateTime}.

If the flow takes this outcome path, the merge fields are replaced with Salesforce data such as the subject, description, location, and start time text for an event. The flow stores the prompt instructions text in the $Output global variable. Each time the flow executes the Add Prompt Instructions element, the text is appended to existing text in the $Output global variable. It can contain details about multiple cases. When the flow finishes, the flow passes the text that's stored in the $Output global variable to the sales email prompt template.

9. Click **Save**.
   a. For Flow Label, enter: `Get Marketing Events`.
   b. Click **Save**, and then click **Activate**.



Now that you completed the flow, you create a sales email prompt template in Prompt Builder. You add the flow to the prompt template. When you preview the prompt template in Prompt Builder, it triggers the flow to run, and the flow sends its prompt instructions to the prompt template.

SEE ALSO:

Flow Element: Add Prompt Instructions

Flow Resource: Global Variables

Transform Data in a Flow

Example of Reusable Template-Triggered Prompt Flow

Let’s create an example flow that’s available to all prompt templates.

Let’s say you’re a Salesforce admin, and your teams request an easier way to get new cases for an account. You create a prompt template that launches a template-triggered prompt flow to get the case details for the prompt template.

Before creating your prompt template, you create a flow that retrieves the new cases. For this example, you create the Get New Cases for Account flow.

EDITIONS

Available in: Lightning Experience

Available for an additional cost in: **Enterprise** and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on.

USER PERMISSIONS

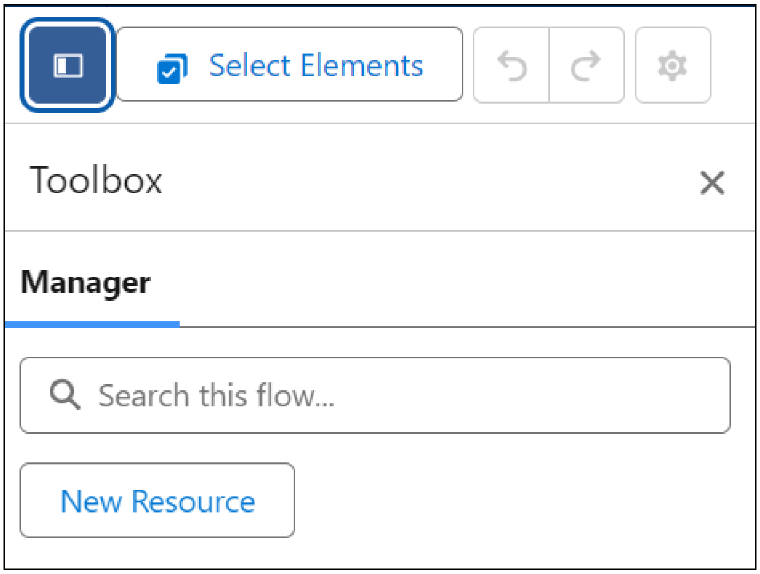
To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and manage prompt templates in Prompt Builder:

- Prompt Template Manager permission set

1. Turn on Einstein.
   a. From Setup, in the Quick Find box, enter `Einstein Setup`, and then click **Einstein Setup**.
   b. Turn on Einstein.
2. Create the flow.
   a. From Setup, in the Quick Find box, enter `Flows`, and then click **Flows**.
   b. Click **New Flow**.
   c. From Flow Builder, select **Start from Scratch**, and then click **Next**.
   d. Select **Template-Triggered Prompt Flow**, and then click **Create**.
3. Configure the flow.
   a. Select **Manual Inputs**.
      Each prompt template type is associated with its prompt template type in Prompt Builder.
   b. In Toolbox Manager, click **New Resource**.



   c. For Resource Type, select **Variable**.
   d. Set the variable API name and description.
      The API name must match the prompte template’s API name if the inputs are Text data types or the flow includes more than one input that matches with the prompt template input . For this example, use API name: Account

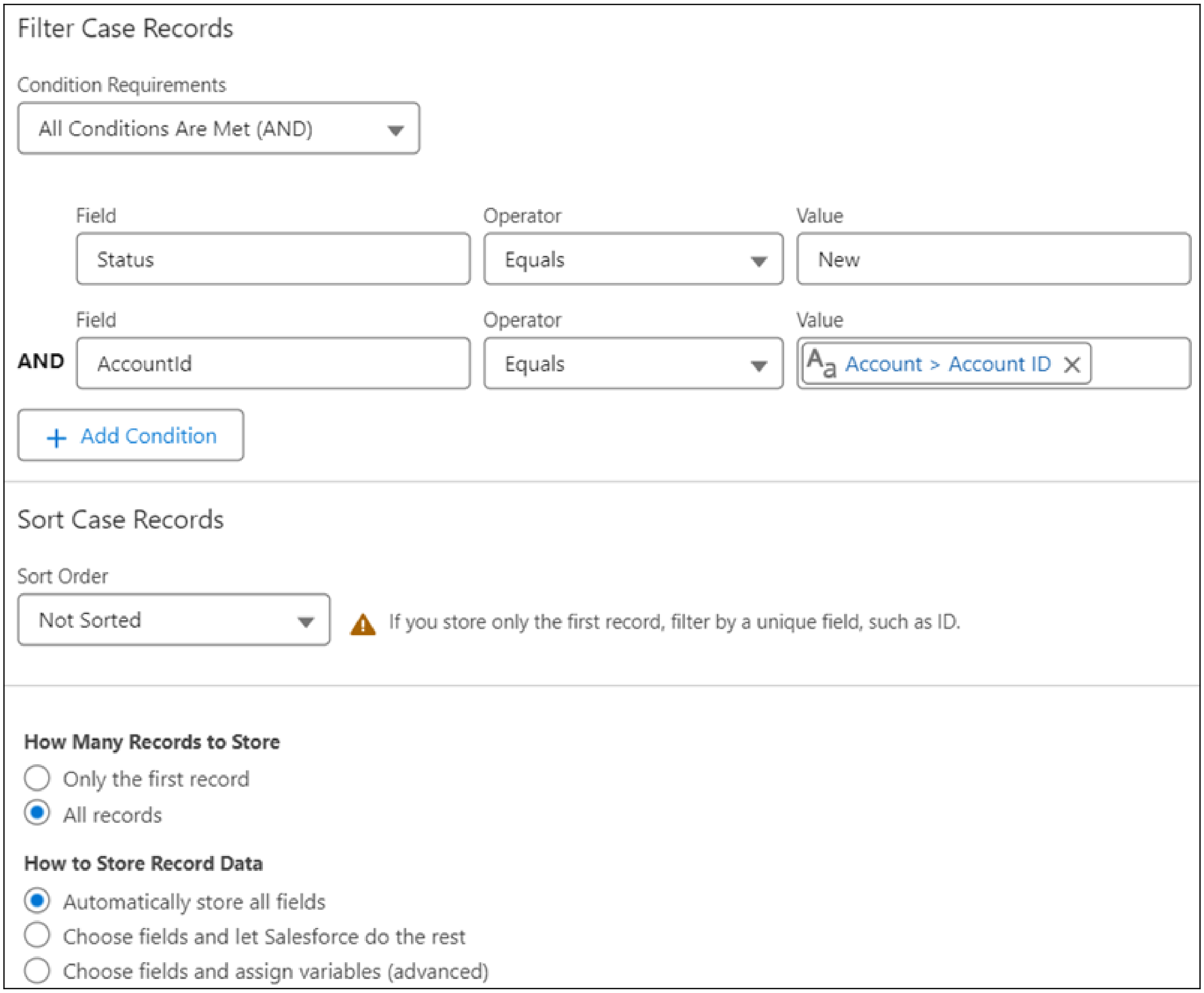
   e. For Data Type, select **Record**.
   f. For Object, select **Account**.

      This data type corresponds to the Account input in a prompt template that references this flow.

   g. Select **Available for input**.

      This option lets Prompt Builder set the input.

4. Add the Get Records element and configure it.
   a. For Label, enter `Get New Cases`.
   b. For Object, select **Case**.
   c. For Condition Requirements, select **All Conditions Are Met (AND)**.
   d. For Field, select **Status**.
   e. For Operator, select **Equals**.
   f. For Value, select **New**.
   g. Click **Add Condition**.
   h. For Field, select **AccountId**.
   i. For Operator, select **Equals**.
   j. For Value, select **{!Account.Id}**.
   k. For How Many Records to Store, select **All Records**.



Filter Case Records

Condition Requirements

All Conditions Are Met (AND)

	Field	Operator	Value
	Status	Equals	New
AND	AccountId	Equals	Account > Account ID

Add Condition

Sort Case Records

Sort Order

Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

○ Only the first record

● All records

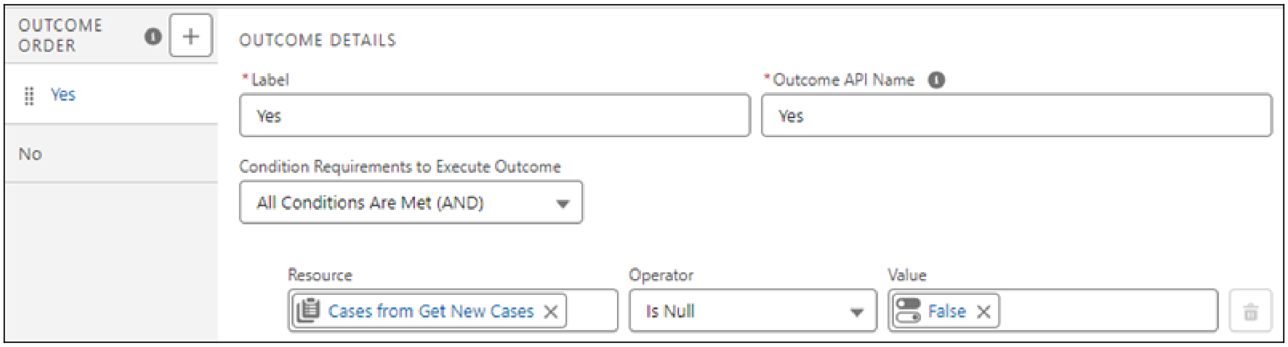
How to Store Record Data

● Automatically store all fields

○ Choose fields and let Salesforce do the rest

○ Choose fields and assign variables (advanced)

5. Add the Decision element and configure it.
   - a. For Label, enter `Any New Cases?`
   - b. On New Outcome, for Label, enter `Yes`.
   - c. For Condition Requirements to Execute Outcome, select **All Conditions Are Met (AND)**.
   - d. For Resource, select **Cases from Get_New_Cases**.
   - e. For Operator, select **Is Null**.
   - f. For Value, select **$GlobalConstant.False**.



*Label: Yes

*Outcome API Name: Yes

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource: Cases from Get New Cases

Operator: Is Null

Value: False

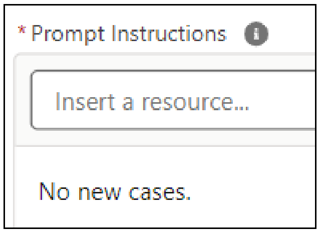
g. Click **Default Outcome**.

h. For Label, delete the text, and enter `No`.

6. On the No outcome path, add the Add Prompt Instructions element.

   a. For Label, enter: `Add Note About No New Cases`.

   b. For Prompt Instructions, enter: `No new cases.`.



* Prompt Instructions

Insert a resource...

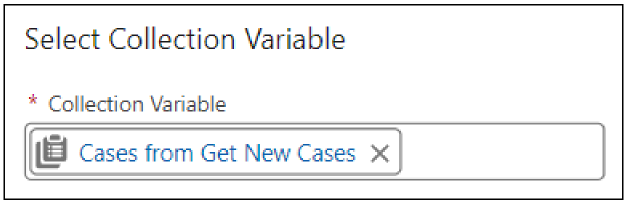
No new cases.

If the flow takes this outcome path, the flow stores this text in the $Output global variable. When the flow finishes, the flow passes the text to the field generation prompt template.

7. On the Yes outcome path, add the Loop element.

   a. For Label, enter `Add Cases`.

   b. For Collection Variable, select **Cases from Get_New_Cases**.



Select Collection Variable

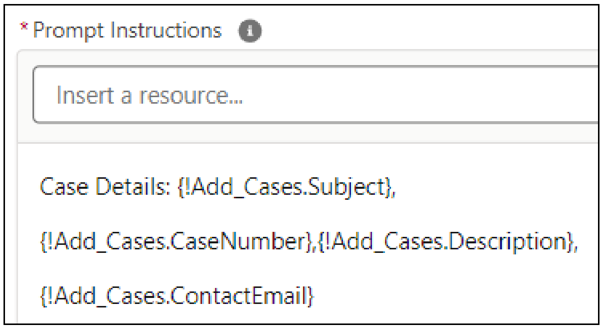
* Collection Variable: Cases from Get New Cases

8. On the For Each path, add the Add Prompt Instructions element.

   a. For Label, enter: `Add Case Details`.

   b. For Prompt Instructions, enter:

```
Case Details: {!Add_Cases.Subject},
{!Add_Cases.CaseNumber},{!Add_Cases.Description},{!Add_Cases.ContactEmail}.
```



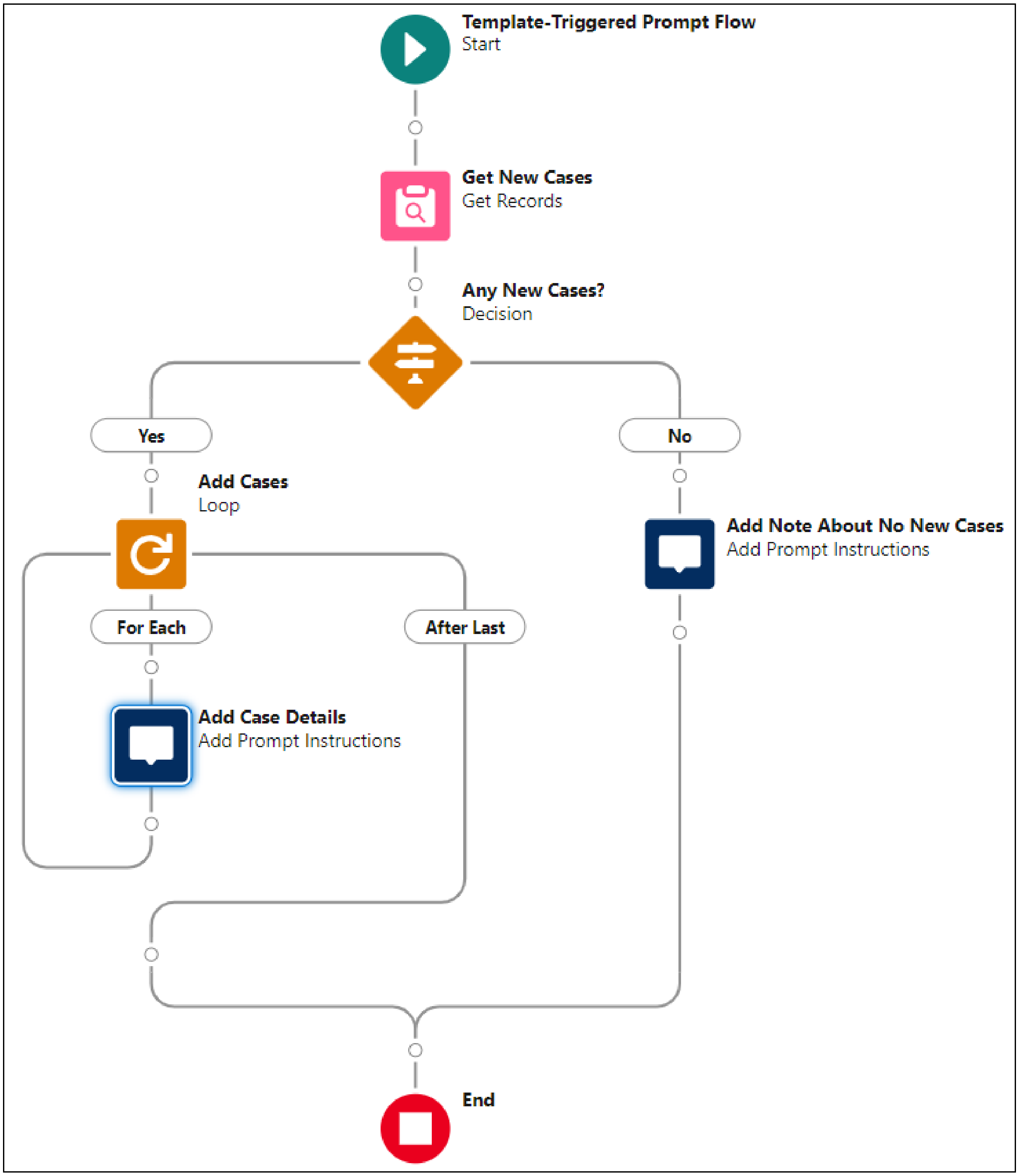
Case Details: {!Add_Cases.Subject},

{!Add_Cases.CaseNumber},{!Add_Cases.Description},

{!Add_Cases.ContactEmail}

If the flow takes this outcome path, the merge fields are replaced with Salesforce data such as the subject, description, case number, email for a case. The flow stores the prompt instructions text in the $Output global variable. Each time the flow executes the Add Prompt Instructions element, the text is appended to existing text in the $Output global variable. It can contain details about multiple cases. When the flow finishes, the flow passes the text that's stored in the $Output global variable to the field generation prompt template.

9. Click **Save**.
   a. For Flow Label, enter: `Get New Cases for Account`.
   b. Click **Save**, and then click **Activate**.



Now that you completed the flow, you create a prompt template in Prompt Builder that includes an Account input. You add the flow to the prompt template. When you preview the prompt template in Prompt Builder, it triggers the flow to run, and the flow sends its prompt instructions to the prompt template.

SEE ALSO:

- Flow Element: Add Prompt Instructions
- Flow Resource: Global Variables
- Transform Data in a Flow

Advanced Flow Concepts

After you understand the basics, you're ready for a closer look at what you can do after a flow finishes. Learn what happens when a flow interacts with the Salesforce database, and how flows perform similar operations at the same time. Get to know how a flow test saves you time and improves reliability.

API Version for Running a Flow

Some run-time behavior improvements are delivered as versioned updates, so that you can control when each flow adopts those updates. Test and upgrade your flows one by one and at your own pace. You can even opt to never adopt versioned updates for one or all your flows.

Flow Environments

A flow environment specifies where a flow can run. For example, if a flow is associated with the Slack flow environment, you can enable users to launch the flow from Slack.

What Happens When a Flow Finishes?

By default, when a flow interview that uses screens finishes, a new interview for that flow begins, and the user is redirected to the first screen. To override the default behavior, you can add a local action to the flow. Some distribution methods also offer other ways to override a flow's finish behavior, such as by setting the `retURL` parameter in a flow URL.

Flows in Transactions

Each flow interview runs in the context of a *transaction*. A transaction represents a set of operations that are executed as a single unit. For example, a transaction can execute Apex triggers and escalation rules in addition to a flow interview. If one interview in a transaction fails, all the interviews in the transaction are rolled back, as well as anything else the transaction did. The transaction doesn't retry any of the operations—including the flow interview.

Flow Bulkification in Transactions

Programmers can design their code so that similar actions are performed together in one batch. For example, one operation to create 50 records rather than 50 separate operations that each create one record. This process is called *bulkification*, and it helps your transaction avoid governor limits. If you're working with flows, you don't even have to think about bulkification. Flow interviews bulkify actions for you automatically.

Apex-Defined Data Type

With the Apex-defined data type, flows can manipulate the kinds of complex data objects that are typically returned from calls to web services. Create Apex-defined variables in flows and directly process JSON returned from web calls.

API Version for Running a Flow

Some run-time behavior improvements are delivered as versioned updates, so that you can control when each flow adopts those updates. Test and upgrade your flows one by one and at your own pace. You can even opt to never adopt versioned updates for one or all your flows.

By default, when you create a flow, it's configured to run in the latest API version. That API version doesn't change as future Salesforce releases roll out. But you can change the API version for running each flow. Starting in Winter '24, flows created with API version 49.0 and earlier display an API version of 0 on the Flows list view in Setup. To display the correct API version number, create another version of the flow, and set the API version for running the flow to 49.0 or later.

Before you select a new API version, review all run-time improvements that were delivered between the currently selected API version and the new API version. You can find all flow and process run-time improvements for an API version in the Salesforce Release Notes. The API version for running a flow is specified in the flow version properties.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Sometimes, a release update is also available as a versioned update. Such a versioned update lets you adopt the changes for individual flows or processes before the release update is enforced. After the release update is adopted or enforced, all flows and processes in your org get the updated behavior regardless of their run-time API versions.

SEE ALSO:

Flow Version Properties

Flow Environments

A flow environment specifies where a flow can run. For example, if a flow is associated with the Slack flow environment, you can enable users to launch the flow from Slack.

Flows can have these environments.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Default

The flow can run from a Visualforce component, Lightning page, flow action, or custom Aura component.

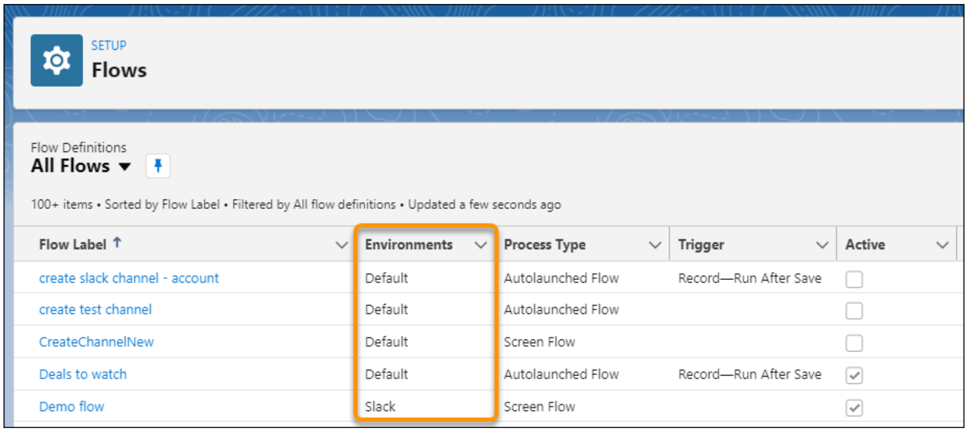
Offline

The flow can run only offline.

Slack

The flow can run in Slack and the default environment. You specify the Slack flow environment when you save the flow. Then, you use the Slack Send Message to Launch Flow action in a different flow to enable users to run the flow from Slack.

You can determine the current environment of a flow from the Flows page of Setup and from the detailed view of a flow. For example:



SETUP

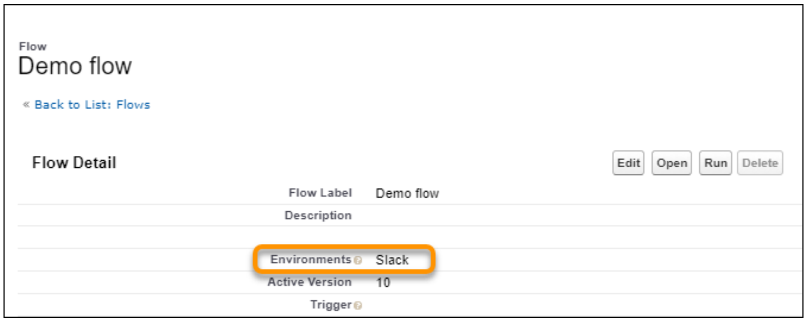
Flows

Flow Definitions

All Flows

100+ items • Sorted by Flow Label • Filtered by All flow definitions • Updated a few seconds ago

Flow Label ↑	Environments	Process Type	Trigger	Active
create slack channel - account	Default	Autolaunched Flow	Record—Run After Save	☐
create test channel	Default	Autolaunched Flow		☐
CreateChannelNew	Default	Screen Flow		☐
Deals to watch	Default	Autolaunched Flow	Record—Run After Save	☑
Demo flow	Slack	Screen Flow		☑



Flow

Demo flow

Flow Detail

Flow Label Demo flow

Description

Environments Slack

Active Version 10

Trigger

Flows in Slack

Running a flow in Slack requires two things. You must have an active screen flow with the Make available in Slack setting selected. You also need a way to run the flow from Slack.

Flows in Slack

Running a flow in Slack requires two things. You must have an active screen flow with the Make available in Slack setting selected. You also need a way to run the flow from Slack.

Screen Components in Flows in Slack

Screens in a screen flow with the Make available in Slack advanced setting selected can only contain components supported by Flows in Slack.

Flows in Slack don't support field-level validation for screen components.

Supported Components	Not Supported Components
• Checkbox • CheckboxGroup (up to 10 choices) • Currency • Date • Date & Time • Display Text • Long Text Area • Number • Picklist (up to 100 choices) • Radio Button (up to 10 choices) • Text	• Address • Dependent Picklist • Display Image • Email • File Upload • Lookup • Multi-Select Picklist • Name • Password • Phone • Section • Slider • Toggle • URL

Input Variables in Flows in Slack

A screen flow with the Make available in Slack advanced setting selected can have text input parameters. Flows designed to run in Slack support only text variables as input parameters, and they can be run from Slack using only the Send Message to Launch Flow action. To get text input from flow users, define text variables that are available for input in your screen flow.

Use functions to convert text input into the format that you need within your screen flow.

Table 1: Common Text Conversion Functions

Function	Category	Description
DATEVALUE	Date and Time Functions	Takes a string with a format of "YYYY-MM-DD" and returns a date value.
DATETIMEVALUE	Date and Time Functions	Takes a string with a format of "YYYY-MM-DD HH:MM:SS" and returns a year, month, day, and GMT time value.
ISNUMBER	Logical Functions	Determines if a text value is a number and returns TRUE if it is. Otherwise, it returns FALSE.
VALUE	Text Functions	Converts a text string to a number.

Send an Active Screen Flow with or Without Input Variables to Slack with an Action

This method of sending an active screen flow to Slack works only with official Salesforce Slack apps. This method is the only way to run active screen flows with input variables and the Make available in Slack advanced setting selected

An active screen flow with the Make available in Slack advanced setting selected can run in the flow default environment. To send the flow to run in the Slack environment, use another flow that calls a version of the Slack Send Message to Launch Flow action.

The Slack Send Message to Launch Flow action has a version for each active screen flow with the Make available in Slack advanced setting selected. For example, your active screen flow is Get Pet Name. When you add the Action element to the new flow that sends this active screen flow to Slack, select the action labeled Get Pet Name with the API name of slackSendMessageToLaunchFlow - Get_Pet_Name.

When the flow containing the action runs, it sends a message to a Slack channel or direct message group. The message includes a button that a channel or group member can click to launch the associated screen flow.

These flow types don't support the Slack Send Message to Launch Flow action: CMS Orchestrator, EvaluationFlow, Journey, Orchestrator, Survey, SurveyEnrich, and TransactionSecurityFlow.

Run an Active Screen Flow Without Input Variables from a Button in a Slack View

This method of invoking an active screen flow from Slack works for active screen flows with no input variables and the Make available in Slack advanced setting selected. This method works with official and custom Slack apps.

Use the Apex SDK for Slack (Beta) to create a view that invokes an active screen flow from a button in Slack.

Run an Active Screen Flow Without Input Variables from a Slack Shortcut

This method of invoking an active screen flow from Slack works for active screen flows with no input variables. Create a custom Slack app to use this method. Custom Slack apps don't have access to Slack Flow Core Actions.

Use the Apex SDK for Slack (Beta) to define a custom Slack app that invokes a flow from a global shortcut, a message shortcut, or a slash command in Slack.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Use Flows with Slack

Run Active Screen Flows from Slack

Apex SDK for Slack (Beta): Define an App

Apex SDK for Slack (Beta): Define an View

What Happens When a Flow Finishes?

By default, when a flow interview that uses screens finishes, a new interview for that flow begins, and the user is redirected to the first screen. To override the default behavior, you can add a local action to the flow. Some distribution methods also offer other ways to override a flow's finish behavior, such as by setting the `retURL` parameter in a flow URL.

Note: For smooth finish behavior, don't create records before the first screen in the flow. You don't want any CRUD operations running unintentionally.

Distribution Method	Default Finish Behavior	Override Options
URL (direct URL, web tab, custom button, custom link)	Starts new interview	• Add a local action to the flow • Set the `retURL` parameter
Lightning page	Starts new interview	Add a local action to the flow
Experience Builder page	Starts new interview	Add a local action to the flow
Flow Quick Action (Does not apply to LWC quick actions that have flows embedded.)	Closes dialog	Add a local action to the flow
Utility bar	Starts new interview	Add a local action to the flow
lightning:flow Aura component	Starts new interview	• Add a local action to the flow • Set the `onstatuschange` action
lightning-flow LWC component	Starts new interview	• Define whether the flow restarts

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Distribution Method	Default Finish Behavior	Override Options
		using `flowfinishbehavior` - Add a local action to the flow - Set the `onstatuschange` action
flow:interview Visualforce component	Starts new interview	- Add a local action to the flow - Set the `finishLocation` attribute

Redirect Flow Users with a Local Action

By default, when a flow finishes, a new interview starts and the user sees the first screen of the flow. To instead redirect the user to another page, build or install a local action that does so. Then add the action to your flow with a Core Action element. For example, a local action can open a record, list view, or URL or to show a toast message. Or it can use the Lightning Console JavaScript API to close a console tab.

SEE ALSO:

Redirect Flow Users with a Local Action

Customize a Flow URL to Control Finish Behavior

Customize a Visualforce Component to Control the Flow's Finish Behavior

Redirect Flow Users with a Local Action

EDITIONS

Available in: Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

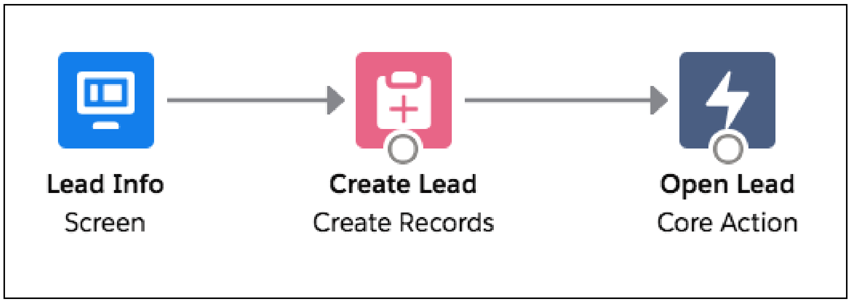
By default, when a flow finishes, a new interview starts and the user sees the first screen of the flow. To instead redirect the user to another page, build or install a local action that does so. Then add the action to your flow with a Core Action element. For example, a local action can open a record, list view, or URL or to show a toast message. Or it can use the Lightning Console JavaScript API to close a console tab.

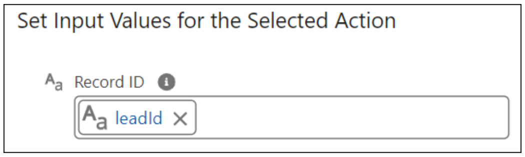
Note: Local actions that fire `force` or `lightning` events often don't work properly when you run the flow from:

- Flow Builder
- Flow detail pages or list views
- Web tabs
- Custom buttons and links

Instead, test and distribute the flow with a Lightning page, Experience Builder page, flow action, or utility bar. Your developer can also add the appropriate event handlers directly to the component.

Example: A flow creates a lead using information entered in the Lead Info screen. The Lead Info screen stores the value of the lead ID in the leadId variable that was manually created. Then it executes the Open Lead local action, which passes the lead ID into an Aura component by using the Record ID attribute. The component uses a Lightning event to open the created lead.




Set Input Values for the Selected Action

Record ID

leadId

Let's look at the Aura component that the local action calls: `c:navigateToRecord`.

Component Markup

```
<aura:component implements="force:lightningQuickAction,
lightning:availableForFlowActions">
   <aura:attribute name="recordId" type="String" />
</aura:component>
```

Design Resource

The `recordId` attribute is declared in the design resource so that it's configurable in the local action.

```
<design:component>
   <design:attribute name="recordId" label="Record ID" />
</design:component>
```

Client-Side Controller

When the local action is executed, the flow calls the `invoke` method, which uses the `force:navigateToSObject` event to navigate to the created record.

```
({    invoke : function(component, event, helper) {
   // Get the record ID attribute
   var record = component.get("v.recordId");

   // Get the Lightning event that opens a record in a new tab
   var redirect = $A.get("e.force:navigateToSObject");

   // Pass the record ID to the event
   redirect.setParams({
      "recordId": record
   });

   // Open the record
   redirect.fire();
}})
```

SEE ALSO:

What Happens When a Flow Finishes?

Lightning Aura Components Developer Guide: Runtime Considerations for Flows That Include Aura Components

Flows in Transactions

Each flow interview runs in the context of a *transaction*. A transaction represents a set of operations that are executed as a single unit. For example, a transaction can execute Apex triggers and escalation rules in addition to a flow interview. If one interview in a transaction fails, all the interviews in the transaction are rolled back, as well as anything else the transaction did. The transaction doesn't retry any of the operations—including the flow interview.

In each transaction, Salesforce enforces governor limits to prevent shared resources from being depleted. Because multiple Salesforce organizations share resources, Salesforce prevents one organization from depleting all the resources and leaving the other organizations high and dry. It's similar to an apartment building that uses one cache of water to service every tenant. If your neighbor uses all the water, you can't take a shower. (It's trite, but hopefully you get the idea.) Per-transaction governor limits help prevent such things from happening.

When Does a Flow's Transaction Start?

Depending on how the flow was distributed, a transaction that runs an interview for that flow starts in different ways.

When Does a Flow's Transaction End?

When a transaction ends depends on whether the flow contains certain elements and whether it originally started because a record was changed.

SEE ALSO:

Flow Bulkification in Transactions

Per-Transaction Flow Limits

Process Limits

When Does a Flow's Transaction Start?

Depending on how the flow was distributed, a transaction that runs an interview for that flow starts in different ways.

[1] The same also applies if the flow is distributed through a workflow rule. The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Distribution Method	Transaction starts when...
Process Builder[1]	A record is created or updated.
Flow URL	The URL is accessed.
Custom button or link	The button or link is clicked.
Visualforce page	The page is accessed.

Distribution Method	Transaction starts when...
`Interview.start()` method	If the method starts via a `before` or `after` trigger, the transaction starts when a record is created or updated. Otherwise, the transaction starts when the method (or a parent method) is invoked. The `start()` method shares its limits with other operations in the transaction and other methods in the class.
REST API (Custom Actions or Flows resource)	When the REST call is made. Depending on how the REST call is implemented, the limits can be shared with other operations.

Note: When a Screen element, local action, or Wait element is executed, the existing transaction ends and a new one begins.

When Does a Flow's Transaction End?

When a transaction ends depends on whether the flow contains certain elements and whether it originally started because a record was changed.

The transaction ends when:

- A Screen, Local Action, or Wait element is executed
- The order of execution has completed—if the flow was triggered when a record was created or updated
- All the interviews in the transaction have finished

Tip: If you think that a flow's interview is likely to hit governor limits within its transaction, consider adding a Screen, Local Action, or Wait element.

If the interview is one of many things being done in a given transaction, that interview shares the transaction's governor limits with the other operations.

Example: You update 100 cases through Data Loader. Due to the order of execution in a transaction and the customizations in your organization, here's what happens.

	Transaction Operation	DML Statement Used	SOQL Query Used
1	Cases are saved to the database, but aren't committed yet.		
2	Case assignment rules are executed. Each case's owner is updated.	✓	
3	Case escalation rules are executed. If any case has been open for 10 days, an email is sent to the owner.		
4	Process is started.		
5	Process looks up the case's account.		✓
6	If the account is hot, process uses Chatter to notify the account owner that there's a new case associated with the account.	✓	
7	Process launches a flow interview.		

	Transaction Operation	DML Statement Used	SOQL Query Used
8	Flow interview looks up the parent account and how many cases it has.		✓
9	Flow interview checks whether the account has more than five open cases.		
10	If it does, flow interview looks up the account's division manager then posts on the account's Chatter feed to notify the division manager and account owner.	✓	✓
11	If it doesn't, flow interview posts on the account's Chatter feed to notify only the account owner.	✓	

SEE ALSO:

Apex Developer Guide : Triggers and Order of Execution

Flow Bulkification in Transactions

Programmers can design their code so that similar actions are performed together in one batch. For example, one operation to create 50 records rather than 50 separate operations that each create one record. This process is called *bulkification*, and it helps your transaction avoid governor limits. If you're working with flows, you don't even have to think about bulkification. Flow interviews bulkify actions for you automatically.

How Does Flow Bulkification Work?

Interview operations are bulkified only when they execute the same element. That means that the interviews must all be associated with the same flow.

Which Flow Elements Can Be Bulkified?

Flows can bulkify any element that performs a DML statement or SOQL query or does something else external to the flow, like sending an email.

Example of Flow Bulkification

This example demonstrates how operations are bulkified for a flow when 100 cases are updated through Data Loader.

SEE ALSO:

Flows in Transactions

How Does Flow Bulkification Work?

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Interview operations are bulkified only when they execute the same element. That means that the interviews must all be associated with the same flow.

When multiple interviews for the same flow run in one transaction, each interview runs until it reaches a bulkifiable element. Salesforce takes all the interviews that stopped at the same element and intelligently executes those operations together. If other interviews are at a different element, Salesforce then intelligently executes those operations together. Salesforce repeats this process until all the interviews finish.

If, despite the bulkification, any interview hits a governor limit, all the interviews in the transaction fail. Any operations that the interviews performed in the transaction are rolled back, and the transaction doesn't try to perform the operations again. Any operations that access external data aren't rolled back.

If an error that isn't due to a governor limit occurs while executing one of these elements, Salesforce attempts to save all successful record changes in the bulk operation up to three times.

- Subflow (Create Records and Update Records elements only)
- Create Records
- Update Records

Example: When you upload 100 cases, the flow MyFlow_2 triggers one interview for each case.

- 50 interviews stop at Create Records element Create_Task_1.
- The other 50 interviews stop at Create Records element Create_Task_2.

The result? At least two groups of bulk operations to execute.

- One for the 50 interviews that execute Create_Task_1
- One for the 50 interviews that execute Create_Task_2

Which Flow Elements Can Be Bulkified?

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flows can bulkify any element that performs a DML statement or SOQL query or does something else external to the flow, like sending an email.

Elements that create, update, or delete records

When a record is created, updated, or deleted, the transaction performs a DML statement.

- Create Records elements
- Update Records elements
- Delete Records elements
- Quick Action, Post to Chatter, or Submit for Approval core actions
- Apex Action elements—depending on the specific Apex action

Elements that look up records

When fields on a record are looked up, the transaction performs a SOQL query.

- Get Records elements
- Update Records elements

- Delete Records elements
- Apex Action elements—depending on the specific Apex action

Elements that send emails

- Send Email core actions
- Email Alert elements
- Apex Action elements—depending on the specific Apex action

SEE ALSO:

Apex Developer Guide : Running Apex with Governor Execution Limits

Example of Flow Bulkification

This example demonstrates how operations are bulkified for a flow when 100 cases are updated through Data Loader.

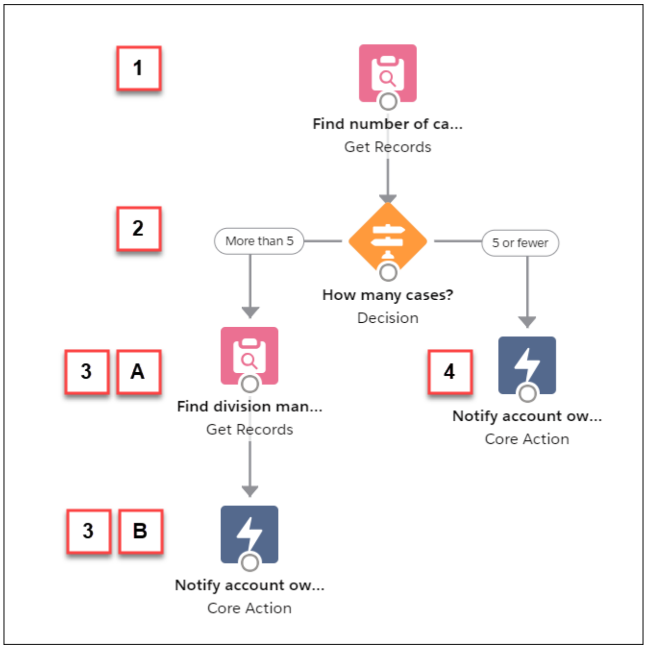
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

The Associated Flow

To better understand, here's the design of the associated flow.



The flow:

- Looks up the case's parent account and how many open cases that account has.

- Checks whether the account has more than five cases open.
- If the account has more than five open cases:
  - Looks up the division manager for the account.
  - Posts on the account's Chatter feed to notify the division manager and the account owner.
- If the account has five or fewer open cases, posts on the account's Chatter feed to notify only the account owner.

The Bulkified Interviews

When you update the records, one flow interview is created for each case simultaneously. All of the interviews are associated with the same flow. Each interview runs until it gets to a bulkifiable element.

The first interview goes through the Get Records element (1). Because Get Records can be bulkified, the interview waits there until all the other interviews have done the same. Then, Salesforce executes all the Get Records operations together (because they're all for the same element in the same flow). Instead of 100 SOQL queries, the transaction issues one SOQL query.

The Decision element (2) evaluates the first interview. The account has six cases, so the interview is routed down the "More than 5" path. The interview proceeds to the second Get Records element (3a). Because it's a bulkifiable element, the interview waits there.

The Decision element (2) evaluates the second interview. This account has one case, so the interview is routed down the "5 or fewer" path. The interview proceeds to the Post to Chatter core action (4). This element is also bulkifiable, so the interview waits there.

After all the interviews have been processed, 30 are waiting to execute the second Get Records element (3a) and the remaining 70 are waiting to execute the Post to Chatter core action (4).

Salesforce executes all the Get Records (3a) operations for the first 30 interviews together. Instead of 30 separate SOQL queries, the transaction issues one.

Next, the transaction returns to the Post to Chatter core action (4), where the 70 interviews are ready to execute their Post to Chatter operations. Remember, the accounts in these interviews don't have more than five cases. Salesforce executes the Post to Chatter operations together. Instead of 100 separate DML statements to create each Chatter post, the transaction issues one DML statement to create all 100 posts at one time. Because the Post to Chatter core action isn't connected to a subsequent element, those 70 interviews finish.

The 30 interviews—which looked up the relevant division manager—proceed to the final Post to Chatter core action (3b). When all 30 interviews are ready, Salesforce executes all 30 Post to Chatter operations together. Instead of issuing 30 separate DML statements for the individual Chatter posts, it issues one. Because the Post to Chatter core action isn't connected to another element, those 30 interviews finish.

Apex-Defined Data Type

With the Apex-defined data type, flows can manipulate the kinds of complex data objects that are typically returned from calls to web services. Create Apex-defined variables in flows and directly process JSON returned from web calls.

A developer can define an Apex class to serve as a pattern for automatic conversion from web to a flow, allowing full manipulation of the resulting objects to be carried out in flows using declarative approaches and no additional code. Apex-defined resources are useful for connecting flows to rich external web objects accessed via Mulesoft and REST calls. If a data type is not supported, flows can pass the value to a Lightning component and you can use Apex to operate on it.

Example: A car dealership has a screen flow that lets customers search the dealership's inventory data, which is stored in another database. The dealership creates an Apex class in their org to define the Car data type. Then, the dealership creates a screen flow that includes

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

two Apex-defined variables. The flow passes the variables between the flow, an Apex action, and a Lightning component.

SEE ALSO:

Considerations for the Apex-Defined Data Type

Build a Flow

After you model the process that you want to automate, design and build the flow in Flow Builder.

Tip: Before you start creating your flow, plan it out. It's easier to use a flow to automate a business process when you understand all the details.

If you're new to Flow Builder or process automation, take a quick tour of the interface, and then check out our trails! They're a great way to learn about these powerful tools and discover how they work. You can also have Einstein generative AI help you get started with a flow by using Einstein for Flow.

- Tour the Flow Builder User Interface
- Automate Your Business Processes with Flow Builder
- Build Flows with Flow Builder
- Let Einstein Help You Build Flows (Beta)

1. Open Flow Builder. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. Select **Start From Scratch**, then click **Next**.
3. Select a flow type, then click **Create**.
4. Add the elements that you want to use to the canvas.
5. If building in free-form, connect the elements to determine their order of execution. If building in auto-layout, connectors are automatically created on the canvas for you.
6. Save your flow.

After you build a flow, make sure that it's working as you expect it to by testing it. Then activate the flow. You're now ready to distribute the flow to users.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Let Einstein Help You Build Flows (Beta)

Save time by describing what you want to automate and letting Einstein generative AI build you a draft flow to get you started with your automation.

Add and Edit Elements

Use elements to define actions that the flow can execute. When they're on the canvas, connect them to create an order of execution.

Define Conditions in a Flow

Control when a flow takes a specific decision outcome or waits for a specific resume event.

Creating Flow Formulas

Flow formulas help make your flows more dynamic by performing calculations, manipulating data, and making decisions based on field or variable values. Formulas can facilitate conditional logic, taking the flow down different paths based on criteria and enforcing validation rules to ensure data integrity.

Save a Draft of Your Flow as You Build

You can interact with your flow while an element is open and access the toolbar while editing that element. And you don’t need to finish configuring an element before saving your flow.

Move and Connect Elements to Change a Flow Route

Identify which elements the flow executes and in what order by connecting the elements on your canvas together. To change the order of execution, move elements and add or remove connectors.

Customize What Happens When a Flow Fails

If your flow contains an element that interacts with the Salesforce database—such as an Update Records element or a Submit for Approval core action—it can fail. Modify the default behavior by adding fault paths to all elements that can fail.

Working with Data in a Flow

The real power of a flow is that it can automate updates to your data, whether the data lives inside your Salesforce org or in an external database. In a flow, you can look up values from records, connect to external systems, create records, update records, delete records—the whole shebang!

Integrate with External Systems from a Flow

With Get Records elements, you can easily look up your Salesforce data in a flow. But what if you need data that lives outside of Salesforce? To connect your flow to an external database, use platform events, external objects, Lightning components, External Services, or Apex.

Show Users Progress Through a Flow with Stages

Keep users informed about which stage they’re in or how far they’ve progressed in a flow. For example, show where in a purchasing flow the user is with breadcrumbs or a progress indicator.

Get User Input with Screen Flow Components

Use screen flow components that accept choice resources to enable users to select choices from a list. Configure flow screen components to react to changes in other components on the same screen. Translate flow screen components.

Use Flows with Slack

Build a screen flow that’s designed to run in a Slack conversation or direct message group. Use another flow, a Slack shortcut, a slash command, or a button in a Slack view to run a flow from Slack.

Use Flows with MuleSoft RPA

Register a MuleSoft RPA process as an external service. Then, build a flow that starts and checks the status of the MuleSoft RPA process.

Extend Your Flow-Building Options

Sometimes your flow must do more than what Flow Builder provides out of the box. You can extend flows by calling Apex classes or adding Lightning components.

SEE ALSO:

Flow Runtime Accessibility Considerations

Build Blocks of Flows

Let Einstein Help You Build Flows (Beta)

Save time by describing what you want to automate and letting Einstein generative AI build you a draft flow to get you started with your automation.

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Note: You can use Einstein to build draft flows (beta) starting on July 16, 2024.

To use Einstein for Flow, turn on Einstein generative AI in Setup. Then, in Setup, find and select **Einstein for Flow (Beta)**, and activate it.

EDITIONS

Available in: Lightning Experience.

Available in: all **Einstein 1 Editions**

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Service, or Einstein for Platform add-on.

Best Practices and Considerations When Creating Flows with Einstein (Beta)
Get the best results when using generative AI to build draft flows.

Example Instructions for Creating Flows with Einstein (Beta)
To build an accurate draft flow by using generative AI, Einstein needs clear and detailed instructions about what you want to automate. The example instructions are based on a sample automation idea, when the flow starts and the actions the flow takes. Look for a flow outcome that's similar to what you want to achieve and write your own instructions that are similar to the example instructions.

Build a Flow with Einstein (Beta)
Build your flow faster and easier by letting Einstein generative AI help you get started.

SEE ALSO:

Video: Get to Know Einstein for Flow (Beta)

Salesforce Help: About Einstein Generative AI

Best Practices and Considerations When Creating Flows with Einstein (Beta)

Get the best results when using generative AI to build draft flows.

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Note: You can use Einstein to build draft flows (beta) starting on July 16, 2024.

EDITIONS

Available in: Lightning Experience.

Available in: all **Einstein 1 Editions**

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Service, or Einstein for Platform add-on.

Write Clear and Detailed Instructions

Einstein needs clear and detailed instructions to build an accurate flow. Describe each step that Einstein must take, including how to start the automation, specific types of records to process, and actions to take. When writing instructions:

- Start the instructions with "Create a flow." Consider including the flow type. For example, "Create a screen flow."
- Explain when the flow starts or describe the type of flow to create. For example, "...that starts when a contact is created..." or "...that has a screen..."
- Include criteria to use to identify specific records. For example, "...accounts with an annual revenue over $500,000..."
- Describe the actions you want the flow to take. For example, "...send an email to the case owner..."

- Use the names of objects and fields to use for criteria and actions. For example, “Create a flow that runs when an opportunity owner is updated...” instead of, “Create a flow that runs when the employee in charge of a sale is updated.”

Watch this video for more information about writing instructions.

Watch a video

Check for Accuracy and Safety

Einstein needs a human in the loop to ensure a flow is set up to complete the actions as intended and that it doesn’t contain problematic wording. After your draft flow opens in Flow Builder:

- Check each element to make sure it’s configured properly and doesn’t contain errors. For example, if you’re updating a field in the flow, ensure that the correct field name and new value are used in the Update Records element.
- If your flow includes outgoing messages, such as an email, check the language used in the message to ensure it doesn’t contain any harmful or unprofessional wording.
- Debug and test your flow to ensure it behaves the way you expect.

Provide Feedback

Einstein is still learning how to build accurate flows. Some flows created by Einstein aren’t built the way you expect. When a flow is built incorrectly, your feedback about what’s wrong with the flow helps Einstein continue to improve and build increasingly accurate draft flows for you. To give feedback, in Flow Builder, hover over the message in the Einstein window and click thumbs up or thumbs down. If you select thumbs down, provide details about why the flow didn’t meet your needs.

If at First You Don’t Succeed, Try New Instructions

Sometimes your draft flow needs minor tweaks to get it running correctly. Other times the flow isn’t at all what you hoped for. If your flow has many problems, start over with modified instructions. Seeing what your initial instructions produced gives you clues about how to change your instructions for your next attempt.

Considerations

- Supported flow types are autolaunched, record-triggered, schedule-triggered, and screen.
- Einstein can generate flows with up to about six elements. If you need a more complex flow, let Einstein create the first part and add elements after the draft flow opens in Flow Builder.
- Standard and custom objects are supported.
- Advanced formulas record variables, Apex actions, and invocable methods aren’t supported.
- In rare instances, too much custom object and custom field data can overload Einstein. If you're trying to create flows that use custom objects or custom fields, and you're repeatedly experiencing the Content Too Large error, turn off the usage of custom objects and custom fields. In Setup, search for and select **Process Automation Settings**. Then click **Ignore references to custom objects and fields when creating flows with Einstein generative AI**.

SEE ALSO:

Example Instructions for Creating Flows with Einstein (Beta)

Build a Flow with Einstein (Beta)

Video: Get to Know Einstein for Flow (Beta)

Salesforce Help: About Einstein Generative AI

Example Instructions for Creating Flows with Einstein (Beta)

To build an accurate draft flow by using generative AI, Einstein needs clear and detailed instructions about what you want to automate. The example instructions are based on a sample automation idea, when the flow starts and the actions the flow takes. Look for a flow outcome that's similar to what you want to achieve and write your own instructions that are similar to the example instructions.

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Note: You can use Einstein to build draft flows (beta) starting on July 16, 2024.

EDITIONS

Available in: Lightning Experience.

Available in: all **Einstein 1 Editions**

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Service, or Einstein for Platform add-on.

What You Want to Automate	When the Flow Starts	What the Flow Does	Example instructions
When a new lead comes in, create a task to remind the owner to call the lead in the next week.	When a record is created (a record-triggered flow)	Creates a record	Create a flow that starts when a lead is created and creates a task for the lead owner. The subject of the task is "Call new lead by next week."
Send an email to an account owner anytime a contact is added to their account.	When a record is created (a record-triggered flow)	Sends an email	Create a flow that sends an email notification to the account owner any time a new contact is added to the account.
When an account owner is changed, update the owner of all related opportunities and contacts.	When a record is updated (a record-triggered flow)	Updates multiple related records	Create a flow that starts when an account owner is changed and automatically updates all an account's opportunities and contacts.
If the Status field of an applicant record has been left blank, update the field to New.	When a record is created or updated (a record-triggered flow)	Uses a custom object and a custom field, updates a record that meets the criteria	Create a flow that starts when an applicant record is created. If the applicant Status field is empty, set it to New.
Every day, find contacts that have a missing phone number and update their description with a reminder to get the phone number at the next meeting.	On a specified day and time (a schedule-triggered flow)	Updates records that meet the criteria	Create a flow that runs every day at 8:00 AM and finds contacts with a blank Phone Number field and updates the description field to say "Get the contact's phone number at the next meeting."
Each Wednesday, find opportunities that haven't been	On a specified day and time	Sends an email to a	Create a flow that runs every Wednesday at 10:00 AM. The flow finds all opportunities that

What You Want to Automate	When the Flow Starts	What the Flow Does	Example instructions
updated in the last 30 days and send an email to their owners to remind them to update their opportunities.	(a schedule-triggered flow)	group of recipients	were created within the last 14 days and updates the Opportunity Next Step field to Assign Executive Sponsor.
Collect information in a form and use that information to create a lead.	When a user clicks a button on a screen (a screen flow)	Creates a record by using information from a form	Create a flow that has a screen and collects a first name, last name, and email address. Then, the flow creates a lead with the information collected on the screen.
Display a screen that shows support agents their open cases and allows them to upload a file to all those cases.	When a user clicks a button on a screen (a screen flow)	Shows information on a screen and allows user interaction	Create a screen flow that looks up and displays all open cases and allows a support agent to select a case and upload an image file as a file attachment.

SEE ALSO:

Best Practices and Considerations When Creating Flows with Einstein (Beta)

Build a Flow with Einstein (Beta)

Video: Get to Know Einstein for Flow (Beta)

Salesforce Help: About Einstein Generative AI

Build a Flow with Einstein (Beta)

Build your flow faster and easier by letting Einstein generative AI help you get started.

EDITIONS

Available in: Lightning Experience.

Available in: all **Einstein 1 Editions**

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Service, or Einstein for Platform add-on.

User Permissions Needed	
To use Einstein to build draft flows:	Manage Flow

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Note: You can use Einstein to build draft flows (beta) starting on July 16, 2024.

To enable Einstein for Flow, turn on Einstein generative AI in Setup. Then, in Setup, find and select **Einstein for Flow (Beta)**, and activate it.

1. From the Flows section in Setup, click **New Flow**, or from the Automation Lightning app, click **New**.
2. Select **Let Einstein Help You Build**, and then click **Next**.
3. Write instructions in one of these ways.
   - To write instructions from scratch, describe what you want to automate in the Instructions field.
   - To get started with sample instructions, expand **Get Started with Sample Instructions** and select a sample. The sample instructions are automatically entered into the Instructions field. Modify the instructions as needed.

4. Click **Create Draft Flow**.
   The draft flow opens in Flow Builder.
5. Check the draft flow for accuracy and safety.
6. Provide feedback in the Einstein window by hovering over the Einstein message and clicking thumbs up or thumbs down.
7. If the flow doesn't meet your needs, take one of these actions.
   - Modify the flow.
   - Click **Edit Instructions** and modify your instructions. Then, click **Create** to create a flow with your updated instructions.
   - Try again in a new window by clicking **Start Over** in the Einstein window.
8. If the flow meets your needs:
   a. Debug and test the flow.
   b. Activate the flow.

SEE ALSO:

Best Practices and Considerations When Creating Flows with Einstein (Beta)

Example Instructions for Creating Flows with Einstein (Beta)

Salesforce Help: About Einstein Generative AI

Video: Get to Know Einstein for Flow (Beta)

Add and Edit Elements

Use elements to define actions that the flow can execute. When they're on the canvas, connect them to create an order of execution.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Add and Edit Elements in Auto-Layout

Create a flow with a streamlined interface that keeps your canvas neat and tidy.

Add and Edit Elements in Free-Form

Create a flow with a drag-and-drop interface that allows you to place elements anywhere.

Add and Edit Elements in Auto-Layout

Create a flow with a streamlined interface that keeps your canvas neat and tidy.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Add an Element:

In the place that you want to put the element, click ⊕, and then select the element. Configure the element.

To keep your changes and hide a properties window on the canvas, click ✕.

To keep your changes and hide a properties window that's not on the canvas, click **Done**.

Note: The new element is automatically connected to the elements before and after it on the path.

Edit an Element:

To edit an element, click it, and then click **Edit Element**. Make your changes.

To keep your changes and hide a properties window on the canvas, click ✕ .

To undo changes to the properties window one at a time, click ↶ .

To keep your changes and hide a properties window that's not on the canvas, click **Done**.

To undo all changes to the properties window, click **Cancel**.

Copy and Paste Elements:

To copy and paste a single element, click it, and then from the options in the element panel, select **Copy Element**. In the location where you want to place the element, click ⊕, and then click **Paste Elements**.

To copy and paste multiple elements, click **Select Elements**. Click ⊞ on each element that you want to copy. Then, click 📋 . In the location where you want to place the elements, click ⊕, and then click **Paste Elements**.

Cut and Paste an Element:

To cut an element, click the element, then from the options in the element panel, select **Cut Element**. To paste the element that you cut, in the desired location, click ⊕, and then click 📋 .

To cut a Decision or Wait element, click the element, then from the options in the element panel, select **Cut Element**. To keep the elements from a Decision outcome or Wait configuration path on the canvas after cutting, from the cut element panel dropdown, select the path that contains the elements that you want to keep. To cut the Decision or Wait element and all of its paths, in the cut element panel dropdown, select **None, cut all paths**.

Note: You must paste a cut element. You can cancel the cut and paste mode, which undoes the cut.

Delete an Element:

To delete an element, click the element, then from the options in the element panel, select **Delete Element**.

To delete a Decision or Wait element, click the element, then from the options in the element panel, select **Delete Element**. To keep the elements from a Decision outcome or Wait configuration path on the canvas after deletion, in the delete element panel dropdown, select the path that holds the elements to keep. To delete the Decision or Wait element and all of its paths, in the delete element panel dropdown, select **None, delete all paths**.

SEE ALSO:

Flow Elements

Add and Edit Elements in Free-Form

Create a flow with a drag-and-drop interface that allows you to place elements anywhere.

To add an element in free-form, drag the element that you want to use onto the canvas.

After creating an element, connect the element to another element to determine the order in which they're executed at run time. Don't forget to connect the Start element to another element.

If you want to change an element's configuration after you created it, double-click the element, make your changes, then click **Done**.

SEE ALSO:

Flow Elements

Move and Connect Elements to Change a Flow Route

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Define Conditions in a Flow

Control when a flow takes a specific decision outcome or waits for a specific resume event.

Before you begin, create the Decision or Wait element to add conditions to.

1. Set up the conditions.

   At run time, the conditions are evaluated in the order you specify.

Column Header	Description
`Resource`	Flow resource whose value you want to evaluate.
`Operator`	The available operators depend on the data type selected for `Resource`.
`Value`	`Resource` and `Value` in the same row must have compatible data types. Options: - Select an existing flow resource, such as a variable or screen component. - Create a resource. - Manually enter a literal value or merge field. When you add or subtract a number from a date value, the date adjusts in days, not hours.

2. Identify the logic between the conditions.

Option	Behavior for Decision Outcomes	Behavior for WAIT Configurations
`All Conditions Are Met`	If one of the conditions is false, the flow evaluates the next outcome's conditions.	If one of the conditions is false, the flow doesn't wait for the associated resume event.
`Any Condition Is Met`	If one of the conditions is true, the flow immediately takes this outcome's path.	If one of the conditions is true, the flow waits for the associated resume event.
`Custom Condition Logic Is Met`	When you select this option, provide the condition logic by entering up to 1000 characters. Use: - Numbers to refer to each condition - *AND*, *OR*, or *NOT* to identify which combination of conditions must true - Parentheses to group parts of the string together If you enter *AND*, it's the same as if you selected **All Conditions Are Met**. If you enter *OR*, it's the same as if you selected **Any Condition Is Met**. If you enter any other logic, make sure that you include a number for each condition.	

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Option	Behavior for Decision Outcomes	Behavior for WAIT Configurations
	For example, for `1 AND NOT(2 OR 3)`, the flow evaluates whether the first condition is `true`, and the second or third condition is `false`.	

SEE ALSO:

Flow Wait Conditions

Flow Elements: Wait

Flow Element: Decision

Creating Flow Formulas

Flow formulas help make your flows more dynamic by performing calculations, manipulating data, and making decisions based on field or variable values. Formulas can facilitate conditional logic, taking the flow down different paths based on criteria and enforcing validation rules to ensure data integrity.

Creating Flow Formulas with Flow Formula Builder

You can use Flow Formula Builder to build an expression in Formula-type resources or in the Collection Filter or Start elements of a record-triggered flow. To build your expression, select from a list of functions and operators. To catch errors as you work, check the formula syntax for each expression.

Creating Flow Formulas with Einstein (Beta)

Save time by using generative AI to build formulas in Flow Builder. Simply describe your formula and let Einstein figure out the functions and operators.

Flow Formula Considerations

When you create a formula resource or add validation to a screen input component, understand the behavior of formulas in flows.

Creating Flow Formulas with Flow Formula Builder

You can use Flow Formula Builder to build an expression in Formula-type resources or in the Collection Filter or Start elements of a record-triggered flow. To build your expression, select from a list of functions and operators. To catch errors as you work, check the formula syntax for each expression.

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Example:

- Automate a process for records where the closed date changed for US-based accounts that have more than 100 employees.
- Automate a process for new case records for Premier support accounts in North America and Europe that have revenue of over US$1,000,000.

SEE ALSO:

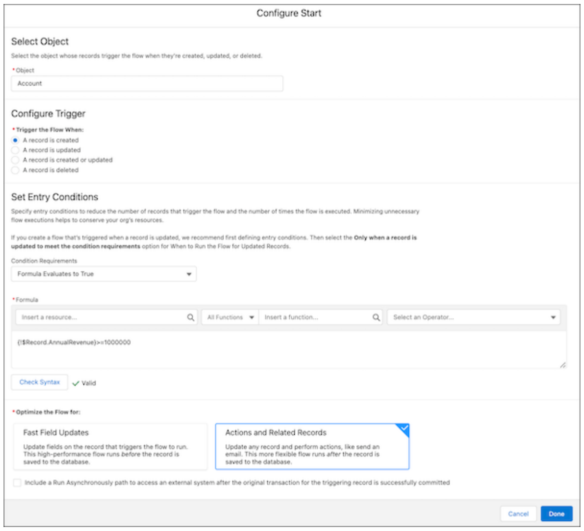
Trailhead: Flow Builder Logic

Examples of Advanced Formula Fields

Create Formulas in the Flow Start Element with Formula Builder

You can use Flow Formula Builder to build an expression in the Start element of a Record-triggered Flow. To build your expression, select from a list of functions and operators. To catch errors as you work, check the formula syntax for each expression.

1. Edit the Start element in a Record-Triggered flow.
2. Select the object with records that trigger the flow.
3. Select when to trigger the flow.
4. In the Set Entry Conditions section, set Condition Requirements to **Formula Evaluates to True**.
5. Insert a resource from the list of Global Variables.
   - You can add more than one resource to an expression.
6. Insert a function.
   - To see a list of functions, click **Insert a function**.
   - To filter functions by category, select the **All Functions** dropdown and choose the category.
7. Select an operator from the dropdown.
8. Complete the expression.
9. After you build each expression, click **Check Syntax**. The formula builder shows one error at a time.
10. Complete the configuration for the Start element.
11. Click **Done**.
12. Save your work.



Configure Start

Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

*Object: Account

Configure Trigger

*Trigger the Flow When:

● A record is created
○ A record is updated
○ A record is created or updated
○ A record is deleted

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements: Formula Evaluates to True

*Formula

```
{!$Record.AnnualRevenue}>=1000000
```

✓ Valid

*Optimize the Flow for:

Fast Field Updates

Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

✓ Actions and Related Records

Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

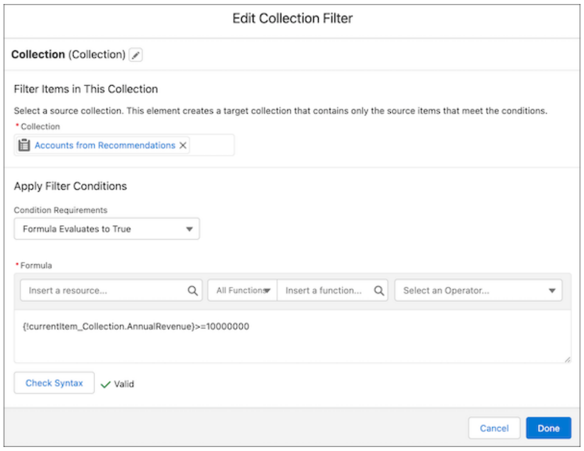
□ Include a Run Asynchronously path to access an external system after the original transaction for the triggering record is successfully committed

Create Formulas in the Flow Collection Filter with Formula Builder

You can use Flow Formula Builder to build an expression in the Collection Filter element of a flow. To build your expression, select from a list of functions and operators. To catch errors as you work, check the formula syntax for each expression.

1. Add an element to a flow.
2. Select **Collection Filter**.

3. Enter the **Label** and **API Name**.
4. Select a collection.
5. In the Apply Filter Conditions section, set Condition Requirements to **Formula Evaluates to True**.
6. Insert a resource from the list of Global Variables.
   - You can add more than one resource to an expression.
7. Insert a function.
   - To see a list of functions, click **Insert a function**.
   - To filter functions by category, select the **All Functions** dropdown and choose the category.
8. Select an operator from the dropdown.
9. Complete the expression.
10. After you build each expression, click **Check Syntax**. The formula builder shows one error at a time.
11. Click **Done**.
12. Save your work.



Edit Collection Filter

Collection (Collection)

Filter Items in This Collection

Select a source collection. This element creates a target collection that contains only the source items that meet the conditions.

* Collection: Accounts from Recommendations

Apply Filter Conditions

Condition Requirements: Formula Evaluates to True

* Formula

```
{!currentItem_Collection.AnnualRevenue}>=10000000
```

✓ Valid

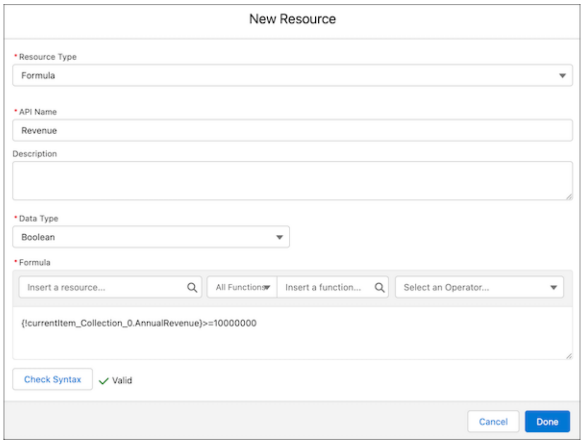
Create Formulas in Flow Formula-type Resources with Formula Builder

You can use Flow Formula Builder to build an expression in Flow Formula-type resources. To build your expression, select from a list of functions and operators. To catch errors as you work, check the formula syntax for each expression.

1. In Flow, in the Toolbox, in Manager, click **New Resource**.
2. From the Resource Type dropdown, select **Formula**.
3. Enter an **API Name**.
4. Select the **Data Type**.
5. Insert a resource from the list of Global Variables.
   - You can add more than one resource to an expression.
6. Insert a function.
   - To see a list of functions, click **Insert a function**.

- To filter functions by category, select the **All Functions** dropdown and choose the category.

7. Select an operator from the dropdown.
8. Complete the expression.
9. After you build each expression, click **Check Syntax**. The formula builder shows one error at a time.
10. Click **Done**.
11. Save your work.



New Resource

* Resource Type: Formula

* API Name: Revenue

Description:

* Data Type: Boolean

* Formula:

```
{!currentItem_Collection_0.AnnualRevenue}>=10000000
```

✓ Valid

SEE ALSO:

Flow Element: Start

Flow Element: Collection Filter

Flow Resource: Formula

Flow Formula Considerations

Formula Operators and Functions by Context

Creating Flow Formulas with Einstein (Beta)

Save time by using generative AI to build formulas in Flow Builder. Simply describe your formula and let Einstein figure out the functions and operators.

User Permissions Needed	
To use Einstein to create formulas:	Manage Flow

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

EDITIONS

Available in: Lightning Experience.

Available in: all **Einstein 1 Editions**

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Service, or Einstein for Platform add-on.

Guidelines and Considerations for Creating Formulas with Einstein (Beta)

Review our guidelines and considerations to get the best results when you're using generative AI to create formulas.

Write a Clear Description of Your Formula

To get an accurate formula, provide Einstein with a detailed description. Review these example descriptions to get an idea of what type of information Einstein is looking for.

Description	Formula
Check to see if the `{!NumberOfEmployees}` field is set to zero.	`BLANKVALUE(NumberOfEmployees, 0)..`
Check if `{!$Event.EndDateTime}` is earlier than the current time	`{!$Event.EndDateTime} < NOW()`
Check if `{!$Opportunity.IsClosed}` is open and `{!$Opportunity.Amount}` is over $50,000.	`AND({!Opportunity.Amount} > 50000, {!Opportunity.IsClosed} = false)`

To get the most accurate formula, when including a resource in your description, use the resource menu to select the resource instead of writing it in manually. To select a resource from the resource menu, click **Insert a resource...**.

Check Your Formula for Accuracy and Safety

Einstein needs a human in the loop to make sure a formula is set up correctly and doesn't output problematic language. After your formula is created, review it for accuracy and safety, and check the syntax. Then, test it in your flow to make sure it works as expected.

If Your Formula Doesn't Work, Try Again

Einstein can give a different result each time you click **Create** to create a formula. So, if your formula isn't what you expect, try your description again to see if you can get a valid result. Otherwise, modify the formula, or start over with a new description.

Consideration

Formulas created by Einstein are available only in Flow Formula Builder. For example, they are not supported in Screen element input validation formulas.

Create a Start Element Criteria Formula with Einstein (Beta)

Quickly add formula entry criteria to a Start element with Einstein generative AI.

Turn on Einstein generative AI in Setup. Then find and select **Process Automation Settings**, and enable formulas created by Einstein.

1. In a flow, open the Start element.
2. In the Set Entry Conditions section, for Condition Requirements, select **Formula Evaluates to True**.
3. In the formula builder, click ✨.
4. Describe the formula you want to create and click **Create**.
   The formula appears in the area where you build a formula from scratch.
5. Check the syntax and test the formula before activating the flow.

Create a Formula Resource with Einstein (Beta)

Create a formula resource with ease by using Einstein generative AI to create the formula.

Turn on Einstein generative AI in Setup. Then find and select **Process Automation Settings**, and enable formulas created by Einstein.

1. In a flow, open the Toolbox and click **New Resource**.
2. For Resource Type, select **Formula**.
3. In the formula builder, click ✨.
4. Describe the formula you want to create and click **Create**.
   The formula appears in the area where you build a formula from scratch.
5. Check the syntax and test the formula before activating the flow.

Flow Formula Considerations

When you create a formula resource or add validation to a screen input component, understand the behavior of formulas in flows.

Formula Functions

- These functions aren’t supported in a flow formula.
  - GETRECORDIDS
  - HOUR
  - IMAGE
  - INCLUDES
  - PARENTGROUPVAL
  - PREVGROUPVAL
  - PRIORVALUE
  - REQUIRE SCRIPT
  - TIMENOW
  - TIMEVALUE
  - VLOOKUP

  For a complete list of operators and functions for building formulas in Salesforce, see Formula Operators and Functions by Context.

- In a flow, the `CONTAINS` function checks all characters within its parentheses. For cross object field references, `CONTAINS` works like it does in the rest of Salesforce. It checks only the first 250 characters in the reference.

  Here’s an example. `varContract` refers to a record variable that contains the values of a contract record. This formula expression checks only the first 250 characters.

  ```
  CONTAINS({!varContract.Account.Description}, "description")
  ```

  This formula expression checks all characters in the field.

  ```
  CONTAINS({!varContract.Description}, "description")
  ```

Flow Data in Formulas

- References to global variables in formulas aren’t validated.

- To reference a platform event in a formula, pass the event data into a record variable in the Wait element. Then reference the appropriate field in that record variable.
- To evaluate a `null` field value as a zero value, use the BLANKVALUE function. For example, to evaluate a `null` value for the `NumberOfEmployees` field as a zero value, use *`BLANKVALUE(NumberOfEmployees, 0)`*.

Gotchas

- A flow formula can contain up to 3,900 characters.
- When you paste a formula, make sure that the apostrophes and quotation marks are straight (") and not curly (“). Otherwise, you get a syntax error when you save the flow.
- Record formula fields are evaluated in flows exactly as they’re evaluated in Salesforce Classic. When a record formula field checks NOT(null/blank), it evaluates to FALSE.

Errors

A formula returns `null` if an error occurs when the expression is evaluated, such as:

- The value that the formula returns doesn’t match its data type.
- The formula contains an unsupported function.

  For example, if your formula resource has a data type of Number, the output must be numeric.

If a flow contains an invalid formula resource, you can’t activate the flow. If a Display Text screen component contains an invalid formula resource, the flow displays an empty string at run time.

SEE ALSO:

Flow Resource: Formula

Flow Resources

Creating Flow Formulas with Flow Formula Builder

Save a Draft of Your Flow as You Build

You can interact with your flow while an element is open and access the toolbar while editing that element. And you don’t need to finish configuring an element before saving your flow.

These steps show you how you can save an in-progress flow with errors and see the warnings the draft generates.

The property window for most elements is now a panel on the Flow Builder canvas. A panel-based property editor isn't available for Screen and Action elements.

Panel-based property editors don’t work in free-form. You can’t save a draft in free-form.

1. From Setup, in the Quick Find box, enter *`Flows`*, select **Flows**, and then click **New Flow**.

   Flow Builder opens.

2. Select **Start from Scratch**, and then click **Next**.
3. Select a flow type, and then click **Create**.
4. Add any element except a Screen or Action to the canvas.
5. Configure the element in the property editor. To intentionally create an error, leave some of the configuration values unspecified.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

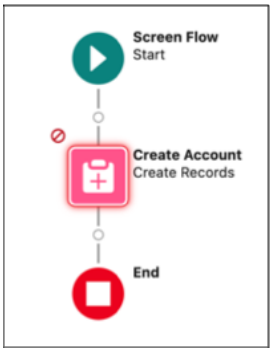
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

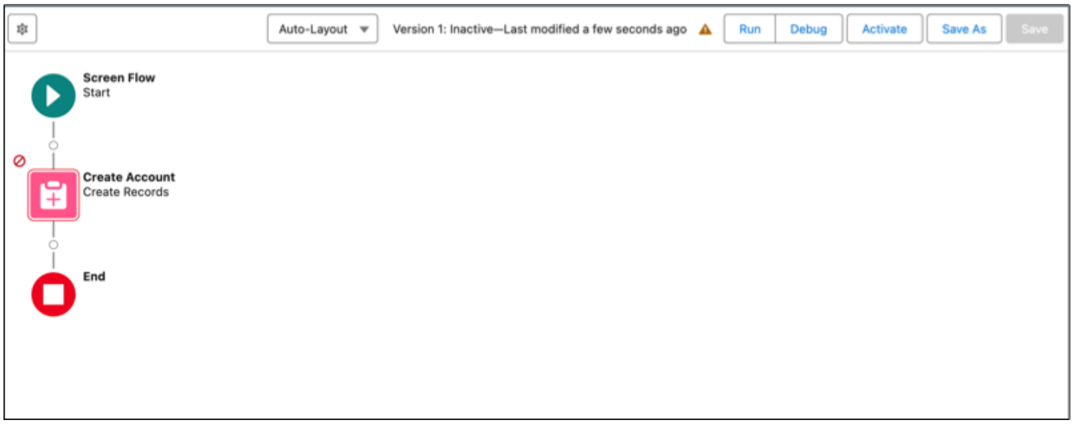
- Manage Flow

6. To retain your changes and return to the canvas, click ✕ .
7. Review the draft of your flow on the canvas. Note the error icon.

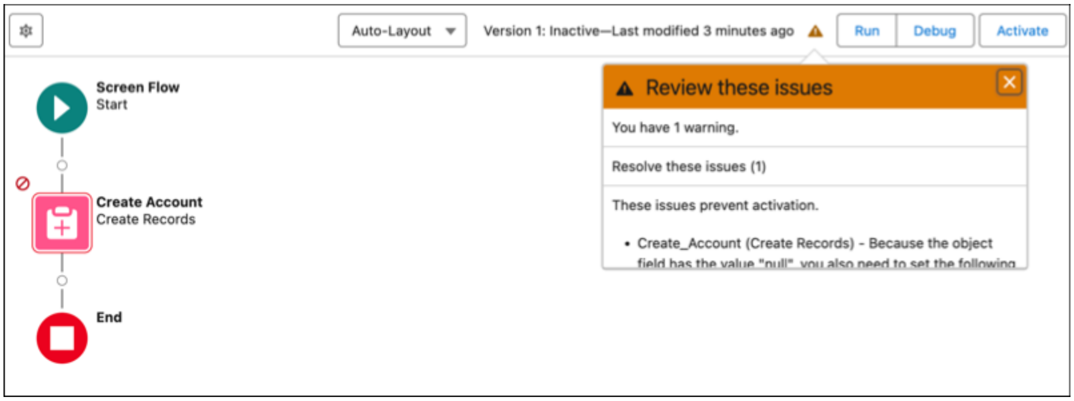


8. To see warnings about errors in the draft, save the flow.

   Your flow is saved, despite having errors. The warning icon (⚠) indicates that there are warnings for the flow.



9. To view the details of the warnings about errors in the draft flow, click ⚠



Review these issues

You have 1 warning.

Resolve these issues (1)

These issues prevent activation.

- Create_Account (Create Records) - Because the object field has the value "null", you also need to set the following

10. Continue configuring your flow until there are no errors.

Move and Connect Elements to Change a Flow Route

Identify which elements the flow executes and in what order by connecting the elements on your canvas together. To change the order of execution, move elements and add or remove connectors.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Route a Flow in Auto-Layout

In auto-layout, Flow Builder automatically creates connectors when an element is added, and it removes connectors when an element is deleted. Place elements in the desired order, and move them to change the order of execution.

Move a Single Element in Auto-Layout

You can't directly move elements in auto-layout. So, to change a single element's location, cut the element, and then paste it to its new location.

1. Click the element that you want to move.
2. Select **Cut Element**.
3. Where you want to place the element, click ⊕, and then click 📋 to paste.

When cutting a Decision or Wait element, you can leave the elements from an outcome or configuration path on the canvas after cutting. To keep elements from a path on the canvas, select the path in the cut element panel. To cut a Decision or Wait element and all of its paths, select **None, cut all paths.** from the cut element panel.

Move Multiple Elements in Auto-Layout

To move multiple elements, select all the elements that you want to move, copy them, paste them to their new location, and then delete the original elements.

1. Click **Select Elements**.
2. Click ⊞ on each element that you want to move.
3. To copy the selected elements, click 📋.
4. Where you want to place the elements, click ⊕.
5. Select **Paste Elements**.
6. For each copied element, click the element, and select **Delete Element**.

Connect Nonconsecutive Elements in Auto-Layout

In Auto-Layout, elements on the canvas are spaced and connected automatically. To connect elements that aren't automatically connected, use a Go To connector.

1. To change a connector to a Go To connector, directly after the element that you want to connect from, click ⊕.
2. Click **Connect to element**.
3. Click ⊞ on the element that you want to connect to.

Route a Flow in Free-Form

In free-form, you create and remove connectors that determine the order of execution for the elements on the canvas.

1. On the canvas, find the node at the bottom of the source element.

2. Drag the node onto the target element.
3. If prompted, select which decision outcome or wait configuration to assign to the path.

Remove Connectors from a Flow in Free-Form

When you need to change the execution path of a flow in Free-Form layout, delete the connector and then add a new one.

1. In your flow, select the connector to delete.

   When you select a connector, its color changes from gray to blue.

2. Press the DELETE key.

If you delete a connector for a specific outcome, the outcome isn't deleted from the source element. However, if you delete an outcome from a decision element, the outcome's connector is also deleted. The same applies to pause configurations.

SEE ALSO:

Flow Connectors

Customize What Happens When a Flow Fails

Customize What Happens When a Flow Fails

If your flow contains an element that interacts with the Salesforce database—such as an Update Records element or a Submit for Approval core action—it can fail. Modify the default behavior by adding fault paths to all elements that can fail.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

What Happens When a Flow Fails?

When you're deciding whether to customize the error handling in your flow, consider how a failed flow behaves.

Configure Every Fault Path to Send You an Email

As a best practice, we recommend configuring the fault connectors in your flow so that you always receive an email when a flow fails. In the email, include the current values of all your flow's resources. The resource values can give you insight into why the flow failed.

Customize the Error Message for Running Flow Users (Best Practice)

As a best practice, we recommend displaying a better message to your user than "An unhandled fault has occurred in this flow". Display a new message only if the distribution method you're using supports flows that contain screens. In other words, don't do it if your flow is distributed through a process.

Other Examples of Error Handling in Flows

Examples of using fault connectors to handle flow errors include requesting corrections from the user and bypassing the error.

SEE ALSO:

Flow Connectors

Flow Elements

Move and Connect Elements to Change a Flow Route

What Happens When a Flow Fails?

When you're deciding whether to customize the error handling in your flow, consider how a failed flow behaves.

When a flow fails, the user running the flow gets this error message.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

```
An unhandled fault has occurred in this flow
An unhandled fault has occurred while processing
the flow. Please contact your system administrator
for more information.
```

The running user can't proceed with the flow or return to a previous part of the flow. The admin who created the flow receives a fault email. The email details the element that failed, the error message from that element, and which elements were executed during the failed interview. Here's an example error message.

```
An error occurred at element Delete_1.
DELETE --- There is nothing in Salesforce matching your
delete criteria.
```

To view the debug details directly in Flow Builder, click **Flow Error: Click here to debug the error in Flow Builder** in the fault email. This link appears only for certain flow types.

For screen flows, Salesforce publishes a Flow Execution Error Event platform event message.

SEE ALSO:

- Customize What Happens When a Flow Fails
- Troubleshoot Flow Errors
- Send Alerts When a Screen Flow Fails

Configure Every Fault Path to Send You an Email

As a best practice, we recommend configuring the fault connectors in your flow so that you always receive an email when a flow fails. In the email, include the current values of all your flow's resources. The resource values can give you insight into why the flow failed.

1. Create a text template that includes the current values of the flow's resources.

   Use a text template so you can see what the exact values of flow variables were when the interview failed. If the flow contains screens, you also see exactly what the user entered and selected.

   In this example, we create the ErrorEmailBodyTextTemplate text template.

```
Error: {!$Flow.FaultMessage}

RESOURCE VALUES
Customer Response: {!CustomerResponseRadioButtons}
Value of Decision's Yes outcome: {!IsWantsToParticipate}
Company: {!CompanyNameText}
Satisfaction: {!SatisfactionPicklist}
Service: {!ServiceText}}
Other Comments:
{!OtherCommentsText}
```

2. Add a fault path to an element in your flow.
   In this example, we added a fault path to the **Create Survey Records** element.

EDITIONS

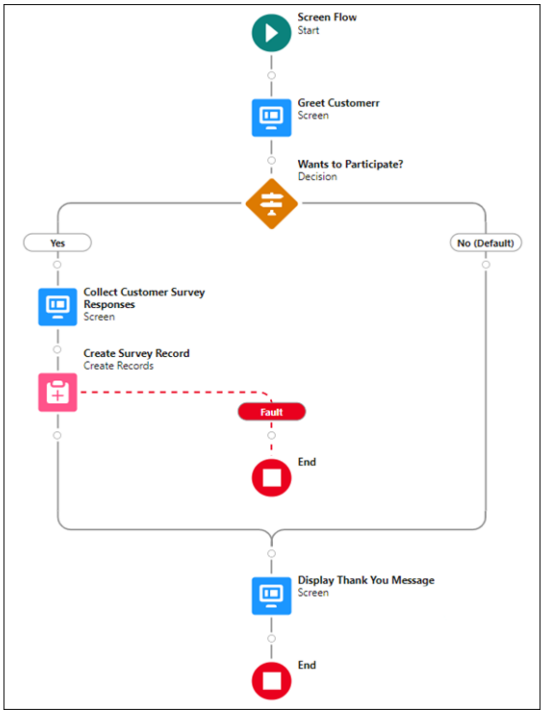
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

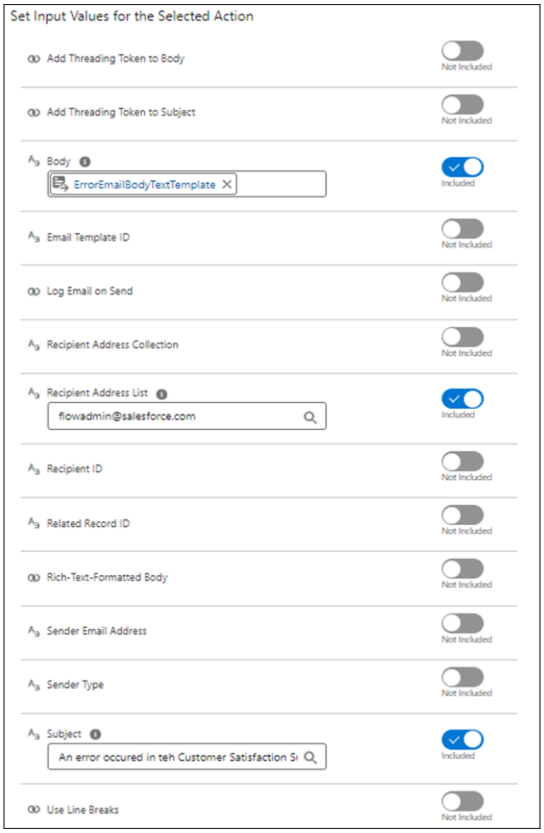
To open, edit, or create a flow in Flow Builder:

- Manage Flow



3. Add a Send Email action to the fault path in the flow.
   a. For **Body**, enter the API name of the text template.
   b. For **Recipient Address List**, enter your email address.
   c. For **Subject**, enter a string to use as the email subject.

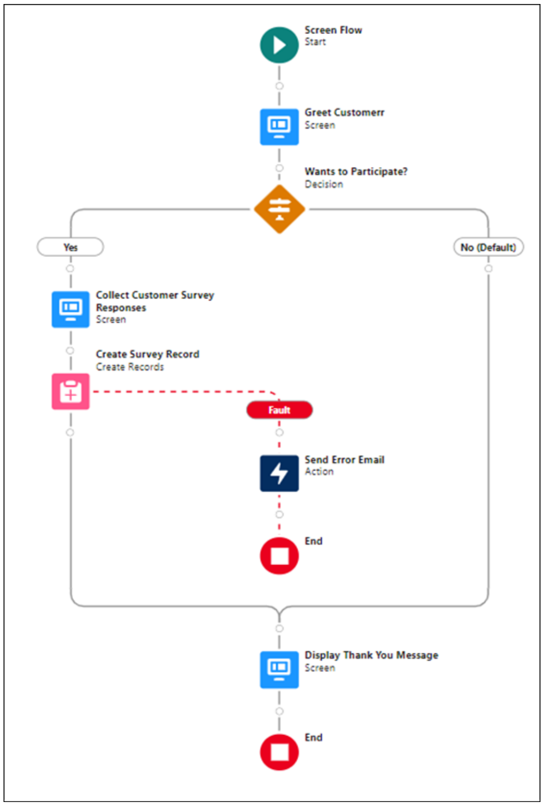
   In this example, we name the element `Send Error Email`, and set **Body** to `{!ErrorEmailBodyTextTemplate}`.



Set Input Values for the Selected Action

Add Threading Token to Body		Not Included
Add Threading Token to Subject		Not Included
Body	ErrorEmailBodyTextTemplate	Included
Email Template ID		Not Included
Log Email on Send		Not Included
Recipient Address Collection		Not Included
Recipient Address List	flowadmin@salesforce.com	Included
Recipient ID		Not Included
Related Record ID		Not Included
Rich-Text-Formatted Body		Not Included
Sender Email Address		Not Included
Sender Type		Not Included
Subject	An error occured in teh Customer Satisfaction S	Included
Use Line Breaks		Not Included

The fault path now connects to the Send Error Email element.



4. From every other element that can fail, add a fault path to the Send Error Email element.

   In this example, Create Survey Record is the only element that supports fault paths.

SEE ALSO:

Flow Resource: Text Template

Flow Core Action: Send Email

Customize What Happens When a Flow Fails

Customize the Error Message for Running Flow Users (Best Practice)

As a best practice, we recommend displaying a better message to your user than “An unhandled fault has occurred in this flow”. Display a new message only if the distribution method you’re using supports flows that contain screens. In other words, don’t do it if your flow is distributed through a process.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

1. Create a text template that contains a friendlier error message.

```
<p><b>Something went wrong with this flow.</b></p>
<p>Your admin has received an email about this error.</p>
```

2. Add a Screen element. In a Display Text screen component, reference the text template.
3. For every element that can fail, draw a fault connector to the Screen element.

SEE ALSO:

Flow Screen Output Component: Display Text

Flow Resource: Text Template

Customize What Happens When a Flow Fails

Other Examples of Error Handling in Flows

Examples of using fault connectors to handle flow errors include requesting corrections from the user and bypassing the error.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Request Corrections from Users

Draw a fault connector to a Screen element, where users can verify the values that they entered, make corrections, and proceed.

Display the Error Message

If the flow is used only internally, such as at a call center, use the fault path to display the error message to the running user. In the same Screen element, ask the user to report the error to the IT department. To do so, draw the fault connector to a Screen element with this Display Text field.

```
Sorry, but you can't read or update records at this time.
Please open a case with IT and include this error message:
{!$Flow.FaultMessage}
```

Create a Case

When an error occurs, automatically create a case that includes the error message and assign it to your IT department. Assign the created case's number to a Text variable (`{!caseNumber}`, for example). Then, in a Screen, display this message to the running user.

```
Sorry, but you can't read or update records at this time.
We filed case {!caseNumber} for you.
```

Ignore Errors

To bypass errors for a given element in your flow, draw the fault connector to the same element as the normal connector.

SEE ALSO:

Customize What Happens When a Flow Fails

Working with Data in a Flow

The real power of a flow is that it can automate updates to your data, whether the data lives inside your Salesforce org or in an external database. In a flow, you can look up values from records, connect to external systems, create records, update records, delete records—the whole shebang!

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Create a Salesforce Record from a Flow

To create a Salesforce record, use either the Create Records or Quick Action element. The right element depends on what the rest of your flow is doing.

Clone Records with a Create Records Element

A flow can clone records in your org. First, populate a record variable with an existing record's values. Identify fields that the running user can't edit, and map all remaining fields to another record variable. Then use the second record variable in a Create Records element to clone the record.

Update Salesforce Records from a Flow

To update field values on existing Salesforce records, use either the Update Records element or a Quick Action core action. The right element depends on what the rest of your flow is doing.

Generate Flow Choice Options From External Data

If your business process interacts with external data, your users can select from it on flow screens.

Modify Records from User Input in Screens

Enable end users to change a collection of records in a screen flow. The Repeater component can reference a collection of data to prepopulate values in child components that end users update at run time.

Sum or Count Items in Collections with the Transform Element

Aggregate data from a source collection to calculate the sum or count of items in that collection, and assign the result to a target data field.

Create a Salesforce Record from a Flow

To create a Salesforce record, use either the Create Records or Quick Action element. The right element depends on what the rest of your flow is doing.

If you want to use a combination of the values from a record variable and values from other resources (like single-value variables or screen input fields), use either a Create Records or Quick Action element. Those two elements differ in these ways.

- Which fields are available in the elements
- Whether the element provides any required fields for the object
- Whether the element lets you store the new record's ID

  Storing the ID is useful, for example, if you create an account and then want to create a contact that's associated with that account (which you obviously need the ID for).

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

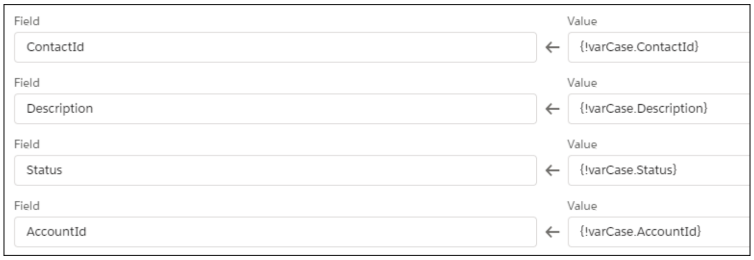
	Field Availability	Required Fields	New Record ID
Create Records	Every field on the object. You manually select the object and every field you want to have a value.	Not indicated	Lets you store the ID of the created record to use later in your flow.
Quick Action (of type Create)	Only fields that are included in the Quick Action layout. If you supplied default values for certain fields when you created the quick action, those values are used when the record is created.	Indicated Requiredness is based on what's marked required in the action layout.	Doesn't let you store the created record's ID for use later.

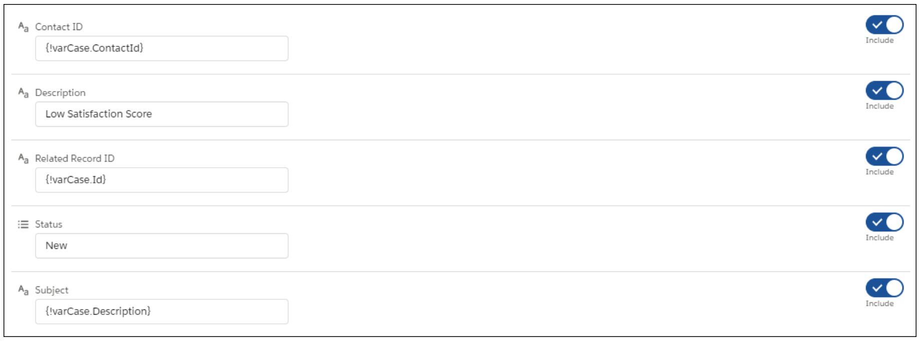
Tip: Use a Quick Action element when all these statements are true.

1. The action is of type Create.
2. The action's layout includes all the fields that you want to update.
3. You don't need to reference the new record's ID later in the flow.

Otherwise, use the Create Records element.

Here's how you'd create a case when a customer's satisfaction score is too low by using each of the create elements.

Create Records	
	Field ContactId ← Value {!varCase.ContactId}
	Field Description ← Value {!varCase.Description}
	Field Status ← Value {!varCase.Status}
	Field AccountId ← Value {!varCase.AccountId}
	You can set any field on the record, but the Create Records element doesn't know which fields are required for this object.

Quick Action (of type Create)	 Contact ID: {!varCase.ContactId} — Include ✓ Description: Low Satisfaction Score — Include ✓ Related Record ID: {!varCase.Id} — Include ✓ Status: New — Include ✓ Subject: {!varCase.Description} — Include ✓ These fields are the only fields that you can set for this element, because they're the only ones available from the action layout.

Clone Records with a Create Records Element

A flow can clone records in your org. First, populate a record variable with an existing record's values. Identify fields that the running user can't edit, and map all remaining fields to another record variable. Then use the second record variable in a Create Records element to clone the record.

Before you begin, review Which Fields Are Inaccessible When a Flow Creates or Updates Records?

1. Populate a record variable with the values from the existing record.
   For example:
   - Look up the record with a Get Records element.
   - Obtain the record from a Flows action in a process.
2. In an Assignment element, copy the writable field values to a new record variable.

   Important: Make sure that `Id` isn't set in the new variable.
3. Add a Create Records element to your flow. Choose to create one record and set the record fields by using all the values from a record variable. Select the second record variable to populate the values of the cloned record.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:
- Manage Flow

SEE ALSO:

Flow Element: Create Records

Update Salesforce Records from a Flow

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To update field values on existing Salesforce records, use either the Update Records element or a Quick Action core action. The right element depends on what the rest of your flow is doing.

On an opportunity record, when a user clicks the "Won" button, a flow updates the opportunity's stage.

To update fields on one or more existing Salesforce records, your flow:

1. Identifies the records to update.
2. Identifies the new field values for those records.
3. Saves those changes to the Salesforce database. (In other words, until the changes are saved to the database, the changes exist only within the flow.)

How do I choose between flow elements that update records?

The main difference between the elements is how it knows which records to update, how it knows the new field values to apply, and how many records it can update.

Quick Action elements can update only one record at a time, while Update Records elements can update multiple records.

	To identify records to update	To identify new field values for the records	Number of records it updates
Update Records, where you chose to specify conditions and set fields individually	In the same element, use filter criteria.	In the same element, map each field that must be updated with a variable or other resource. All resources are supported, so long as the resource's data type matches the selected field's data type.	At least one.
Update Records, where you chose to use the IDs and field values from a record variable or record collection variable	Populate a record variable or record collection variable in another element.	In another element, such as an Assignment element, update the values in the record variable or record collection variable.	If a record variable: one. If a record collection variable: at least one.
Quick Action	Populate a single-value variable with the ID in another element. Use this ID for the Related Record ID parameter.	In the same element, map each field that must be updated with a variable or other resource. All resources are supported, so long as the resource's data type matches the selected field's data type.	Exactly one.

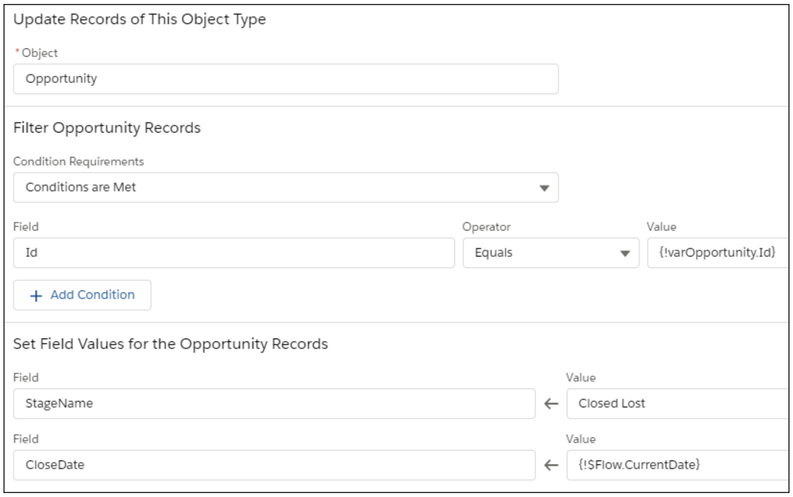
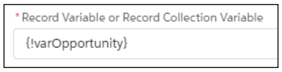
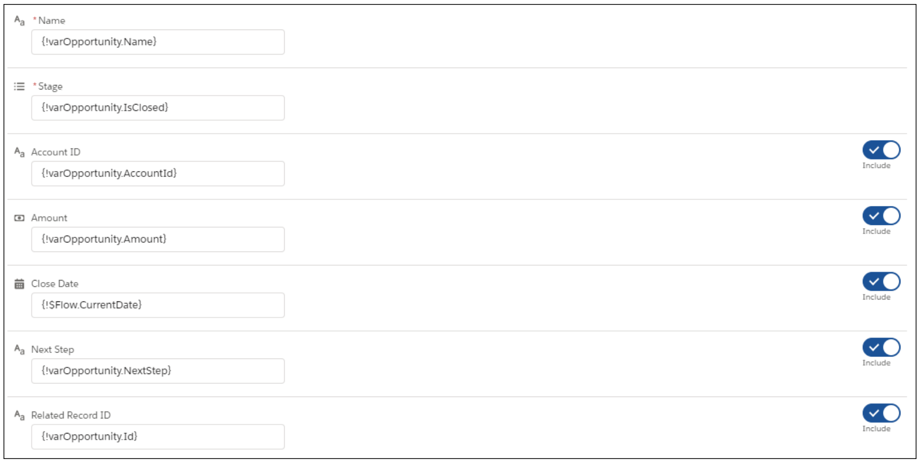
If all the following statements are true, use a Quick Action element.

- You must update exactly one record
- You've already populated a variable with the record's ID

- The Quick Action's layout includes all the fields you must update

If any of those statements aren't true, use an Update Records element.

Here's how you'd update an opportunity's stage by using each of the update elements.

Update Records, where you chose to specify conditions and set fields individually	 Update Records of This Object Type * Object: Opportunity Filter Opportunity Records Condition Requirements: Conditions are Met Field: Id · Operator: Equals · Value: {!varOpportunity.Id} Set Field Values for the Opportunity Records Field: StageName ← Value: Closed Lost Field: CloseDate ← Value: {!$Flow.CurrentDate}
Update Records, where you chose to use the IDs and field values from a record variable or record collection variable	 * Record Variable or Record Collection Variable: {!varOpportunity} Assumes {!svarOpportunity} is already populated with the right fields.
Quick Action (of type Update)	 * Name: {!varOpportunity.Name} * Stage: {!varOpportunity.IsClosed} Account ID: {!varOpportunity.AccountId} ✓ Include Amount: {!varOpportunity.Amount} ✓ Include Close Date: {!$Flow.CurrentDate} ✓ Include Next Step: {!varOpportunity.NextStep} ✓ Include Related Record ID: {!varOpportunity.Id} ✓ Include Two fields are required by the associated action layout, so they're required in this element. Related Record ID identifies which opportunity to update.

Generate Flow Choice Options From External Data

EDITIONS

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow
-

If your business process interacts with external data, your users can select from it on flow screens.

You can allow users to select external data.

1. Create a collection choice set and reference an Apex-defined collection from an external service, Apex action, or another screen component.
2. Add your collection choice set to the appropriate choice component, such as a picklist.

Example: For example, a car dealership stores car inventory data outside of Salesforce. They have an external service to access the inventory, and now their users can select which cars to view.

At runtime, the picklist options are generated based on whatever data was populated in the referenced Apex-defined collection.

Modify Records from User Input in Screens

EDITIONS

Available in: Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Enable end users to change a collection of records in a screen flow. The Repeater component can reference a collection of data to prepopulate values in child components that end users update at run time.

When the Repeater component includes a record collection data source, a repeater instance is generated at run time for each item in that collection. The values that are shown from the collection depend on how you map the values into the repeater's child components.

At run time, users can add up to 30 instances of the Repeater component to the screen.

Here’s an example where the goal is to support an end user changing the beneficiaries for an insurance policy. Because you can’t know how many beneficiaries a policy contains, you add the Repeater component to prepopulate existing beneficiary information. By setting a data source for the Repeater component, you can enable users to easily adjust details for beneficiaries. At run time, each Repeater instance contains the details about each beneficiary that are associated with the policy, including name, contact information, and benefit allocation.

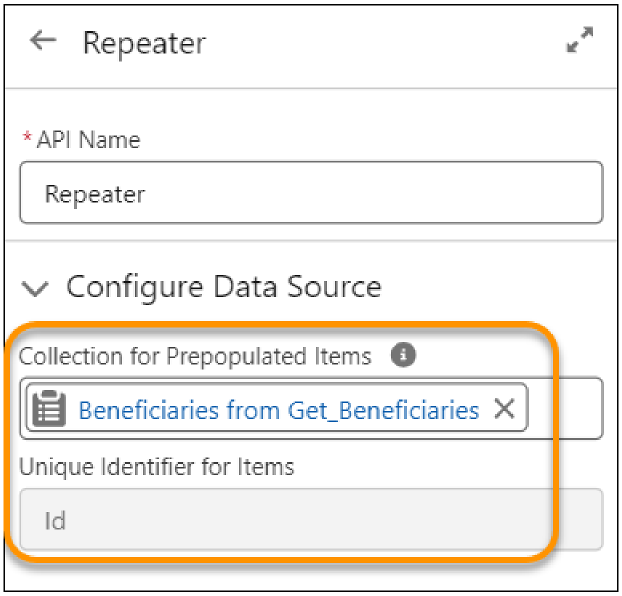
Note: These steps use the Create Records element, which creates or updates existing records.

1. Create a screen flow.
2. Add the Get Records element.

   For example, use the element to retrieve existing beneficiaries for this insurance policy.

3. Add the Screen element.
4. Add the Repeater component to the screen.
   a. For the Collection for Prepopulated Items field, select a collection. The Collection for Prepopulated Items field supports only Record data types. In this example, select the collection of beneficiary records.

      Fields from the selected collection become available to child components in the Repeater. The unique identifier for items is the API name of the field that contains a unique identifier for each item in the collection. This field is set automatically to the object’s ID field.



Repeater

* API Name

Repeater

Configure Data Source

Collection for Prepopulated Items

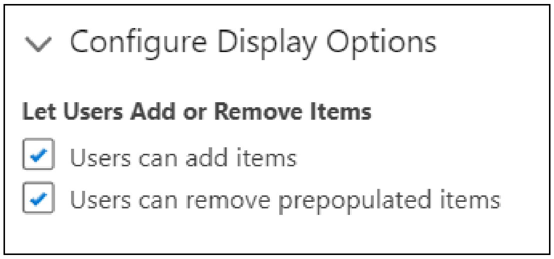
Beneficiaries from Get_Beneficiaries

Unique Identifier for Items

Id

b. Choose to let users add new items or remove prepopulated items.

For example, sometimes you don't want to give end users the ability to remove items that are required. Users can always remove items that were added manually. In this example, select both options.



Configure Display Options

Let Users Add or Remove Items

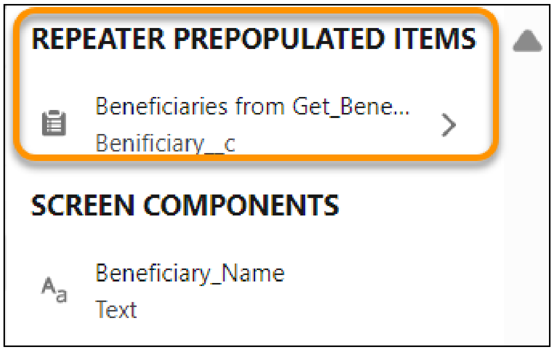
☑ Users can add items

☑ Users can remove prepopulated items

c. Add the relevant child components into the Repeater.

d. For child component inputs that reference a value in the collection for prepopulated items, under Repeater Prepopulated Items, select an item on the resource.

For example, set the Default Value for the Text component or the Value for the Toggle component.



REPEATER PREPOPULATED ITEMS

Beneficiaries from Get_Bene...
Benificiary__c

SCREEN COMPONENTS

Beneficiary_Name
Text

5. To process the results of the Repeater so that you can later save the changes in the Salesforce database, add the Loop element.

a. For Collection Variable, select the relevant output collection from the Repeater.

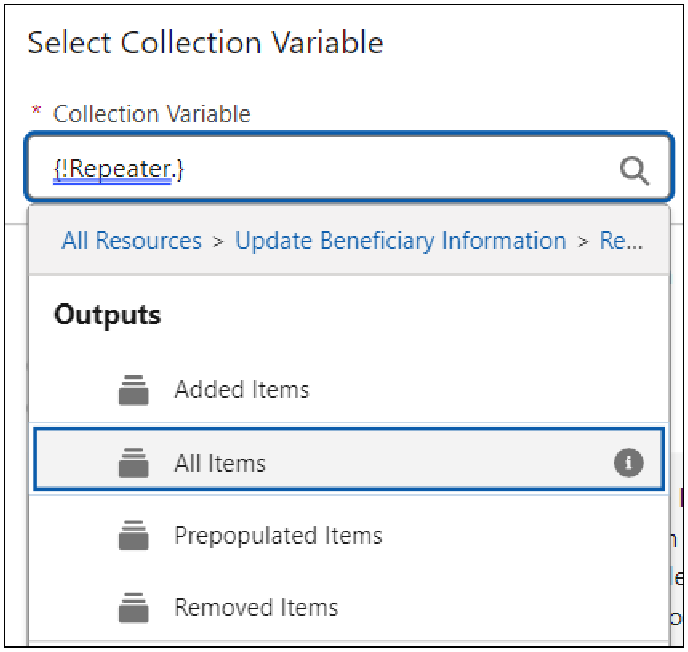
Added Items
: The items the user added manually to the Repeater component that weren't prepopulated from the collection for prepopulated items at run time. We recommend selecting this option for creating records.

All items
: The added items and prepopulated items in the Repeater component from the collection for prepopulated items at run time. We recommend selecting this option for creating or updating existing records.

Prepopulated Items
: The items in the Repeater component that were prepopulated from the collection for prepopulated items at run time. This output collection excludes items that the user added or removed. We recommend selecting this option for updating existing records.

Removed Items
: The items the user removed manually from the Repeater component that were prepopulated from the collection for prepopulated items at run time. Only the `UniqueField__Id` is available to reference within the loop. We recommend selecting this option for updating or deleting existing records.

For this example, we select All Items because we're only updating changes to existing records. We recommend processing all items and then using Create Records to create records or update existing records.


Select Collection Variable

* Collection Variable

{!Repeater.}

b. In the loop path, add the Assignment element.

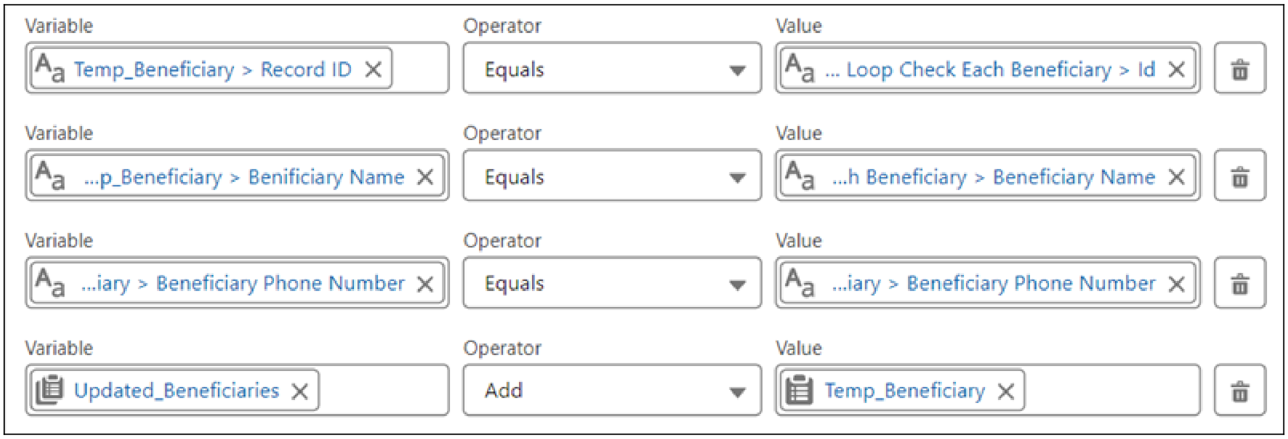
c. In this assignment, assign each repeater output value to a field in a temporary beneficiary record variable.

d. Make sure to assign the ID field to the Record ID field on the temporary record variable.

Without a record ID, a flow fails to update records. Read-only records aren't updated because they can't be modified.

e. After you assign all the fields, add the temporary beneficiary record variable to a beneficiary collection variable.

f. After you assign all the fields, add the temporary beneficiary record variable to a beneficiary collection variable.



Variable	Operator	Value
Temp_Beneficiary > Record ID	Equals	... Loop Check Each Beneficiary > Id
...p_Beneficiary > Benificiary Name	Equals	...h Beneficiary > Beneficiary Name
...iary > Beneficiary Phone Number	Equals	...iary > Beneficiary Phone Number
Updated_Beneficiaries	Add	Temp_Beneficiary

If you process different output collections, such as removed items, then you can add another Loop element and Assignment element to assign the removed records to another collection record variable. After you assign all the fields, add the Delete Records element to delete them from the database.

6. Add the Create Records element.
   a. For How to set record field values, select **From a Record Variable**.
   b. For How Many Records to Create, select **Multiple**.
   c. For Record Collection, select the record collection that contains the records.

      In this example, select the collection of updated beneficiaries.
   d. Enable **Update Existing Records**.
   e. For Fields to use to identify existing records, select **Record ID**.

      In the beneficiary example, the record ID uniquely identifies each beneficiary.
   f. Select what to do if a record fails to create or update.
7. Save and activate the flow.

Sum or Count Items in Collections with the Transform Element

Aggregate data from a source collection to calculate the sum or count of items in that collection, and assign the result to a target data field.

Before you begin, understand the structure of your source and target data, such as whether the data contains multiple levels of collections within other collections. Mapping fields in a collection requires rules to preserve data integrity. See Flow Element: Transform.

Count the number of items in a source collection, or add the field values on each item in the source collection to calculate their sum.

1. Add the Transform element to your flow.
   a. Enter the label, API name, and description.
   b. For Source Data, click the Add Resource button ⊞ and select the flow resource to transform the data.

      Select a resource that references a collection to aggregate.
   c. For Target Data, click the Add Resource button ⊞ and select the data type.
   d. If the target data is a collection, select **Allow multiple values (collection)**.
   e. For a Record or Apex-Defined data type, select the Apex class or object for the target data that the Transform element generates.

      For example, if you specified that the target data is a collection and that the record data type is the Account object, the Transform element generates an account collection. If you didn't specify a collection, the target data is a single account.
2. Map the source collection to the target data field that's a Number data type.
   a. Hover over a source collection and click the Map button ◉.
   b. Next to a target data field that's a Number data type, click the Map button ◉.

      If a target field doesn't have the Map button ◉, you can't map to it. When mapping fields in a collection, the source and target fields must be at the same hierarchical level in their respective resources. See Flow Element: Transform.
3. For Aggregate Type, select **Count** or **Sum**.
4. For Field to Transform, select the source data field on each item in the source collection to calculate the transformed value.

   This field is available only for the sum aggregate type.
5. Save your flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

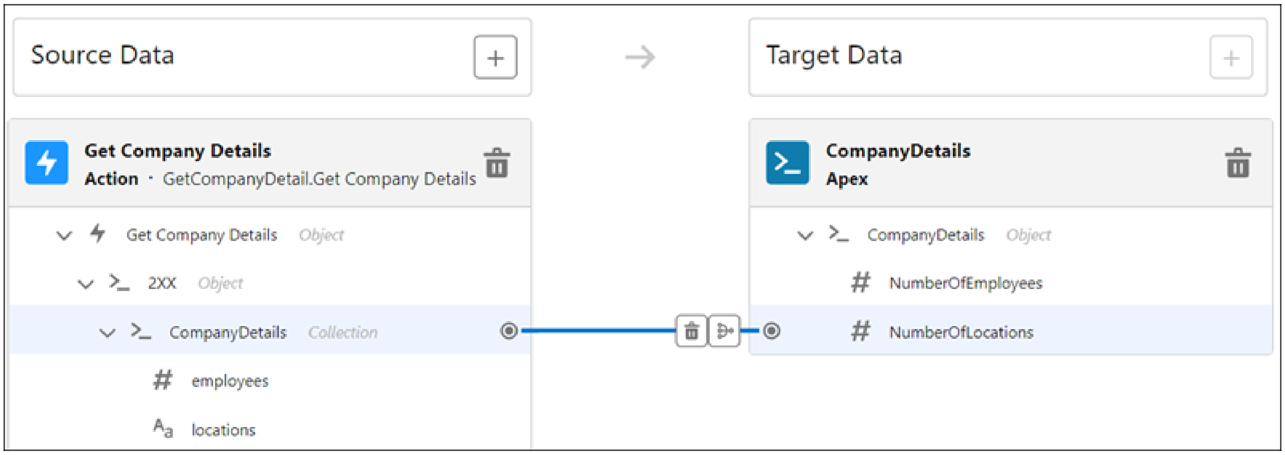
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

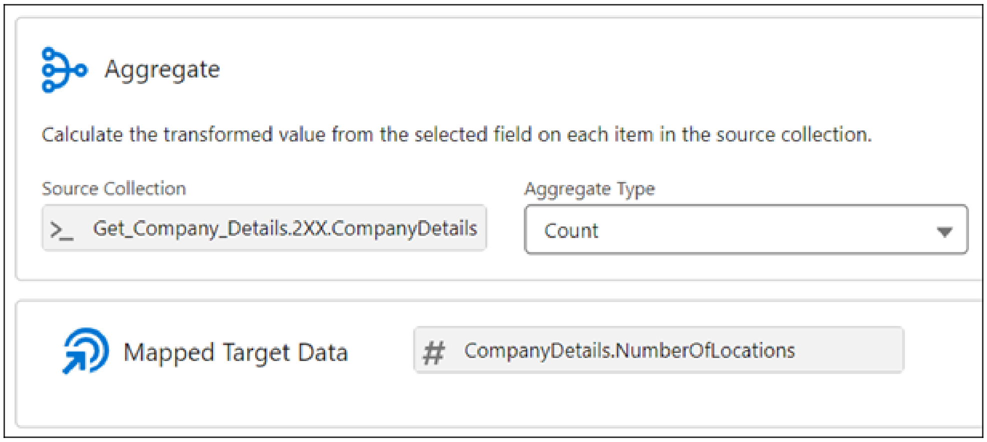
To open, edit, or create a flow in Flow Builder:

- Manage Flow

Example: Let's say you create a flow that retrieves data about company locations from an external system. The external data includes company locations and the number of employees per location. In the flow, you map the CompanyDetails source collection to the NumberOfLocations target data field that stores the transformed value.



You configure the Transform element to count the number of company locations.



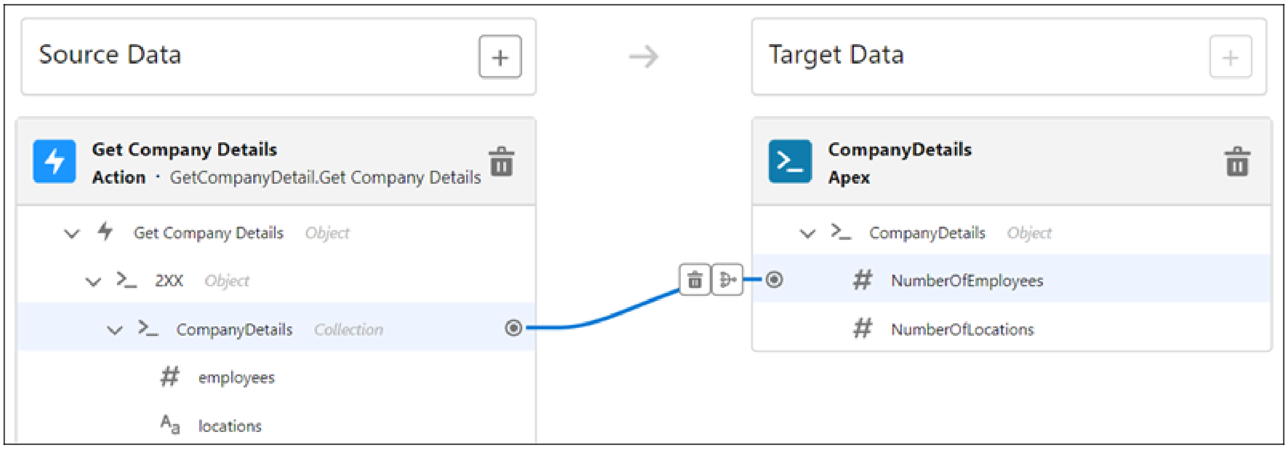
Calculate the transformed value from the selected field on each item in the source collection.

Source Collection: Get_Company_Details.2XX.CompanyDetails

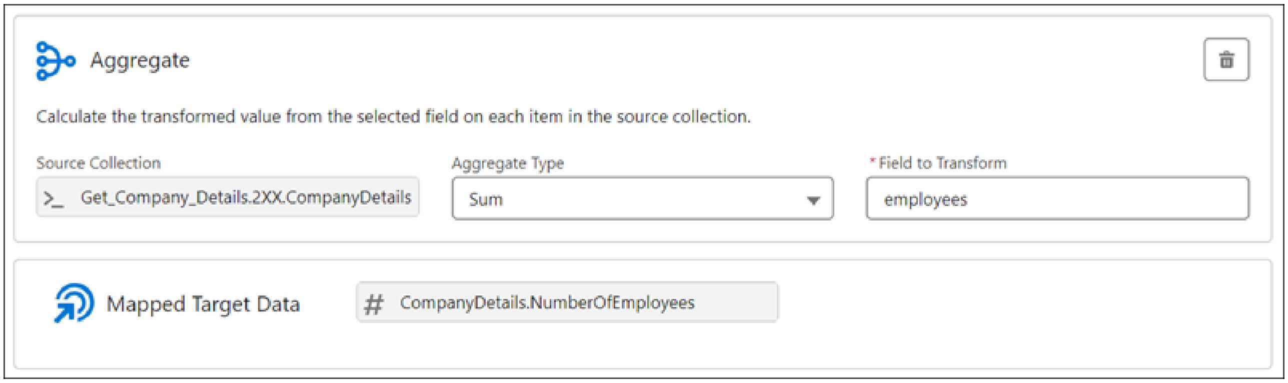
Aggregate Type: Count

Mapped Target Data: CompanyDetails.NumberOfLocations

Next, you map the CompanyDetails source collection to the NumberOfEmployees target data field that stores the transformed value.



You configure the Transform element to calculate the sum of employees for all locations.



Aggregate

Calculate the transformed value from the selected field on each item in the source collection.

Source Collection: Get_Company_Details.2XX.CompanyDetails

Aggregate Type: Sum

* Field to Transform: employees

Mapped Target Data: CompanyDetails.NumberOfEmployees

SEE ALSO:

Transform Data in a Flow

Flow Element: Transform

Integrate with External Systems from a Flow

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: **Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions**

With Get Records elements, you can easily look up your Salesforce data in a flow. But what if you need data that lives outside of Salesforce? To connect your flow to an external database, use platform events, external objects, Lightning components, External Services, or Apex.

Platform Events

Deliver secure and expandable custom notifications within Salesforce or from external sources by using platform events. To publish event messages from your flow, add a Create Records element, where the specified object is the platform event. To subscribe to messages, add a Wait element.

External Objects

Reference data that's stored outside your Salesforce org by using external objects. After you've mapped the external system to an external object in your org, use flow data elements to get, create, or update data in the external system.

Custom Lightning Components

Connect to a database that's behind your firewall without going through the Salesforce server by calling a local action. All local actions appear in Flow Builder as Core Action elements.

External Services

Connect to any external system without writing a line of code. You tell us which endpoint and schema you want to use, and we generate Apex classes for you. The Apex classes appear in Flow Builder as Apex actions. External Services supports OpenAPI 2.0 JSON schema format.

Apex

If you want more control, write your own Apex code to integrate with an external system. To make your Apex code available in the Flow Builder, use either the `@InvocableMethod` annotation or the `Process.Plugin` interface.

Integration Option	Declarative	Server-Side	Client-Side	Synchronous	Asynchronous
Platform events	✓	✓			✓
External objects	✓	✓		✓	
Custom Lightning components			✓	✓	✓
External Services	✓	✓		✓	
Apex		✓		✓	✓

Building Flows with Third Party Connectors

Third-party connectors are software tools that enable different applications or systems to communicate with each other. Third-party connectors serve as a bridge connecting different platforms, enabling interoperability and data exchange. Third-party connectors improve workflow, enhance efficiency, and provide a more connected experience for users.

HTTP Callout

HTTP Callout pulls or sends data between the flow and an external system via Flow Builder without using code. You can set up direct integrations as needed without having to work with a developer or call a middleware tool. After you configure the HTTP callout action in a flow, Flow Builder auto-generates an external service registration, an invocable action, and Apex classes that you can use to create an Apex-defined resource for flows. You can then use the data output of the API request as input within Flow Builder and across Salesforce.

Transform Data in a Flow

Automate how you transform data between Salesforce and an external system using the Transform element. You can create mappings between the source and target data, or set fixed values for the target data.

Generate Flow Choice Options From External Data

If your business process interacts with external data, your users can select from it on flow screens.

Building Flows with Third Party Connectors

Third-party connectors are software tools that enable different applications or systems to communicate with each other. Third-party connectors serve as a bridge connecting different platforms, enabling interoperability and data exchange. Third-party connectors improve workflow, enhance efficiency, and provide a more connected experience for users.

You can create a flow with a third-party connector as a trigger or an action. Each third-party connector has its own triggers and actions. For more information about supported triggers and actions, refer to Third-Party Connectors and view the respective connector reference.

Example: You can use third-party connectors in your flows for no-code connectivity to external systems. For example:

- When a contact is updated in NetSuite, update the related Contact record in Salesforce.
- When an Order record is updated in Salesforce, update the data in QuickBooks Online.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Build a Flow with a Third Party Connector as an Action

TBD

Build a Flow with a Third Party Connector as a Trigger

TBD

Build a Flow with a Third Party Connector as an Action

TBD

TBD

TBD

1. TBD

Example: TBD

TBD

Flow Example: Build a Flow with a Third Party Connector as an Action

TBD

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: **Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions**

USER PERMISSIONS

To create flows:

- Manage Flow

To add third party connectors to flows:

- Manage Integration Connections

Flow Example: Build a Flow with a Third Party Connector as an Action

TBD

TBD

TBD

1. TBD

Example: TBD

TBD

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: **Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions**

USER PERMISSIONS

To create flows:

- Manage Flow

To add third party connectors to flows:

- Manage Integration Connections

Build a Flow with a Third Party Connector as a Trigger

TBD

TBD

TBD

1. TBD

Example: TBD

TBD

Build a Flow with a Third Party Connector as a Trigger

TBD

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: **Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions**

USER PERMISSIONS

To create flows:

- Manage Flow

To add third party connectors to flows:

- Manage Integration Connections

Build a Flow with a Third Party Connector as a Trigger

TBD

TBD

TBD

1. TBD

Example: TBD

TBD

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: **Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions**

USER PERMISSIONS

To create flows:

- Manage Flow

To add third party connectors to flows:

- Manage Integration Connections

HTTP Callout

HTTP Callout pulls or sends data between the flow and an external system via Flow Builder without using code. You can set up direct integrations as needed without having to work with a developer or call a middleware tool. After you configure the HTTP callout action in a flow, Flow Builder auto-generates an external service registration, an invocable action, and Apex classes that you can use to create an Apex-defined resource for flows. You can then use the data output of the API request as input within Flow Builder and across Salesforce.

Admins who have the Customize Applications user permission can use HTTP Callout. You must be familiar with how APIs work and have the API document for the endpoint you're calling.

EDITIONS

Available in:

- Lightning Experience
- **Enterprise**, **Unlimited**, and **Developer** Editions

The HTTP Callout configuration is powered by External Services, which makes the action reusable and invocable in Flow Builder and across your Salesforce org. Before you configure an integration with HTTP Callout, you set up authentication in Setup > Named Credentials so that the external service can connect to the API.

Example: You can use HTTP Callout to connect a flow to a variety of APIs.

- Get address information using a map API—When a user enters an address into a screen flow, a real-time call to the Maps API provider is triggered. The provider sends back details on the address and displays business hours within the screen flow.
- Get weather conditions with a weather service API—Perform a daily callout to a weather service and check the weather for a specific area. If the weather meets certain conditions, notify students that class is indoors or outdoors.

- Get payment authorization information with a payment processing API–Call a payment processor with an invoice number and verify whether a payment has been made and has cleared.
- Add records to an inventory system when product records are added to Salesforce.
- Add leads and contacts to an external marketing automation tool.
- Process a payment transaction through an external payment gateway.
- Update existing order information in an external order management system.

HTTP Callout Considerations

When using HTTP Callout, keep these considerations in mind.

Configuring an HTTP Callout Action

Connect Flow Builder to an HTTP-based API without using code. To establish a connection between the API and Salesforce, you create an external credential and a named credential in Setup. Then, HTTP Callout guides you through describing the functionality of your API and the endpoint that you're connecting to. Based on the information that you enter, Flow Builder generates an external service with a corresponding invocable action that you can use across Salesforce.

Manage HTTP Callout Actions

When you create an HTTP Callout action in a flow, Flow Builder creates an external service object and an invocable action object. Anyone with permissions in your org can reuse the invocable action in Flow Builder and across Salesforce.

SEE ALSO:

Salesforce Help : Named Credentials

External Link: FDA API

External Link: NPPES NPI Registry API

HTTP Callout Considerations

When using HTTP Callout, keep these considerations in mind.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- JSON response with fields in a list that have the same data type is supported. For example, [1, 2, 3, 4] is a list of numbers with the same data type. But ["one", 2, "three", "four"] doesn't have the same data type because 2 is a number, and the other values are strings.
- Enum data type isn't supported. For example, if the API includes a Status field with finite values of Accepted and Rejected, the values don't appear in Flow Builder as a multi-picklist. The Status field is instead inferred as a string data type and the callout response includes one value. See Enums.
- Float and long data types aren't supported. If the API response provided during configuration includes a field with float or long data types, you can set the data type to integer or double.
- Include headers within the named credential when configuring the authentication, which supports global merged fields. Set headers on the URL endpoint for the callout rather than the invocable action.

Limits

- Every time you create an HTTP callout action within Flow Builder, it auto-generates an external service registration. See Callouts and Callbacks: Limits and Usage.

- You can have up to 15 nested levels of fields and objects in a JSON hierarchy.

SEE ALSO:

Salesforce Help: Salesforce Features and Edition Allocations

Salesforce Help: Schema Definition Support

Considerations for the Apex-Defined Data Type

Configuring an HTTP Callout Action

Connect Flow Builder to an HTTP-based API without using code. To establish a connection between the API and Salesforce, you create an external credential and a named credential in Setup. Then, HTTP Callout guides you through describing the functionality of your API and the endpoint that you're connecting to. Based on the information that you enter, Flow Builder generates an external service with a corresponding invocable action that you can use across Salesforce.

Guidelines for Authenticating HTTP Callout Actions

Before you configure an HTTP Callout action within Flow Builder, set up a named credential to authenticate calls to the external system. From Setup, follow these guidelines to create a permission set, auth provider, external credential, and named credential, in that order. Then grant users access to the User Credentials object.

Configure an HTTP Callout Action

HTTP Callout guides you through entering the details about the HTTP web-based service or REST API endpoint that you're connecting to. After you complete the configuration, you invoke the action in a flow. HTTP Callout assumes that you're familiar with the API that you want to call.

Use the HTTP Callout Action: Example Flow

In this example, you use the HTTP Callout action in a flow that manages a pet store's inventory.

Guidelines for Authenticating HTTP Callout Actions

Before you configure an HTTP Callout action within Flow Builder, set up a named credential to authenticate calls to the external system. From Setup, follow these guidelines to create a permission set, auth provider, external credential, and named credential, in that order. Then grant users access to the User Credentials object.

EDITIONS

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Permission Set

Required. Grants users access to make the callout.

If an existing permission set doesn't exist, create one. Then assign the permission set to each user who can make the callout. Alternatively, consider creating a legacy-named credential to save time because it doesn't require a permission set.

Auth. Provider

Required only for the OAuth protocol use cases. Interacts with the identity provider to obtain tokens.

External Credential

Required. Defines the authentication.

Create the external credential from the Named Credential Setup page, on the External Credential tab.

Select the appropriate protocol based on the API's requirement:

- Custom. Select for:
  - Basic authentication (username and password)
  - Key or token
  - No authentication
- OAuth 2.0
- AWS Signature Version 4

Add a Permission Set Mapping to the external credential and select the related permission set.

For APIs that require a key or token in the header:

- Add an Authentication Parameter to the Permission Set Mapping. Enter a unique Name and enter the key in the Value field.
- Add a Custom Header. Enter the exact Name that the API expects. For the Value, point to the name Authentication Parameter. Example: $Credential.namedCredApiName.authParameterName

For APIs that require basic authentication (username and password):

- Add an Authentication Parameter to the Permission Set Mapping to name and store the value of the username.
- To name and store the value of the password, add another Authentication Parameter.
- Add a Custom Header. Enter the Name "Authorization." For the value, point it to the username and password Authentication Parameters. Example: {!'Basic ' & BASE64ENCODE(BLOB($Credential.BasicAuth.Username & ':' & $Credential.BasicAuth.Password))}

Named Credential

Required. Defines the name and URL of the endpoint. The named credential is associated with the HTTP callout action when you create the action in Flow Builder.

The external credential and named credential are separate because APIs often have multiple endpoints that are addressed with the same authentication, for example, calendar.google.com/api and drive.google.com/api.

Enter a Label name that describes the endpoint you're connecting to, enter the base URL in the URL field, and select the External Credential from the previous step.

For the Generate Authorization Header field, leave checked for no authentication and OAuth use cases. Deselect for key or token and basic authentication (username and password) use cases.

For the Allow Formulas in the HTTP Header field, check if the header that contains a formula, which includes basic auth use cases.

Access to User Credentials Object

Required for all use cases except for Open APIs that don't require authentication. The Named Credentials subsystem stores secret tokens or values in the User Credentials object.

Using Permission Sets or Profiles, grant the needed access (Read, Create, Edit, Delete) to the User Credentials Object.

SEE ALSO:

Named Credentials and External Credentials

Configure an HTTP Callout Action

HTTP Callout guides you through entering the details about the HTTP web-based service or REST API endpoint that you're connecting to. After you complete the configuration, you invoke the action in a flow. HTTP Callout assumes that you're familiar with the API that you want to call.

Before you configure an HTTP callout action:

- Create an external credential and a named credential so that the external service can connect to the API. See Guidelines for Authenticating HTTP Callout Actions.
- Prepare a sample API response in JSON format. If a sample doesn't exist in the API's documentation, you can use a third-party API platform to retrieve a response.
- Start with an API that's well documented. If details such as field requirements or definitions are inaccurate or outdated, it can take time to debug.

1. From Setup, in the Quick Find box, enter `Flows`, and select **Flows**.
2. Open an existing flow that you want to use the callout in or create one.
3. Click **+** and select **Action**.
4. Click **+Create HTTP Callout**.
5. Configure the external service that connects Salesforce to the HTTP-based API.
   a. Enter a name for the external service, such as ConnectToMaps. The name must begin with a letter and contain only alphanumeric characters without spaces.
   b. For reference, enter a description of what the external service is connecting to.
   c. Select the named credential that you created for this external service.
   d. Click **Next**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

6. Configure the invocable action that you can use in Flow Builder or across Salesforce.
   a. For Label, enter the action that the callout performs, for example, Connect to Maps.
   b. Select the operation or method required by the API that you're calling.

      Generally, GET brings in external data. The rest of the methods modify data in an external system. The functionality depends on the API.

   c. Enter a description of the API performing the operation, and include a link to the API document so that you can reference it if you change the callout.
7. Add the URL endpoint for the request.

   The base URL shows the Named Credential URL for the external service. If you can't see the entire URL, click the field.

   a. Enter the URL path for the endpoint that starts with a slash (/) and doesn't contain a question mark (?).
   b. If you have path variables, place them in curly brackets, and select the variable's data type. Path variables can contain only alphanumeric characters and underscores.
   c. Enter the variable's description, including usage details.
8. Add query parameter keys if the API you're calling has them. When you use this action in a flow, you input values for the defined keys.
   a. Click **+Add Key**.
   b. Enter the key value and select the data type.
   c. To require a value when the action is invoked in a flow, select **Require**.

d. Enter the query parameter key's description, including usage details.

9. For POST, PUT, PATCH, and DELETE, provide a sample API request body. Salesforce generates a data structure from the sample request.

   a. Click **New**.

   b. Paste a sample JSON request.

   c. Click **Review** and confirm that the Apex data structure is correct.

   d. To make changes, edit the text in Sample JSON Request or select data types from the data structure.

   e. Click **Review**.

   f. Click **Done**.

10. Provide a sample API response body. Salesforce generates a data structure from the sample response.

    a. Click **New**.

    b. Paste a sample JSON response.

    c. Click **Review** and confirm that the Apex data structure is correct.

    d. To make changes, edit the text in Sample JSON Response or select data types from the data structure.

    e. Click **Review**.

    f. Click **Done**.

11. Click **Save**.

    Flow Builder creates the action and external service. The action is available in the Actions window in the flow.

12. For POST, PUT, PATCH, and DELETE, set the body for the flow by creating a resource and selecting the Apex class for the external server's body.

    a. For body, select **New Resource**.

    b. Enter an API name for the new resource.

    c. Enter a description that describes the variable and how it's used.

       The data type and Apex class are set for you. The Apex class for the HTTP callout action uses this naming convention: ExternalServiceName__HTTP Callout Label_IN_body. For example, if the external service name is MyCustomES and the label for the new HTTP Callout is Get Accounts, then the Apex class is MyCustomES_Getx20Accounts_IN_body, where x20 represents a space in the label.

    d. Click **Done**.

    e. For body, select the Apex-defined variable that you created.

    f. Click **Done**.

To transform the data between Salesforce and the external system in the flow, use the Transform element. For POST, PUT, PATCH, and DELETE methods, you then add an Assignment element before the HTTP callout action. In the Assignment element, assign values for each of the fields on the Apex-defined variable. The HTTP callout action references the Apex-defined variable as the input to create the data in the external server.

Flow Builder creates Apex classes automatically from the inferred data structures to store data that's transferred between Salesforce and the external server. The Apex class naming convention for HTTP callout actions that use the GET method is ExternalServiceName__HTTP

Callout Label_OUT_2XX. For example, if the external service name is MyCustomES and the label for the new HTTP Callout is Get Accounts, then the Apex class is MyCustomES_Getx20Accounts_OUT_2XX.

SEE ALSO:

HTTP Callout Considerations

Use the HTTP Callout Action: Example Flow

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

In this example, you use the HTTP Callout action in a flow that manages a pet store's inventory.

Let's say that you want to make managing your pet store's inventory easier by integrating with your supplier's ordering system so that you can place orders in real time. You want to create a flow that reviews your inventory levels and triggers a call to your supplier's API when they're low. When you click **Submit Order**, it prompts a real-time API call that automatically places your order.

Before you configure an HTTP callout action:

- Create an external credential and a named credential so that the external service can connect to the API. See Guidelines for Authenticating HTTP Callout Actions. In this example, the external credential is PetStore and the named credential is Acme Pet Supplies.
- Prepare a sample API response in JSON format. If a sample doesn't exist in the API's documentation, you can use a third-party API platform to retrieve a response. In this example, the sample API response used is:

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2024-10-09T08:49:33.605Z",
  "status": "placed",
  "complete": true
}
```

- Start with an API that's well documented. If details such as field requirements or definitions are inaccurate or outdated, it can take time to debug.

1. Open Flow Builder.
   - From Setup, in the Quick Find box, enter *Flows*, and then select **Flows**.
   - From the Automation Lightning app, select the **Flows** tab.
   - From the **Flows** tab in any Lightning app.
2. Create the flow.
   a. Click **New Flow**.
   b. From Flow Builder, select **Start from Scratch**, and then click **Next**.

c. Select **Screen Flow**, and then click **Create**.

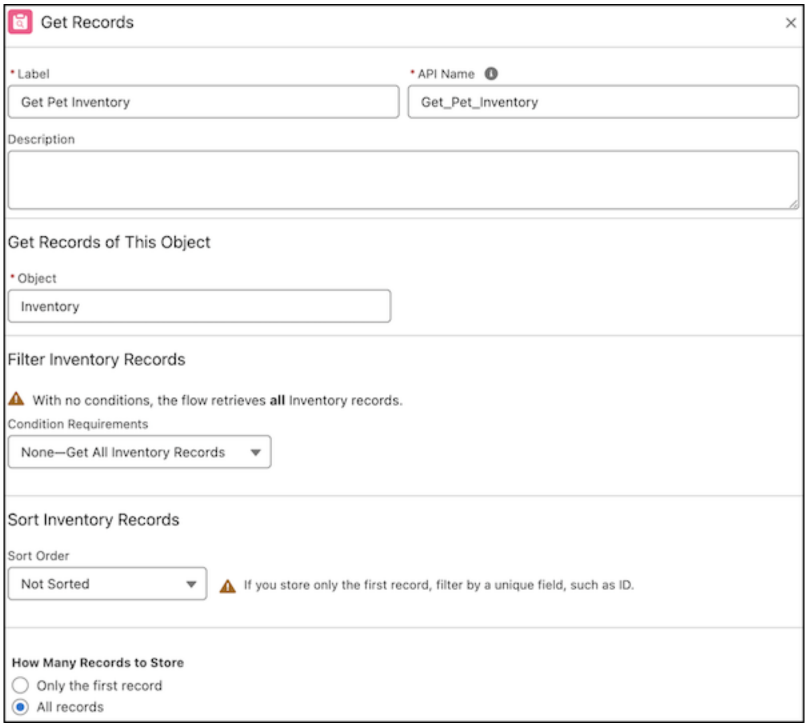
3. Add the Get Records element and configure it.

   a. For Label, enter `Get Pet Inventory`.

   b. For Object, select **Inventory**.

   c. For Condition Requirements, select **None—Get All Inventory Records**.

   d. For How Many Records to Store, select **All Records**.



Get Records

* Label: Get Pet Inventory

* API Name: Get_Pet_Inventory

Description

Get Records of This Object

* Object: Inventory

Filter Inventory Records

⚠ With no conditions, the flow retrieves **all** Inventory records.

Condition Requirements: None—Get All Inventory Records

Sort Inventory Records

Sort Order: Not Sorted

⚠ If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

○ Only the first record

● All records

4. After the Get Pet Inventory element, add a Screen element and configure it.

   a. For Label, enter `Reorder Pet Inventory`.

   b. For API Name, enter `Reorder_Pet_Inventory`.

   c. Under Configure Header, enable **Show Header**.

   d. Under Configure Footer, enable **Show Footer**.

5. Add a Section component to the Reorder Pet Inventory screen and configure it.

   a. Enable **Include Header**.

   b. For Label, enter `Pet Inventory`.

   c. For API Name, enter `Available_Pets`.

   d. Under Configure columns, add the required columns, and set the width.

6. Add a Data Table component as a child of the Pet Inventory section and configure it.

   a. For API Name, enter `petTable`.

   b. For Label, enter `Available Pets`.

   c. Enable **Use Label as the table title**.
   d. For Source Collection under Configure Data Source, enter `{!Get_Pet_Inventory}`.
   e. Enable **Show search bar**.
   f. Under Configure Columns, add required columns.

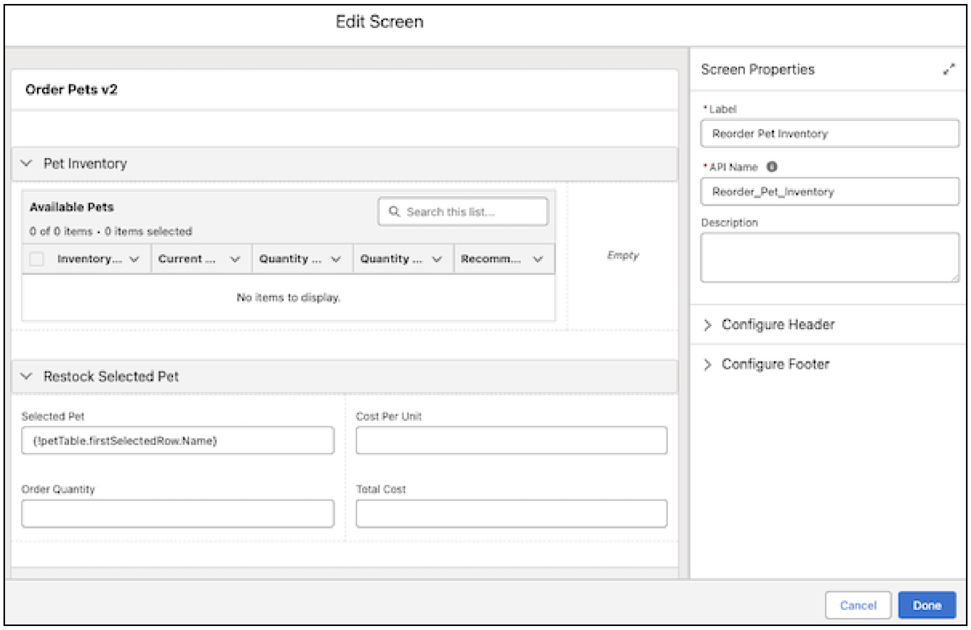
7. Add another Section component as a child of the Reorder Pet Inventory screen and configure it.
   a. Enable **Include Header**.
   b. For Label, enter `Restock Selected Pet`.
   c. For API Name, enter `Reorder_Information`.
   d. Under Configure Columns, add the required columns, and set their width.

8. Add a Text component as a child of the Restock Selected Pet section and enter these values.
   a. For Label, enter `Selected Pet`.
   b. For API Name, enter `Selected_Pet`.
   c. For Default Value, enter `{!petTable.firstSelectedRow.Name}`.

9. Add a Currency component as a child of the Restock Selected Pet section and enter these values.
   a. For Label, enter `Cost Per Unit`.
   b. For API Name, enter `Cost_Per_Unit`.
   c. For Default Value, enter `{!petTable.firstSelectedRow.Current_MSRP__c}`.

10. Add a Number component as a child of the Restock Selected Pet section and enter these values.
   a. For Label, enter `Order Quantity`.
   b. For API Name, enter `Order_Quantity`.
   c. For Decimal Places, enter `0`.

11. Add another Number component as a child of the Restock Selected Pet section and enter these values.
   a. For Label, enter `Total Cost`.
   b. For API Name, enter `Total_Cost`.
   c. For Default Value, enter `{!formulaTotalCost}`.
   d. For Decimal Places, enter `2`.



Edit Screen

Order Pets v2

Pet Inventory

Available Pets

0 of 0 items • 0 items selected

□	Inventory...	Current ...	Quantity ...	Quantity ...	Recomm...
No items to display.					

Empty

Restock Selected Pet

Selected Pet: {!petTable.firstSelectedRow.Name}

Cost Per Unit:

Order Quantity:

Total Cost:

Screen Properties

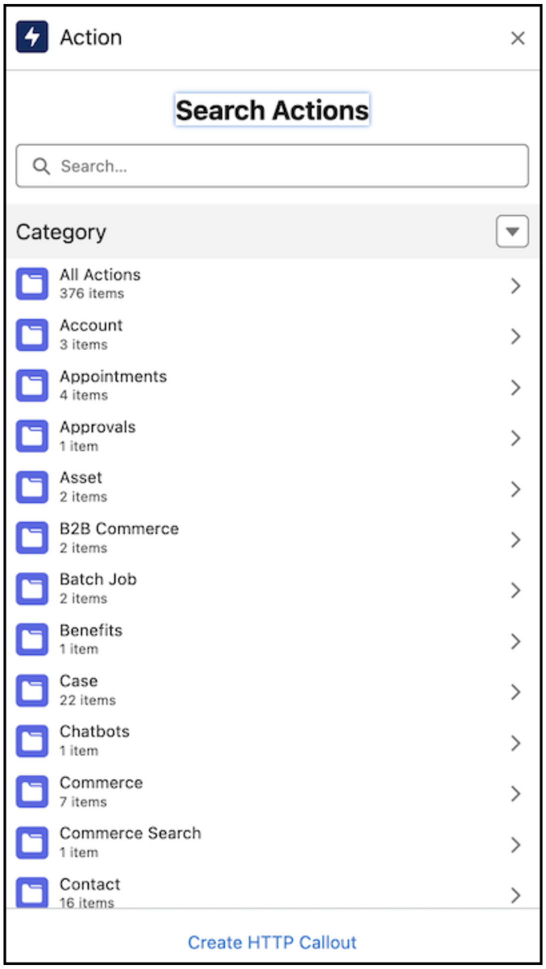
* Label: Reorder Pet Inventory

* API Name: Reorder_Pet_Inventory

Description:

12. After the Screen element, add an Action element and configure it.

a. Click **Create HTTP Callout**.



b. In Create External Service, for Name, enter `OrderPetTest`.

c. For Named Credential, select **Acme Pet Supplies**.

   The URL associated with the credential is autopopulated.

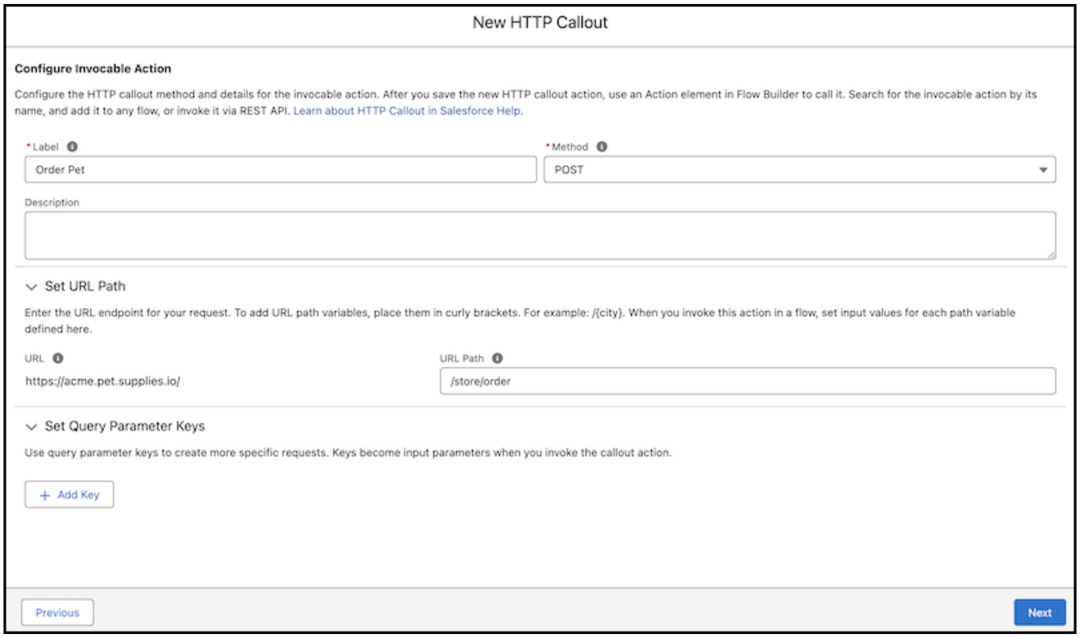
d. Click **Next**.

e. In Configure Invocable Action, for Label, enter `Order Pet`.

f. For Method, select **POST**.

g. For URL Path, enter `/store/order`.

h. Click **Next**.



New HTTP Callout

Configure Invocable Action

Configure the HTTP callout method and details for the invocable action. After you save the new HTTP callout action, use an Action element in Flow Builder to call it. Search for the invocable action by its name, and add it to any flow, or invoke it via REST API. Learn about HTTP Callout in Salesforce Help.

*Label: Order Pet

*Method: POST

Description:

Set URL Path

Enter the URL endpoint for your request. To add URL path variables, place them in curly brackets. For example: /{city}. When you invoke this action in a flow, set input values for each path variable defined here.

URL: https://acme.pet.supplies.io/

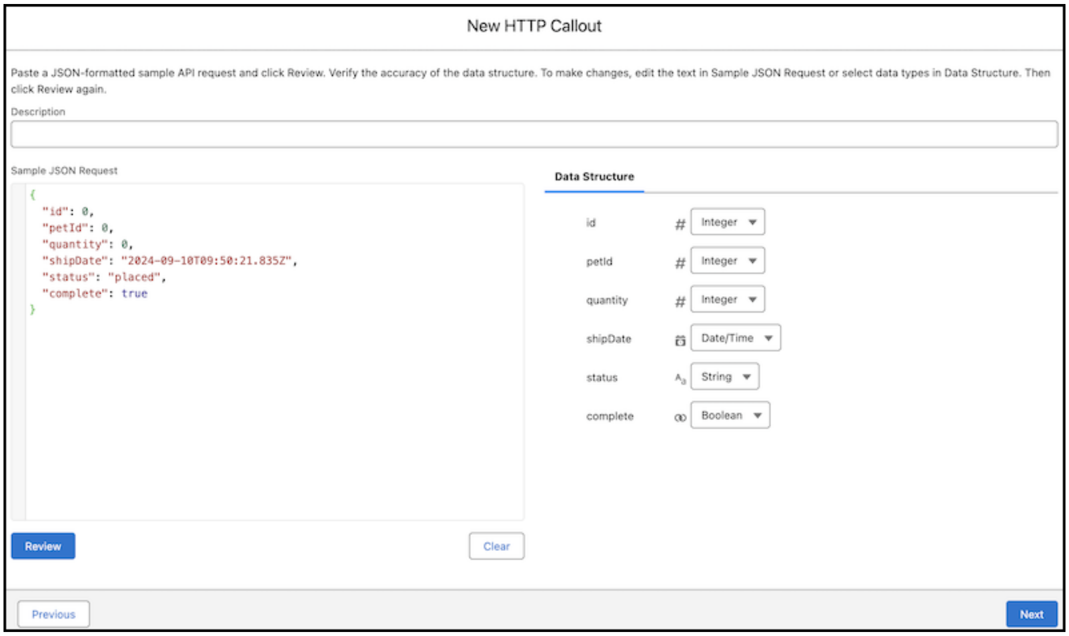
URL Path: /store/order

Set Query Parameter Keys

Use query parameter keys to create more specific requests. Keys become input parameters when you invoke the callout action.

i. For Sample JSON Request, enter the JSON snippet required for this example.

j. To confirm data structure parameters, click **Review**.



New HTTP Callout

Paste a JSON-formatted sample API request and click Review. Verify the accuracy of the data structure. To make changes, edit the text in Sample JSON Request or select data types in Data Structure. Then click Review again.

Description:

Sample JSON Request

```
{
  "id": 0,
  "petId": 0,
  "quantity": 0,
  "shipDate": "2024-09-10T09:50:21.835Z",
  "status": "placed",
  "complete": true
}
```

Data Structure

id	#	Integer
petId	#	Integer
quantity	#	Integer
shipDate		Date/Time
status		String
complete		Boolean

k. In Select Sample Response Method, select **Connect for Schema**, and then click **Next**.

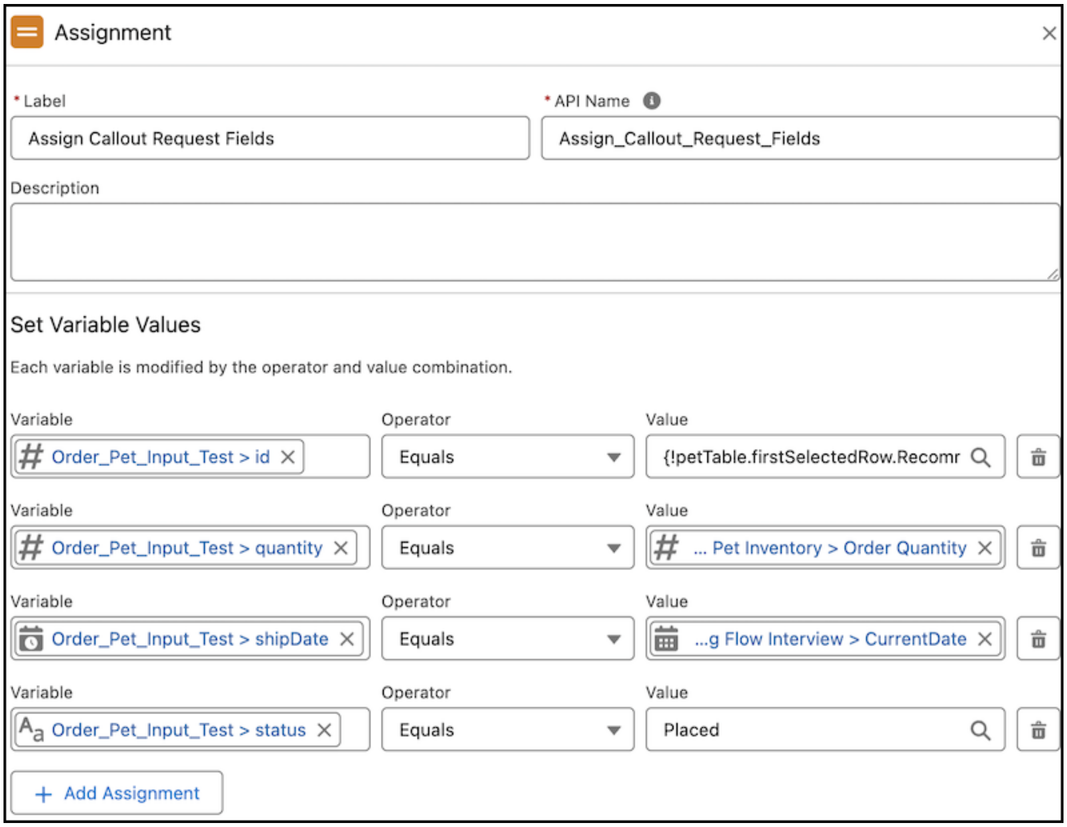
l. Click **Save**.

m. For Label, enter `Callout to Order Pets`.

n. For API Name, enter `Callout_to_Order_Pets`.

o. In Set Body Request, for Value, enter `{!Order_Pet_Input_Test}`.

13. Before the Callout to Order Pets action, add an Assignment element and configure it.
    a. For Label, enter `Assign Callout Request Fields`.
    b. For API Name, enter `Assign_Callout_Request_Fields`.
    c. For Set Variable Values, enter `{!Order_Pet_Input_Test.id}` **Equals**. `{!petTable.firstSelectedRow.Recommended_Reorder_Quantity__c}`.
    d. Click **Add Assignment**, and enter `{!Order_Pet_Input_Test.quantity}` **Equals** `{!Order_Quantity}`.
    e. Click **Add Assignment**, and enter `{!Order_Pet_Input_Test.shipDate}` **Equals** `{!$Flow.CurrentDate}`.
    f. Click **Add Assignment**, and enter `{!Order_Pet_Input_Test.status}` **Equals** `Placed`.



Assignment

* Label: Assign Callout Request Fields

* API Name: Assign_Callout_Request_Fields

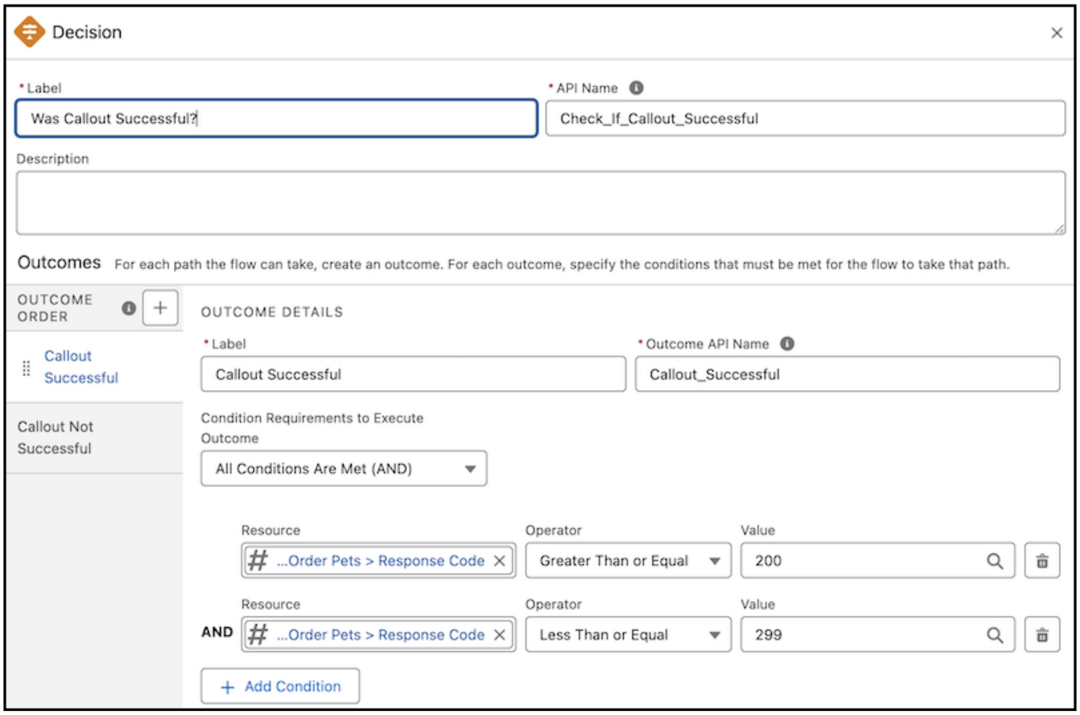
Description

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
Order_Pet_Input_Test > id	Equals	{!petTable.firstSelectedRow.Recomr
Order_Pet_Input_Test > quantity	Equals	... Pet Inventory > Order Quantity
Order_Pet_Input_Test > shipDate	Equals	...g Flow Interview > CurrentDate
Order_Pet_Input_Test > status	Equals	Placed

14. After the Callout to Order Pets action, add a Decision element and configure it.
    a. For Label, enter `Was Callout Successful?`.
    b. For API Name, enter `Was_Callout_Successful`.
    c. For the New Outcome label, enter `Callout Successful`.
    d. For Condition Requirements to Execute Outcome, select **All Conditions Are Met (AND)**.
    e. For Resources, enter `{!Callout_to_Order_Pets.responseCode}`**Greater Than or Equal** `200`.
    f. Click **Add Resources**, enter `{!Callout_to_Order_Pets.responseCode}` **Lesser Than or Equal** `200`.
    g. Click **Default Outcome**.
    h. For Label, delete the text, and then enter `Callout Not Successful`.



Decision

*Label: Was Callout Successful?

*API Name: Check_If_Callout_Successful

Description:

Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

OUTCOME ORDER

- Callout Successful
- Callout Not Successful

OUTCOME DETAILS

*Label: Callout Successful

*Outcome API Name: Callout_Successful

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

	Resource	Operator	Value
	...Order Pets > Response Code	Greater Than or Equal	200
AND	...Order Pets > Response Code	Less Than or Equal	299

15. On the Callout Not Successful outcome path, add a Screen element and configure it.

  a. For Label, enter `Error Screen`.

  b. For API Name, enter `Error_Screen`.

  c. Under Configure Header, enable **Show Header**.

  d. Under Configure Footer, enable **Show Footer**.

16. On the Callout Successful outcome path, add an Update Records element and configure it.

  a. For Label, enter `Update Inventory Quantity on Hand`.

  b. For API Name, enter `Update_Inventory_Quantity_on_Hand`.

  c. For How to Find Records to Update and Set Their Values, select **Specify conditions to identify records, and set fields individually**.

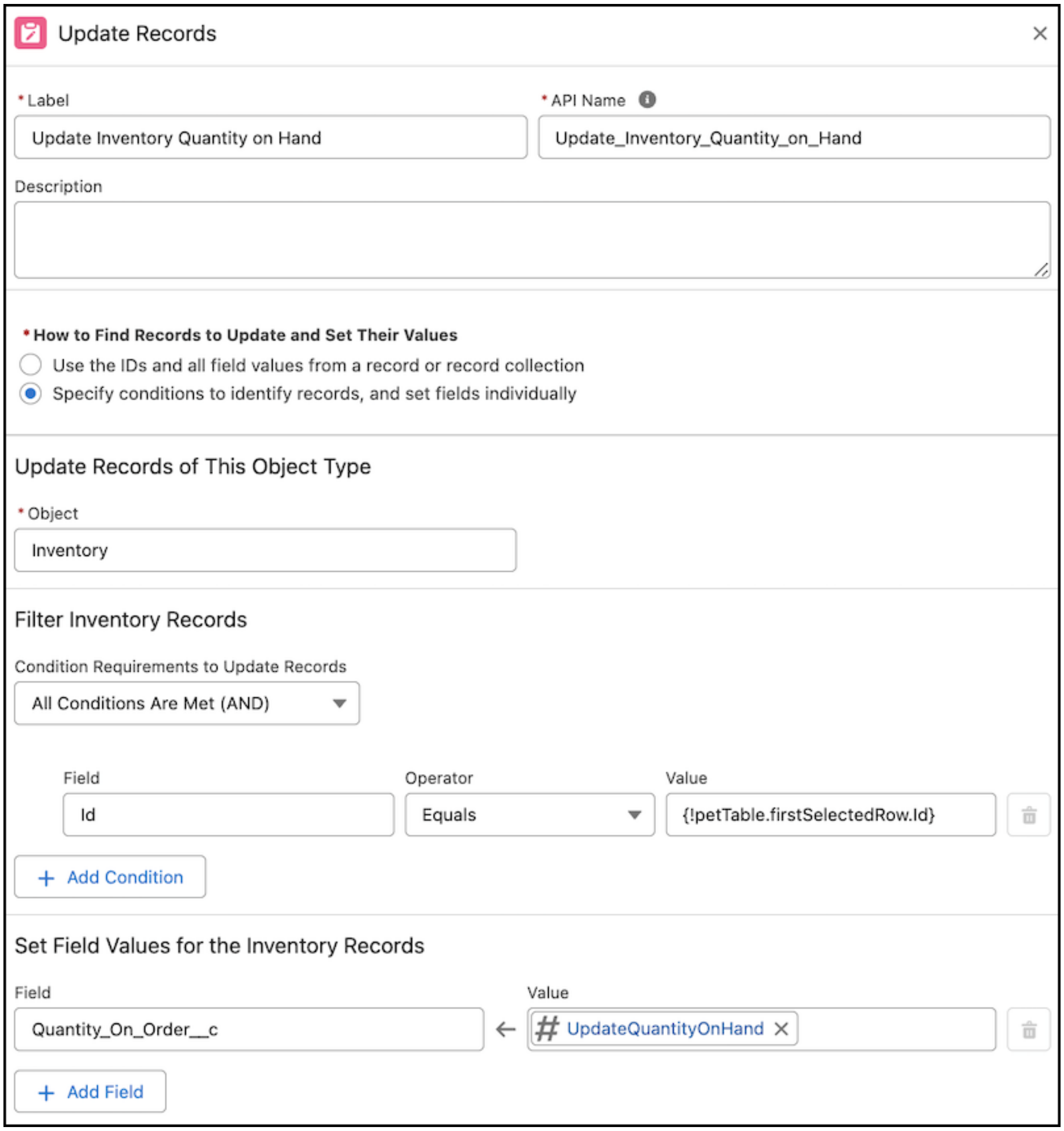
  d. For Object, select **Inventory**.

  e. For Condition Requirements to Update Records in Filter Inventory Records, select **All Conditions Are Met (AND)**.

  f. For Field, Operator, and Value, select **Record ID Equals** `{!petTable.firstSelectedRow.Id}`.

  g. In Set Field Values for the Inventory Records, for Field, select **Quantity on Order**.

  h. In Set Field Values for the Inventory Records, for Value, enter `{!UpdateQuantityOnHand}`.



Update Records

* Label: Update Inventory Quantity on Hand

* API Name: Update_Inventory_Quantity_on_Hand

Description:

* How to Find Records to Update and Set Their Values

○ Use the IDs and all field values from a record or record collection

● Specify conditions to identify records, and set fields individually

Update Records of This Object Type

* Object: Inventory

Filter Inventory Records

Condition Requirements to Update Records: All Conditions Are Met (AND)

Field	Operator	Value
Id	Equals	{!petTable.firstSelectedRow.Id}

Set Field Values for the Inventory Records

Field		Value
Quantity_On_Order__c	←	UpdateQuantityOnHand

17. On the Callout Successful outcome path, next to the Update Inventory Quantity on Hand element, add a Screen element and configure it.

a. For Label, enter `Order Placed`.

b. For API Name, enter `Order_Placed`.

c. Under Configure Footer, enable **Show Footer**.

d. For Navigation, select **Use a custom label**.

e. For Next or Finish Button Label, enter `OK`.

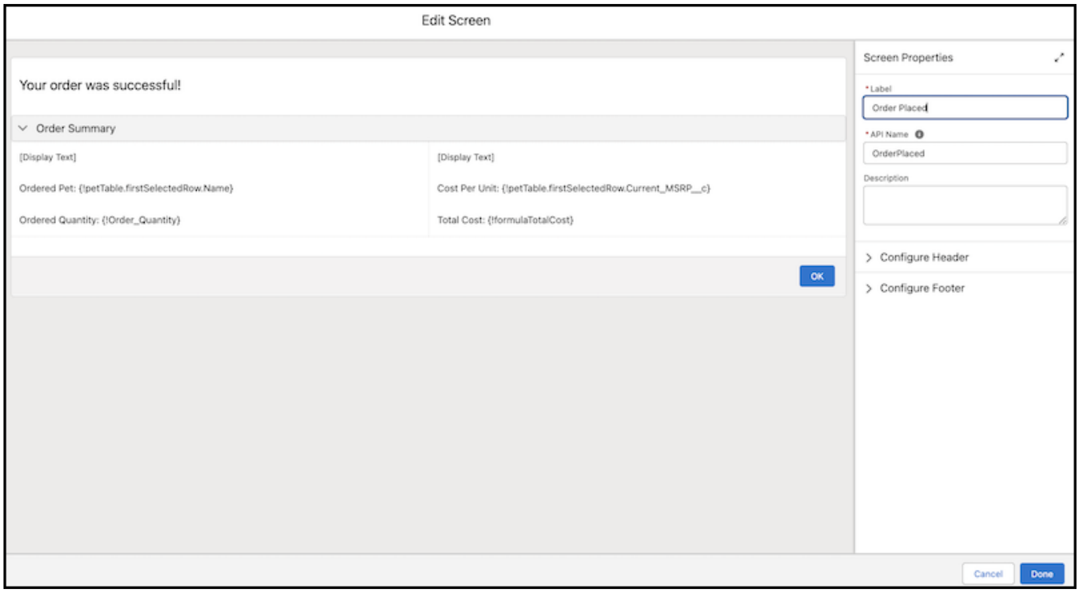
f. Hide the Previous and Pause buttons.

g. Add a Display component to the Screen element.

h. For the API Name of the Display component, enter `SuccessDisplayHeader`.

i. For Description of the Display component, enter `Your order was successful!`.

- **j.** Add a Section component to the Screen element.
- **k.** For Label of the Section component, enter `Order Summary`.
- **l.** For the API Name of the Section component, enter `OrderSummary`.
- **m.** Set the Column width based on the requirement.
- **n.** Add required Display components as children of the Section, and configure them based on the requirement.



Your order was successful!

Order Summary

[Display Text]	[Display Text]
Ordered Pet: {!petTable.firstSelectedRow.Name}	Cost Per Unit: {!petTable.firstSelectedRow.Current_MSRP__c}
Ordered Quantity: {!Order_Quantity}	Total Cost: {!formulaTotalCost}

Screen Properties

*Label: Order Placed

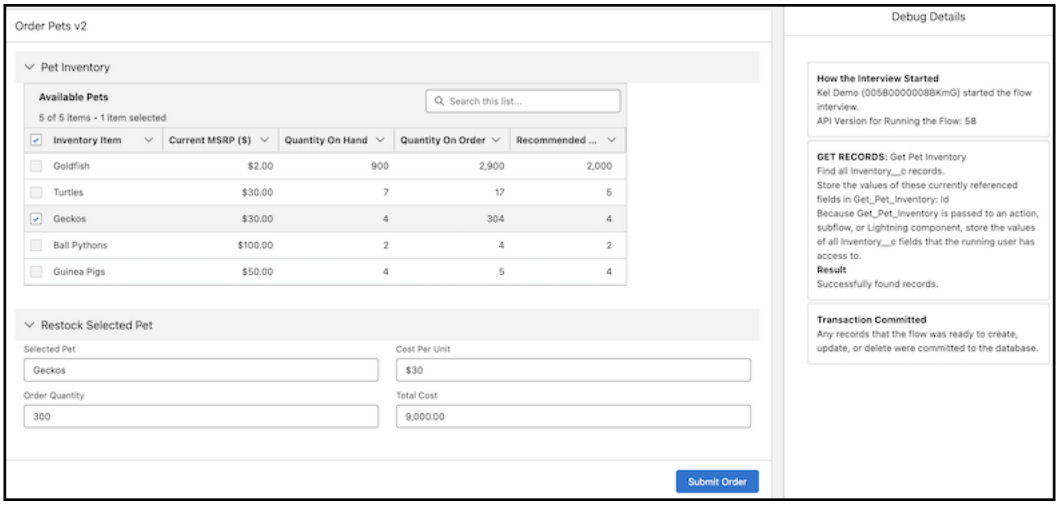
*API Name: OrderPlaced

Description:

18. Save the flow, and then click **Debug** to troubleshoot the flow for any errors.

19. Click **Run**.

20. Select an inventory item that you want to order.


Order Pets v2

Pet Inventory

Available Pets

5 of 5 items • 1 item selected

☑	Inventory Item	Current MSRP ($)	Quantity On Hand	Quantity On Order	Recommended ...
□	Goldfish	$2.00	900	2,900	2,000
□	Turtles	$30.00	7	17	5
☑	Geckos	$30.00	4	304	4
□	Ball Pythons	$100.00	2	4	2
□	Guinea Pigs	$50.00	4	5	4

Restock Selected Pet

Selected Pet: Geckos

Cost Per Unit: $30

Order Quantity: 300

Total Cost: 9,000.00

Debug Details

How the Interview Started
Kel Demo (005B0000008BKmG) started the flow interview.
API Version for Running the Flow: 58

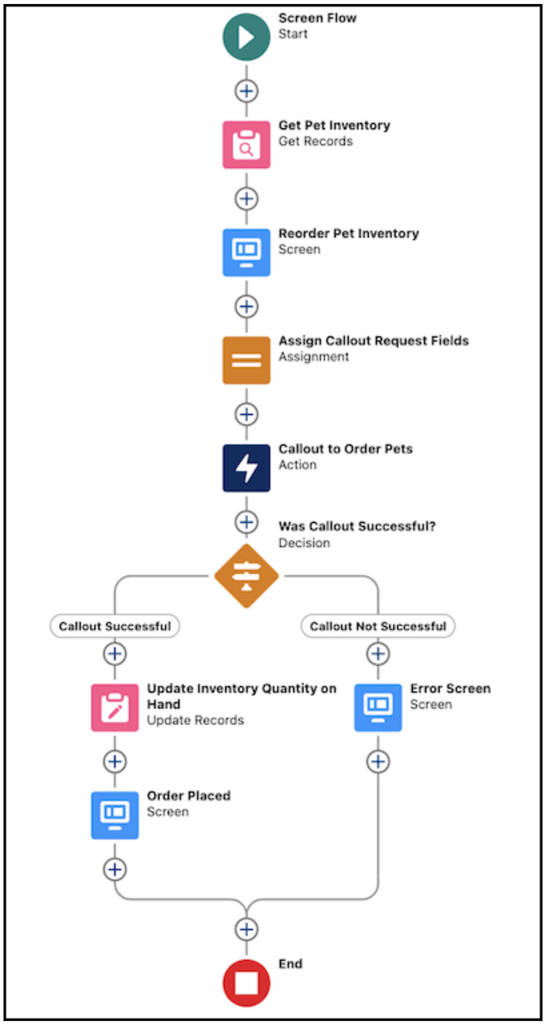
GET RECORDS: Get Pet Inventory
Find all Inventory__c records.
Store the values of these currently referenced fields in Get_Pet_Inventory: Id
Because Get_Pet_Inventory is passed to an action, subflow, or Lightning component, store the values of all Inventory__c fields that the running user has access to.
Result
Successfully found records.

Transaction Committed
Any records that the flow was ready to create, update, or delete were committed to the database.

21. Enter an order quantity, and then click **Submit Order**.
The order is placed successfully. If the flow fails, troubleshoot the flow errors.

22. (Optional) To restart the flow by using the same or different values for the input variables, click **Change Inputs or Run Again**.

You can't convert the debug run to a test in the screen flow.



SEE ALSO:

HTTP Callout Considerations

Manage HTTP Callout Actions

When you create an HTTP Callout action in a flow, Flow Builder creates an external service object and an invocable action object. Anyone with permissions in your org can reuse the invocable action in Flow Builder and across Salesforce.

To update flow-specific input values of an HTTP Callout action, edit the respective Action element from the Flow Builder canvas. For example, update a query parameter value from the respective Action element.

Update an HTTP Callout Action Configuration

To update the configuration of an HTTP callout action, edit it from the External Services page in Setup. For example, update a path URL, keys, or a JSON sample. After you create an HTTP callout action from Flow Builder, you can reuse it in another flow or across Salesforce via Apex, Einstein Bots, or quick actions.

Change the Named Credential for an HTTP Callout Action

To select a different named credential for an HTTP Callout action after the action is created, update the action from the External Services page in Setup.

Delete an HTTP Callout Action

To delete an HTTP Callout action, you delete the external service registration record. If an external service registration and HTTP Callout action are referenced in any flows, you can't delete them.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

Update an HTTP Callout Action Configuration

To update the configuration of an HTTP callout action, edit it from the External Services page in Setup. For example, update a path URL, keys, or a JSON sample. After you create an HTTP callout action from Flow Builder, you can reuse it in another flow or across Salesforce via Apex, Einstein Bots, or quick actions.

1. From Setup, in the Quick Find box, enter `External Services`, and select **External Services**.
2. Locate the external service that you created with HTTP Callout in Flow Builder.
3. Click the external service name.
4. Click the dropdown menu for the operation, and select **Edit HTTP Callout Action**.
5. Update the action and the sample response.
6. Save your changes.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

SEE ALSO:

Configure an HTTP Callout Action

Salesforce Help: Edit an External Service

Change the Named Credential for an HTTP Callout Action

To select a different named credential for an HTTP Callout action after the action is created, update the action from the External Services page in Setup.

1. From Setup, in the Quick Find box, enter `External Services`, and select **External Services**.
2. Locate the external service that you created with HTTP Callout in Flow Builder.
3. Click the arrow in the service's Actions column, and select **Edit**.
4. Select a Named Credential from the **Select a Named Credential** dropdown list.
5. Click **Save & Next**.
6. From Select operations, confirm that the invocable action's operation is selected, and click **Next**.
7. From External Service actions, click **Done**.

SEE ALSO:

Salesforce Help: Edit an External Service

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

Delete an HTTP Callout Action

To delete an HTTP Callout action, you delete the external service registration record. If an external service registration and HTTP Callout action are referenced in any flows, you can't delete them.

1. From Setup, in the Quick Find box, enter `External Services`, and select **External Services**.
2. Locate the external service that you created with HTTP Callout in Flow Builder.
3. Click the arrow in the service's Actions column, and select **Delete**.
4. Click **OK**.

SEE ALSO:

Salesforce Help: Edit an External Service

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create an external credential, a named credential, and an HTTP callout action:

- Customize Applications

Transform Data in a Flow

Automate how you transform data between Salesforce and an external system using the Transform element. You can create mappings between the source and target data, or set fixed values for the target data.

Watch this demo (English only) of transforming data in Flow Builder.

Watch a video

For another viewing option, see Transform Your Data with Flow Builder (English Only).

Before you begin, understand the structure of your source and target data, such as whether the data contains multiple levels of collections within other collections. Mapping fields in a collection requires rules to preserve data integrity. See Flow Element: Transform.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

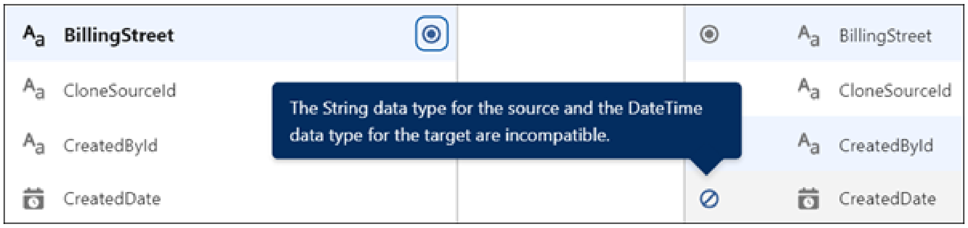
1. Add the Transform element to your flow.
   a. Enter the label, API name, and description.
   b. For Source Data, click the Add Resource button ⊞ and select the flow resource to transform the data.
   c. For Target Data, click the Add Resource button ⊞ and select the data type.
   d. If the target data is a collection, select **Allow multiple values (collection)**.
   e. If the data type is a record or Apex-defined, select the Apex class or object for the target data that the Transform element generates.

      For example, if you specified that the target data is a collection and that the record data type is the Account object, the Transform element generates an account collection. If you didn't specify a collection, the target data is a single account.

2. Map the source and target data.
   a. Hover over a source data field and click the Map button ◉.
   b. Next to a target data field, click the Map button ◉.

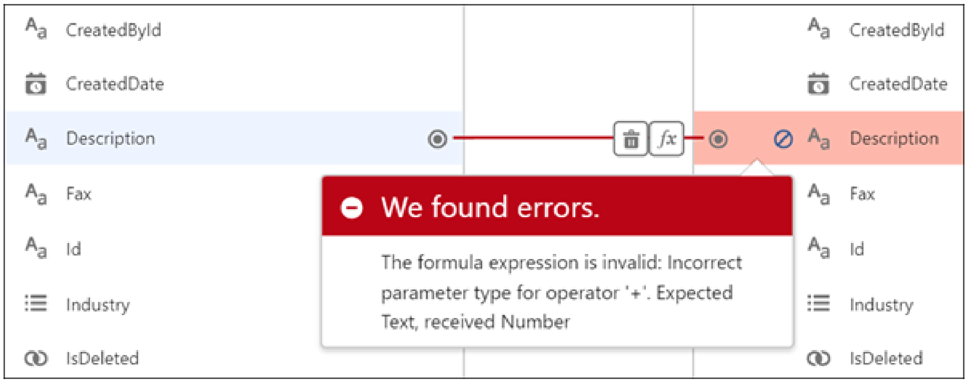
      If a target field doesn't have the Map button ◉, you can't map to it.

   c. To view a mapping tip next to a target field or collection that's unavailable for mapping, hover over the target field or collection, and then hover over the error icon.



The String data type for the source and the DateTime data type for the target are incompatible.

   d. To view a mapping tip of a misconfiguration error, hover over the error icon ⊘ shown next to a target data field or collection.



We found errors.

The formula expression is invalid: Incorrect parameter type for operator '+'. Expected Text, received Number

The Transform element adds a dashed line to indicate mappings within a collapsed object or collection. The Transform element adds dotted lines to identify the collections that contain the mapped fields so that you can easily view the collections in both resource data structures. When mapping fields in a collection, the source and target fields must be at the same hierarchical level in their respective resources. See Flow Element: Transform.

3. To transform data with a formula, click the mapped field name, and then click fx then **Formula**.
4. To delete a mapping, click the field name, and then click the Delete button 🗑.

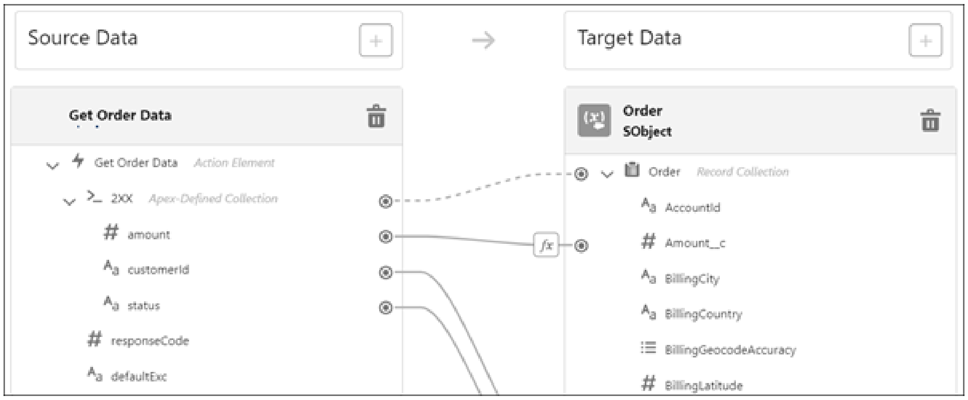
After you finish transforming data, you can save the target data to Salesforce or to an external system.

To save the target data to Salesforce, add the Update Records element, and then configure it to reference the resource with the same name as the Transform element. For example, if the API name of the Transform element is Return_Order, select Return_Order for Record or Record Collection in the Update Records element.

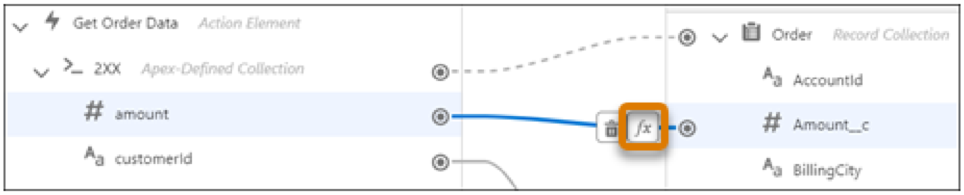
To save the target data to an external system, create an HTTP callout action that uses a method like POST.

Example: A sales agent runs a screen flow to return two orders for store credit. The agent's customer has an account in Salesforce. The customer's order data is stored in a system outside Salesforce. To update the Order record in Salesforce, the flow gets the latest order data from the external system and transforms the data, so the flow can save the changes in Salesforce.

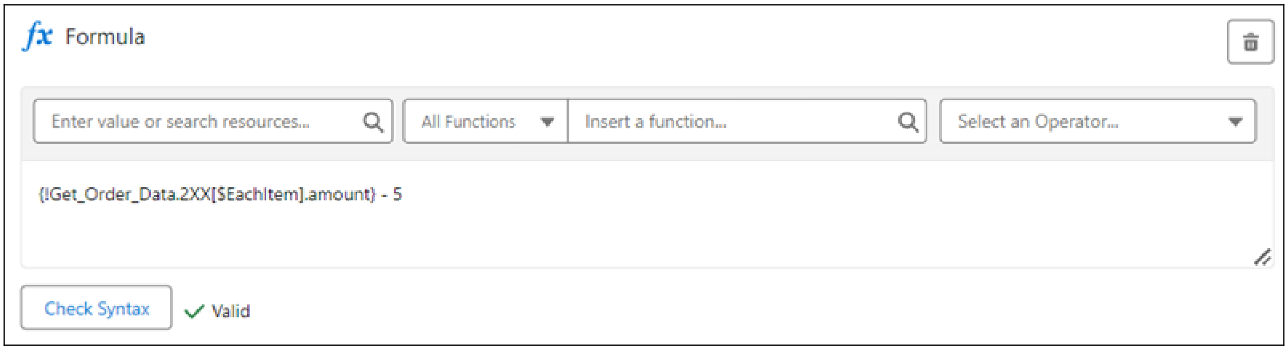
In the Transform element, the amount, customerId, and status source data fields are mapped to the target data fields.



The mapping between the source data's amount field and the target data's Amount__c field uses a formula to transform the data.



The formula subtracts a fixed amount from the source data's amount field for a restocking fee. The result of the formula is the transformed value for the target data's Amount__c field. The formula shows the merge field syntax, where [$EachItem] represents each item in a collection. In this example, the merge field references a collection of 2XX, and each 2XX item has an amount field. For each amount field, the formula subtracts 5.



```
{!Get_Order_Data.2XX[$EachItem].amount} - 5
```

Next in the flow, the Update Records element saves the changes to the database. And finally, the Action element makes an HTTP callout to update the order status in the external system.

SEE ALSO:

Flow Element: Transform

HTTP Callout

Sum or Count Items in Collections with the Transform Element

Generate Flow Choice Options From External Data

If your business process interacts with external data, your users can select from it on flow screens.

You can allow users to select external data.

1. Create a collection choice set and reference an Apex-defined collection from an external service, Apex action, or another screen component.
2. Add your collection choice set to the appropriate choice component, such as a picklist.

Example: For example, a car dealership stores car inventory data outside of Salesforce. They have an external service to access the inventory, and now their users can select which cars to view.

At runtime, the picklist options are generated based on whatever data was populated in the referenced Apex-defined collection.

EDITIONS

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow
-

Options for Sending Emails from Flows

There are several ways you can use a flow to send an email. Each option offers unique benefits, from simplicity and speed to customization and flexibility. The best option to choose depends on the specific requirements of your communication strategy.

When deciding which email option in your flow is right for your communication strategy, consider these questions:

- Is your content static or dynamic? Static content can't change and supports only certain merge fields. Dynamic content can change based on logic.
- Do you want to include attachments? Only email alerts can send attachments to a recipient list.
- Do you require merge fields? Some options accept only minimal merge fields: recipient (the person receiving the email), sender (the person sending the email), and organization (your Salesforce org information).
- Do you want to send a marketing email to an audience segment using a template and pre-built campaign flow? If you have Marketing Cloud Growth, Salesforce Pro Suite, or Salesforce Starter editions, see Examples of Campaign Use Cases in Marketing Cloud Growth and Learn About Salesforce Starter and Pro Suite.
- How many emails do you plan to send daily? Send Email actions and Email Alerts have different daily limits. See Daily Allocations for Email Alerts on page 1003 and the Usage section of Flow Core Action: Send Email on page 517.

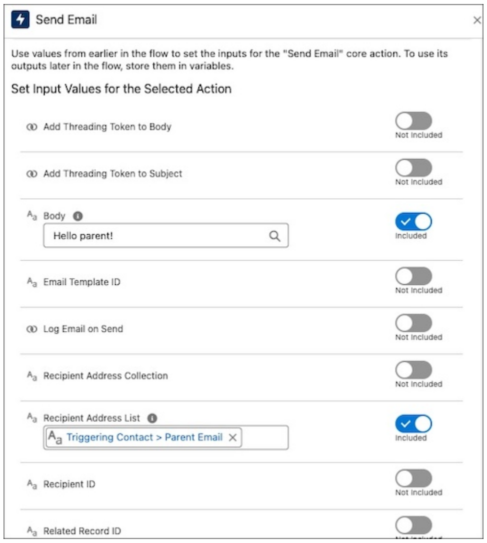
Goal	Content	Recipients	Email Template	Use	Best For	How
Send an email to a fixed group of recipients. Email content is static, includes an attachment, and only has merge fields for recipient, sender, and organization.	Static	Static	Yes	Email Alert on page 330	Because you provide only the record ID of the recipient, use an Email Alert element for a quick flow configuration. Ideal for static content, attachments, and limited merge fields. Email Alerts can have only these merge fields: recipient, sender, and organization. If you already have an email alert configured in Workflow Rules or Process Builder, you can use the email alert when you migrate to Flow Builder.	First, build the email template and email alert outside of the flow. Add an Action element to the flow and select the email alert you created.
Send an email to a group of recipients that changes based on logic. Email content is static, includes an attachment, and has merge fields only for recipient, sender, and organization.	Static	Dynamic	Yes	Send Email Action on page 517 (Email Template)	Use when you want to change recipients based on logic, but you're using static content and limited merge fields. The Send Email action with an email template can have only these merge fields: recipient, sender, and organization. You can't reuse email recipients,	Build the email template outside of the flow. Add an Action element, and select Send Email. For the email template, select a resource that contains the ID of the email template to use. Configure the rest of the Send Email Action.

Goal	Content	Recipients	Email Template	Use	Best For	How
					but the flow does optionally log the email to the recipient's record and another related record.	
Send an email to a group of recipients that changes based on logic. Email content also changes based on logic.	Dynamic	Dynamic	No	Send Email on page 517 (Manual)	Use when you want to change the recipients and the email content based on logic and don't need to reuse the content or recipients. This approach can log the email to the recipient's record and another related record. Perfect for highly personalized communications, such as tailored promotional offers or updates.	Build the email and configure the recipients within the flow. Add an Action element, select Send Email, and configure the action. To use rich text, add Text Template resources for the subject and body and select them in the Send Email action. To use plain text, enter the static subject and body in the corresponding parameters in the Send Email action.

Send Email Action Example: Use a Secondary Email Address

When a new student contact record is created, you want to send an email to the student’s parent. The parent’s email address is stored in a secondary custom email field called Parent Email on the contact record. Create a record-triggered flow that triggers when a contact is createdIn the Send Email action, for Recipient Address List, select theTriggering Contact and then the Parent Email field.

Send Email Action Configuration for Secondary Email Address



Send Email

Use values from earlier in the flow to set the inputs for the "Send Email" core action. To use its outputs later in the flow, store them in variables.

Set Input Values for the Selected Action

Add Threading Token to Body	Not Included
Add Threading Token to Subject	Not Included
Body: Hello parent!	Included
Email Template ID	Not Included
Log Email on Send	Not Included
Recipient Address Collection	Not Included
Recipient Address List: Triggering Contact > Parent Email	Included
Recipient ID	Not Included
Related Record ID	Not Included

Flow Example: Send and Log an Email to a New Contact

Use the Send Email action to send an email to a new contact and log it on the contact and related account records' Activity Timelines. You can save time by reusing email content with an existing Lightning or Classic email template. In this example, the email template has merge fields. Use the Recipient ID and Related Record ID to populate them. Text Template resources are different from email templates and aren't used in this example.

Before you get started, create an email template if you don't already have one.

1. Create a record-triggered flow.
   a. From Setup, in the Quick Find box, enter `Flows`, and then select **Flows**.
   b. Click **New Flow**.
   c. Select **Start From Scratch**, and then click **Next**.
   d. Select **Record-Triggered Flow**, and click **Create**.
2. Configure the Start element.
   a. For Object, select **Contact**.
   b. For Trigger the Flow When, select **A record is created**.
   c. Use the default values for the other fields, and click **Done**.
3. Add a Get Records element.
   a. Click ⊕.
   b. In the search bar, enter `Get Records`, and then click **Get Records**.
   c. For Label, enter `Get Email Template` and use the default API name Get_Email_Template.
   d. For Object, enter `Email Template` and select **Email Template**.
   e. In the Filter Email Template Records section, for Field, select **Name** (Email Template Name). For Operator, select **Equals**. For Value, enter the name of the existing email template to use, for example `New Contact Email.Template`.
   f. Use the default values for the other fields, and click **Done**.
4. Add a Send Email action element after the Get Records element.
   a. Click ⊕.
   b. In the search bar, enter `Send Email`, and then click **Send Email**.
   c. For Label, enter `Send New Contact Email` and use the default API name Send_New_Contact_Email.
   d. Include Email Template ID. For the Email Template ID value, select **Email Template from Get_Email_Template**, and then select **Id** (Email Template ID).
   e. Include Log Email on Send. For the Log Email on Send value, select **{!$GlobalConstant.True}**.
   f. Include Recipient ID. For the Recipient ID, select **{!$Record}**, and then select **Id** (Contact ID).
   g. Include Related Record ID. For the Related Record ID, select **{!$Record}**, and then select **AccountId** (Account ID).
   h. Use the default values for the other fields, and click **Done**.

   Important: If the Sender Type is OrgWideEmailAddress, ensure that the user running the flow has the proper profile configurations required by the specific org-wide email address being used. Proceeding without the proper configuration results in an error.

EDITIONS

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

5. Save and name the flow.
6. To make sure that the flow works as expected, debug and test it using different scenarios.
7. Activate the flow.

SEE ALSO:

Debug a Flow in Flow Builder
Flow Core Action: Send Email
Flow Resource: Global Variables
Email Alert Actions
Email Templates
Email Templates in Lightning Experience
Email Templates in Salesforce Classic
Email Template Builder

Show Users Progress Through a Flow with Stages

Keep users informed about which stage they're in or how far they've progressed in a flow. For example, show where in a purchasing flow the user is with breadcrumbs or a progress indicator.

First, define all the possible stages in your flow with stage resources. When you configure a stage, you set the stage's label, order, and whether it's active by default. Then, throughout the flow identify which of the stages are relevant to the flow user by setting flow global variables.

- The `$Flow.ActiveStages` global variable identifies all the stages that are relevant to the flow's current path.
- The `$Flow.CurrentStage` global variable identifies which stage the flow is at. Make sure that this stage is included in `$Flow.ActiveStages`.

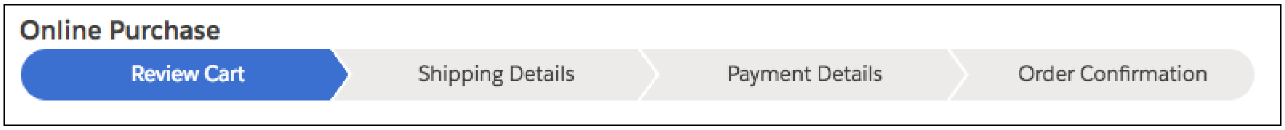
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: The Online Purchase flow includes stages for users to review their cart, enter shipping details, enter payment details, and confirm their order. The stages display at run time using a custom screen component.

At this point, `$Flow.ActiveStages` contains the Review Cart, Shipping Details, Payment Details, and Order Confirmation stages, and `$Flow.CurrentStage` is set to Review Cart.



Online Purchase

Plan the Stages in Your Flow

Before you start adding stages to your flow, plan out all the possible stages for your flow. If your flow includes decisions, you can configure different stages for different branches of your flow.

Define the Stages in Your Flow

After you've identified the stages for each branch of your flow, configure the stages.

Identify the Relevant Stages in Your Flow

Throughout your flow, identify which stages are relevant to the user by assigning values to the stage global variables.

Represent Your Flow's Stages Visually

The standard flow runtime doesn't represent a flow's stages. However, you can use a custom component to visually represent the stages.

SEE ALSO:

Flow Stage Considerations

Plan the Stages in Your Flow

Before you start adding stages to your flow, plan out all the possible stages for your flow. If your flow includes decisions, you can configure different stages for different branches of your flow.

1. List all the possible stages for your flow.
2. Determine in which order the stages occur.
3. Identify which stages are active by default.

   This step is optional, but when you identify the default active stages, you don't have to populate `$Flow.ActiveStages` and `$Flow.CurrentStage` with an Assignment element at the beginning of your flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: Your flow has five sections: Review Cart, Shipping Details, Billing Details, Payment Details, and Order Confirmation. The corresponding stages are in the same order.

If the user's billing details are the same as the shipping details, the flow skips the Billing Details section. The other sections are required for every permutation of the flow. So configure Review Cart, Shipping Details, Payment Details, and Order Confirmation to be active by default.

Stage	Order	Active by Default
Review Cart	1	Selected
Shipping Details	2	Selected
Billing Details	3	Not selected
Payment Details	4	Selected
Order Confirmation	5	Selected

SEE ALSO:

Flow Stage Considerations

Define the Stages in Your Flow

After you've identified the stages for each branch of your flow, configure the stages.

1. On the Resources tab, double-click **Stage**.
2. Enter the stage's label and order, and specify whether it's active by default.

SEE ALSO:

Flow Resource: Stage

Flow Stage Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Identify the Relevant Stages in Your Flow

Throughout your flow, identify which stages are relevant to the user by assigning values to the stage global variables.

The `$Flow.ActiveStages` global variable identifies the stages that are relevant to the flow's current branch. `$Flow.CurrentStage` identifies which stage the flow is in. To update which stages are referenced in the global variables, use an Assignment or Subflow element.

- To add stages to `$Flow.ActiveStages`, use an Assignment element with one of these operators.

Operator	Description
add	Adds stages to the end of `$Flow.ActiveStages`.
add at start	Adds stages to the beginning of `$Flow.ActiveStages`.

- To add a stage to `$Flow.ActiveStages` in between two other stages, define it as a default active stage in another flow. Then use a Subflow element to call the second flow.
- To remove stages from `$Flow.ActiveStages`, use an Assignment element with one of these operators.

Operator	Description
remove after first	Removes all stages after the first instance of a specified stage.
remove all	Removes all instances of the specified stages.
remove before first	Removes all stages before the first instance of a specified stage.
remove first	Removes the first instance of a specified stage.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Operator	Description
remove position	Removes the stage at a specified position.

- To change what's selected as `$Flow.CurrentStage`, use an Assignment element with the **equals** operator. Make sure that the selected stage is included in `$Flow.ActiveStages`.
- To count the number of active stages and assign that number to a variable, use an Assignment element with the **equals count** operator.

To reference a stage in another flow, enter the fully qualified stage name: *`flowName:stageName`* or *`namespace.flowName:stageName`*. At run time, the assignment works only if a Subflow element calls the stage's flow.

SEE ALSO:

Flow Resource: $Flow Global Variables

Flow Element: Assignment

Flow Operators in Assignment Elements

Flow Stage Considerations

Represent Your Flow's Stages Visually

The standard flow runtime doesn't represent a flow's stages. However, you can use a custom component to visually represent the stages.

To visually represent the stages, you can add a custom Aura component or custom Lightning web component to your flow's screens or add a `lightning:flow` component to your custom Aura component.

- Screen component—When you map a stage to a screen component attribute, the flow passes the stage's label into the attribute.
- `lightning:flow` component—The `onstatuschange` attribute in the standard `lightning:flow` component returns the names and labels for the flow's active stages and current stage.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Lightning Aura Components Developer Guide : Display Flow Stages with an Aura Component

Lightning Aura Components Developer Guide : Display Flow Stages By Adding a Progress Indicator to a Flow Screen

Get User Input with Screen Flow Components

Use screen flow components that accept choice resources to enable users to select choices from a list. Configure flow screen components to react to changes in other components on the same screen. Translate flow screen components.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Using Choice Resources with Flow Screen Components

A key part of configuring flow screen components that display choices is selecting the choices to display in that field. The choices appear as radio buttons, checkboxes, or picklist options. Use at least one of these resources: choices, record choice sets, or picklist choice sets.

Choose a Lookup Option for a Flow Screen

Salesforce provides three ways for users to search for and select records on flow screens. Generally, use the Lookup component when you want users to search an unfiltered list of records. Use the Choice Lookup component when you want users to search a filtered list of records or static choices. Use a lookup record field when you want users to have the option to create a record from the lookup field. This table describes the differences between the three options.

Make Your Screen Flows Reactive

Reactivity enables supported flow screen components and formulas to react to changes on the same screen in real time. With reactivity, you can build screens that feel like single-page applications and reduce the number of screens that users have to navigate to complete tasks. Reactivity is supported with API version 59.0 and later.

Translate Flow Screen Components

To use multilingual labels for most flow screen components, use Translation Workbench. Flow screen components in beta and some generally available flow screen components don't support this functionality. For those generally available flow screen components, use the `$Label` global variable to specify custom labels. You must create the custom labels before referencing them in a flow.

Flow Example: Calculate a Discount In Real Time

Use reactive components to enable a sales rep to calculate and apply a product discount on one screen.

Flow Example: Retrieve Opportunities for an Account that a User Selects

Create a screen flow that enables a user to search for and select an Account record, and then click a button on the same screen to launch an autolaunched flow that retrieves all the Opportunity records associated with the Account record. Display the Opportunity records in a table on the same screen.

Flow Example: Create a Contact for Each Beneficiary on a Policy

Add a Repeater component to a screen, and then add child components to represent a beneficiary on a policy. Loop over the output Repeater component to create a collection of Contact records. Then use the collection to create a Contact record for each beneficiary.

Using Choice Resources with Flow Screen Components

A key part of configuring flow screen components that display choices is selecting the choices to display in that field. The choices appear as radio buttons, checkboxes, or picklist options. Use at least one of these resources: choices, record choice sets, or picklist choice sets.

Record choice sets and picklist choice sets are easier to configure and don't require as much maintenance as choices. We recommend using a choice resource only when you can't use the other two.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

If you want the user to select...	Use this resource
Something that can't be generated from a record, picklist field, or multi-select picklist field	Choice
From a collection of existing records	Collection Choice Set
From a set of values that correspond to an existing picklist or multi-select picklist field	Picklist Choice Set
From a set of filtered records	Record Choice Set

SEE ALSO:

- Flow Resource: Record Choice Set
- Flow Resource: Picklist Choice Set
- Flow Resource: Choice

Choose a Lookup Option for a Flow Screen

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Salesforce provides three ways for users to search for and select records on flow screens. Generally, use the Lookup component when you want users to search an unfiltered list of records. Use the Choice Lookup component when you want users to search a filtered list of records or static choices. Use a lookup record field when you want users to have the option to create a record from the lookup field. This table describes the differences between the three options.

	Lookup Record Field on Flow Screen	Choice Lookup Screen Flow Component	Lookup Screen Flow Component
Search an unfiltered list of records	Supported	Not supported	Supported
Search a filtered list of records	Not supported	Supported	Not supported
Search a list of static choices	Not supported	Supported	Not supported
Select multiple records	Not supported	Supported	Supported
Show all auto-complete results during search	Supported	Not supported	Supported
Use in Lightning Web Runtime (LWR) site	Not supported	Supported	Supported
Require users to select an option	Supported The field must be required on the object.	Supported	Supported

	Lookup Record Field on Flow Screen	Choice Lookup Screen Flow Component	Lookup Screen Flow Component
Create a record to populate the lookup field	Supported	Not supported	Not supported
Access records in user context	Supported	Supported	Supported
Access records in system context	Not supported	Supported	Not supported
Format of stored output value	Record ID of the record resource	Choice value such as Record ID	Record ID or text collection of record IDs

SEE ALSO:

Flow Screen Input Component: Choice Lookup

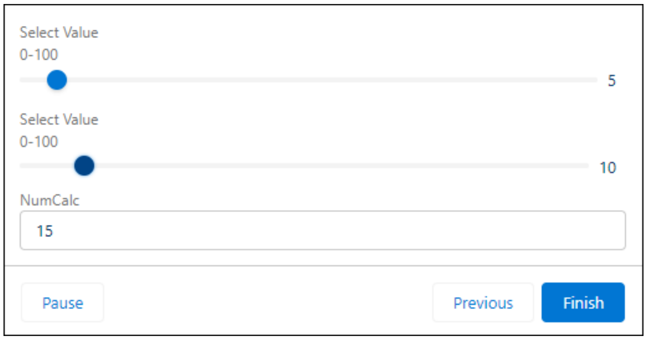
Make Your Screen Flows Reactive

Reactivity enables supported flow screen components and formulas to react to changes on the same screen in real time. With reactivity, you can build screens that feel like single-page applications and reduce the number of screens that users have to navigate to complete tasks. Reactivity is supported with API version 59.0 and later.

Available in: both Salesforce Classic and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Before you create a reactive interaction on a screen, decide which screen component is the source of the interaction and which component reacts to changes in the source. For example, consider a screen with three components:

- Two Slider components that enable users to select values
- A Number component that displays the sum of the values that the user selects in the Slider components



In this scenario, the Slider components are the sources of the reactive interaction and the Number component reacts to changes in the sources. Note that all three components in the reactive interaction produce values of the type number. For reactive interactions to work, the source and reactive values must be of the same type.

To build a reactive screen, it's easiest to use a standard component. If you use custom flow components, ensure that you review the reactivity examples in the Lightning Web Components Developer Guide.

1. Create a screen flow, and add a Screen element to it.

2. In the Screen element, add and configure the component that is the source for the reactive interaction.

   For example, add and configure a Data Table component that displays a list of names for the user to choose from.

3. Add and configure the component that reacts to changes in the source component.

   For example, add a Name component and set the First Name field to **DataTableAPIName.firstSelectedRow.FirstName**. The Name component displays the First Name value of the first row that the user selects in the Data Table.

4. Save and run the flow.

While adding reactivity to your screens, consider these behaviors:

- Manual outputs of components don't support reactivity. If you manually set a component output, that variable doesn't change on the same screen when referenced in other components.
- Help text and labels don't react to changes in other components. This consideration doesn't apply to labels in custom Lightning Web Component that are configured to respond to events in other components.
- Data types must match when you're mapping an output to another component's input to support reactivity.
- If validation rules exist for custom components, reactive changes don't trigger validation.
- The global variable $Flow is reactive. All other global variables such as Custom Labels, Custom Settings, $Organization, $Profile, are not reactive.
- When mapping a `DateTime` field to `Time`, the value is converted to GMT and stays converted when navigating between screens. If mapped to a `DateTime` field, the locale is preserved. For example, if the time value is 8:00 AM in your locale, the converted GMT time could be several hours off your time locale (such as 4:00 PM). Refer to A Note About Date/Time and Time Zones for information about Converting Between Date/Time and Text and Date/Time in time zones: Using Date, Date/Time, and Time Values in Formulas

Reactive Screen Flow Components

A subset of the screen flow components in Salesforce support reactivity.

Reactive Screen Flow Formula Operators

A subset of the formula operators supported in screen flows support reactivity.

Recommendations for Building Reactive Screens

When you build screen flows with reactive components and formulas, keep these recommendations in mind.

Reactive Screen Flow Components

A subset of the screen flow components in Salesforce support reactivity.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions. Reactivity isn't supported in Classic runtime for flows.

Standard Screen Flow Components that Support Reactivity

You can configure these components to react to changes in other components on the same screen in real time.

Supported Components	Level of Reactivity
Address	Full
Checkbox	Full
Choice Lookup	Full
Currency	Full
Data Table	Full

Supported Components	Level of Reactivity
Date	Full
Date & Time	Full
Dependent Picklist	The Dependent Picklist component doesn’t react to changes but can push changes to other components.
Display Text	If you reference the output of another component, and the output is a record variable of type Currency, the value appears differently in the Display Text component than in other parts of the flow. For example, the flow renders currency values without a currency symbol. To avoid this issue, wrap the reference in a formula. For instance, if a Data Table contains records that include an Amount field of type Currency, you can include a reference to the Amount field by wrapping the reference in a Currency formula. Additionally, if you reference a Choice resource from a previous screen in a Display Text component, the component shows the Choice Label instead of the Choice Value. If you include a reference to another screen component within the body of the Display Text component, the text is subject to the same character limit as reactive formulas. Formatting such as applying bold adds characters that count toward the limit. If the text exceeds the character limit, the Display Text component isn't reactive. When you debug a flow that includes a Display Text component that reacts to changes on the same screen, Flow Builder doesn't update the Debug Details pane when the text changes. If you reference a non-Text collection in the Display Text component, you can't also reference the output of another component on the same screen. For example, if you reference a collection of Date values and the output of a Text component on the same screen, the flow returns an error.
Email	Full
Long Text Area	If you reference the output of another component, and the output is a record variable of type Currency, the value appears differently in the Display Text component than in other parts of the flow. For example, the flow renders currency values without a currency symbol. To avoid this issue, wrap the reference in a formula. For instance, if a Data Table contains records that include an Amount field of type Currency, you can include a reference to the Amount field by wrapping the reference in a Currency formula. If you include a reference to another screen component within the body of the Long Text Area component, the text is subject to the same character limit as reactive formulas. Formatting such as applying bold adds characters that count toward the limit. If the text exceeds the character limit, the Long Text Area component isn't reactive. If you reference a non-Text collection in the Long Text Area component, you can't also reference the output of another component on the same screen. For example, if you reference a collection of Date values and the output of a Text component on the same screen, the flow returns an error.
Lookup	Full
Multi-Select Picklist	Full

Supported Components	Level of Reactivity
Name	The `Available Options and salutationOptions` input parameter doesn't support reactivity.
Number	Full
Password	Full
Phone	Validation isn't triggered when the pattern or value input is changed.
Picklist	Full
Radio Buttons	Full
Repeater	In a Repeater component, child components can reference the output of other child components. However, a child component can't reference the output of child component in a different Repeater component. Choice components that reference a collection choice set resource in the Choice field aren't reactive inside Repeater components. References to Repeater components aren't supported in formulas.
Slider	Setting reactivity for sliders can set the values outside the allowable range.
Text	Full
Toggle	Full
URL	Validation isn't triggered when the pattern or value input is changed.

Lightning Web Component Reactivity

Reactivity is supported in Lightning web components. However, to configure a Lightning web component to trigger reactivity in a different component, fire attribute change events following recommended best practices. To configure a Lightning web component to react to changes in other components, update the component's state when the component's `@api` parameter changes.

Flow Components that Don't Support Reactivity

These components can't be configured to react to changes in other components on the same screen in real time:

- Display Image
- File Upload
- Aura components

Flow Resources that Don't Support Reactivity

These flow resources can't be configured to react to changes in other components on the same screen in real time:

- Variable that you set to the output of a component
- Record choice set

Flow Fields that Don’t Support Reactivity

Record fields (Dynamic Forms for Flow) on screens can’t be configured to react to changes in other components on the same screen in real time.

Reactive Screen Flow Formula Operators

A subset of the formula operators supported in screen flows support reactivity.

When a formula with these operators references components that support reactivity on the same screen, the flow recalculates the formula in real time when the values of the referenced components change.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Supported Formula Operators and Functions	Type
+	Math Operator
-	Math Operator
*	Math Operator
/	Math Operator
^	Math Operator
=	Logical Operator
<>	Logical Operator
<	Logical Operator
>	Logical Operator
<=	Logical Operator
>=	Logical Operator
& and + (Concatenate)	Text Operator
ADDMONTHS	Date and Time Function
DATE	Date and Time Function
DATEVALUE	Date and Time Function
DATETIMEVALUE	Date and Time Function
DAY	Date and Time Function
MONTH	Date and Time Function
NOW	Date and Time Function
TODAY	Date and Time Function
WEEKDAY	Date and Time Function
YEAR	Date and Time Function
AND	Logical Function

Supported Formula Operators and Functions	Type
BLANKVALUE	Logical Function
CASE	Logical Function
IF	Logical Function
ISBLANK	Logical Function
ISNULL	Logical Function
ISNUMBER	Logical Function
NOT	Logical Function
NULLVALUE	Logical Function
OR	Logical Function
ABS	Math Function
CEILING	Math Function
EXP	Math Function
FLOOR	Math Function
LN	Math Function
LOG	Math Function
MAX	Math Function
MCEILING	Math Function
MFLOOR	Math Function
MIN	Math Function
MOD	Math Function
ROUND	Math Function
SQRT	Math Function
BEGINS	Text Function
CONTAINS	Text Function
FIND	Text Function
LEFT	Text Function
LEN	Text Function
LOWER	Text Function
MID	Text Function
RIGHT	Text Function

Supported Formula Operators and Functions	Type
SUBSTITUTE	Text Function
TEXT	Text Function
TRIM	Text Function
UPPER	Text Function
VALUE	Text Function
INCLUDES	Advanced Function

Considerations

- Reactivity isn't supported on cross-object formulas, which are formulas that span two related objects and reference merge fields on those objects. For example, consider an Account selected in a Data Table. The account Name, for example `myContactDataTable.FirstSelectedRow.Account.Name`, isn't reactive and doesn't appear in any components referenced on the same screen.
- If a reactive formula relies on a field with an initial value of `null`, the flow doesn't calculate the formula until the user supplies a value. To ensure that your reactive formulas function as expected, consider null values in your configuration. If needed, use the `BLANKVALUE` formula operator to set null values to zero: `BLANKVALUE({!resource},0)`.
- Reactive formulas are limited to 3,900 characters. When you save a formula, Salesforce adds characters to ensure that the formula functions properly. Consequently, your formula can exceed the character limit even if the number of characters you supplied is below the limit. If you exceed the character limit, reactivity isn't supported.

Recommendations for Building Reactive Screens

When you build screen flows with reactive components and formulas, keep these recommendations in mind.

- Map a component's input directly to another component's output. Don't use merge field references in your input such as `'Hello there, {!Screen_Input_1}!'`. Use a formula resource instead.
- Don't use components that actively poll or query data without connecting it to a change event on the screen. If you use reactivity to query data, ensure that you consider user permissions and limit its results to meet performance expectations.
- Check and plan for null or empty values in your reactive formulas. Null or empty values that don't meet conditional visibility requirements can't work with reactivity.
- Check for self-referencing items as these can cause infinite recursion issues such as in places where a field using a formula that appends text to itself.
- Plan for or address any flow scenarios where the formula input that you're evaluating is blank.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Translate Flow Screen Components

To use multilingual labels for most flow screen components, use Translation Workbench. Flow screen components in beta and some generally available flow screen components don't support this functionality. For those generally available flow screen components, use the `$Label` global variable to specify custom labels. You must create the custom labels before referencing them in a flow.

Flow Translation Best Practices

When you use Translation Workbench to translate flows, note these best practices.

Use Multilingual Labels in Data Table Column Headers

Add your translated column headers to the Data Table component using custom labels.

SEE ALSO:

Salesforce Help: Considerations for Translating Flows

Salesforce Help: Translation Workbench

Flow Translation Best Practices

When you use Translation Workbench to translate flows, note these best practices.

- Keep your labels as short as possible. The translated label can't exceed 1,000 characters (or 255 characters for definition name and version name). If you have a long label for a display text field, consider breaking it up into multiple fields.
- When updating a primary label, check whether it has translations, and update as needed.
- Avoid text templates when translating an email body or other formatted block text.
- Avoid using logic that references translated values.

SEE ALSO:

Salesforce Help: Considerations for Translating Flows

Translate Flow Screen Components

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Use Multilingual Labels in Data Table Column Headers

Add your translated column headers to the Data Table component using custom labels.

Before you begin, create and translate your custom labels. From Setup, enter `Custom Labels` in the Quick Find box, and select **Custom Labels**.

1. In a Data Table component, click **Configure Columns**.
2. For Source Field, select the field to use for the column.
3. Select **Custom column label**.

   A new Label field appears.
4. In the new Label field, enter the resource that contains the custom label using the `{!$Label.customLabelName}` expression.

   The resource menu isn't available in the Label field, so the expression must be entered instead of selected.
5. Repeat these steps for each column that you want to use a custom label.

SEE ALSO:

Salesforce Help: Custom Labels

Salesforce Help: Considerations for Translating Flows

Translate Flow Screen Components

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

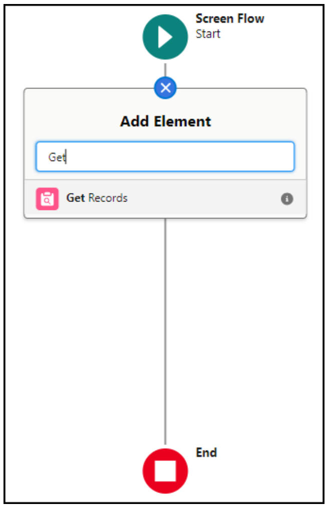
Flow Example: Calculate a Discount In Real Time

Use reactive components to enable a sales rep to calculate and apply a product discount on one screen.

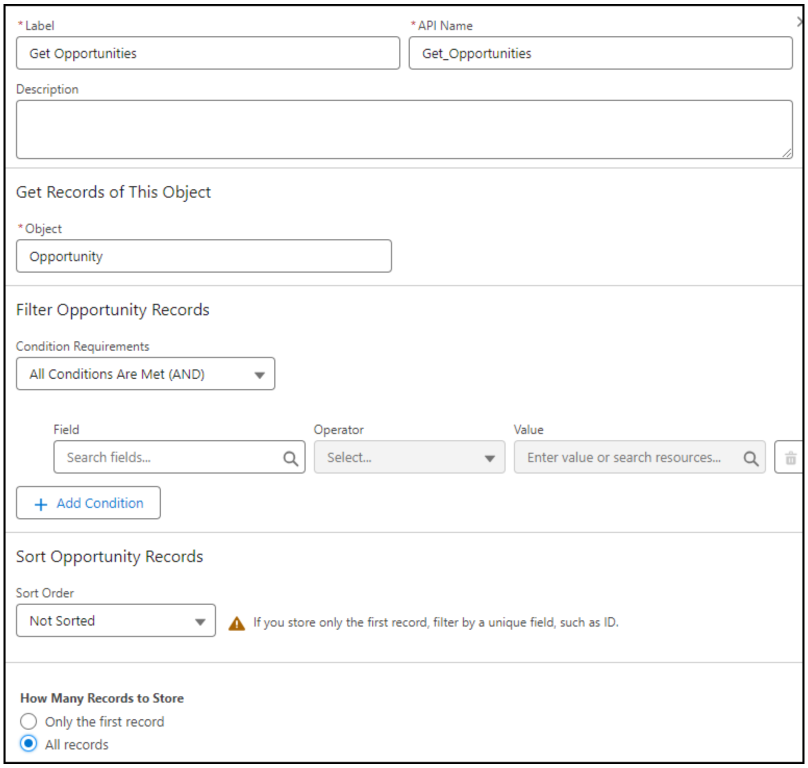
Available in: both Salesforce Classic and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

In this example, the screen includes a Data Table component with a list of opportunities that the sales rep can select from. The screen also includes a Slider component that represents the discount percentage. Finally, the screen includes a Display Text component that displays the discounted amount of the Opportunity.

1. Create a screen flow.
2. Add a Get Records data element to the flow.
   a. Select the Add Element icon on the canvas.
   b. Locate and select the **Get Records** data element.



   c. Configure the Get Records data element to retrieve and store a list of Opportunities.



*Label: Get Opportunities

*API Name: Get_Opportunities

Description:

Get Records of This Object

*Object: Opportunity

Filter Opportunity Records

Condition Requirements: All Conditions Are Met (AND)

Field	Operator	Value
Search fields...	Select...	Enter value or search resources...

Sort Opportunity Records

Sort Order: Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store

○ Only the first record

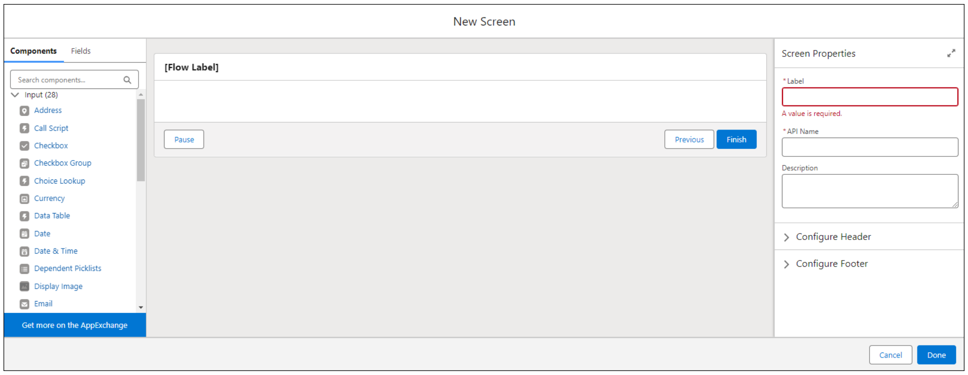
● All records

3. Add a screen to the flow.

   a. Select the Add Element icon on the canvas.

   b. Locate and select the **Screen** interaction element.

   Note: The New Screen window appears.



New Screen

[Flow Label]

Screen Properties

*Label:

A value is required.

*API Name:

Description:

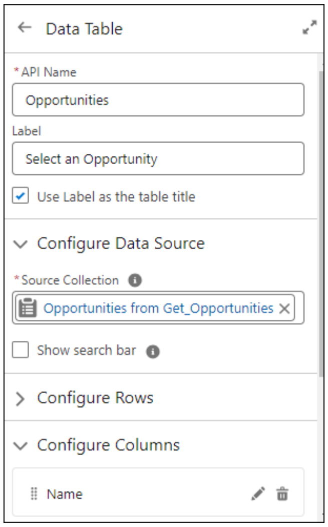
4. To generate a list of opportunities that a sales rep can select from, add a Data Table component to the screen.

   a. In the Components pane, select **Data Table**.

   b. Configure the component with these values.

      - API Name—Opportunities
      - Label—Select an Opportunity

- Use Label as the table title—Select the checkbox
- Source Collection—The list of opportunities you created in a previous step
- Configure Columns—Specify the opportunity Name and Amount as two separate columns



Data Table

* API Name

Opportunities

Label

Select an Opportunity

☑ Use Label as the table title

Configure Data Source

* Source Collection

Opportunities from Get_Opportunities

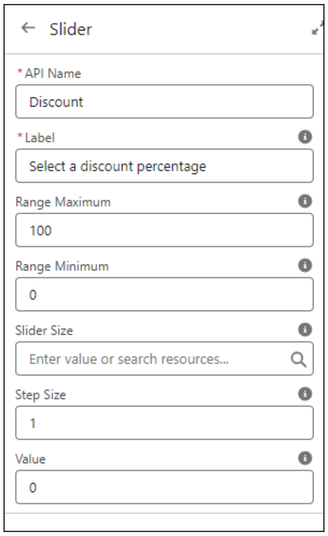
☐ Show search bar

Configure Rows

Configure Columns

Name

5. To enable a sales rep to select a discount, add a Slider component to the screen.
   a. In the Components pane, select **Slider**.
   b. Configure the component with these values.
      - API Name—Discount
      - Label—Select a discount percentage



Slider

* API Name

Discount

* Label

Select a discount percentage

Range Maximum

100

Range Minimum

0

Slider Size

Enter value or search resources...

Step Size

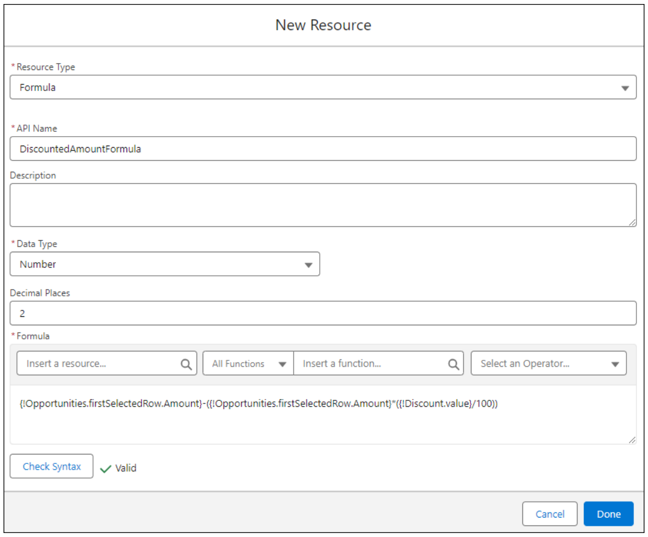
1

Value

0

6. To display the discounted amount of the opportunity, add a Display Text component to the screen.
   a. In the Components pane, select **Display Text**.

b. In the configuration pane for the Display Text component, create a formula resource to calculate the discounted amount. For example: `{!Opportunities.firstSelectedRow.Amount}-({!Opportunities.firstSelectedRow.Amount}*({!Discount.value}/100))`.



New Resource

* Resource Type: Formula

* API Name: DiscountedAmountFormula

Description:

* Data Type: Number

Decimal Places: 2

* Formula:

```
{!Opportunities.firstSelectedRow.Amount}-({!Opportunities.firstSelectedRow.Amount}*({!Discount.value}/100))
```

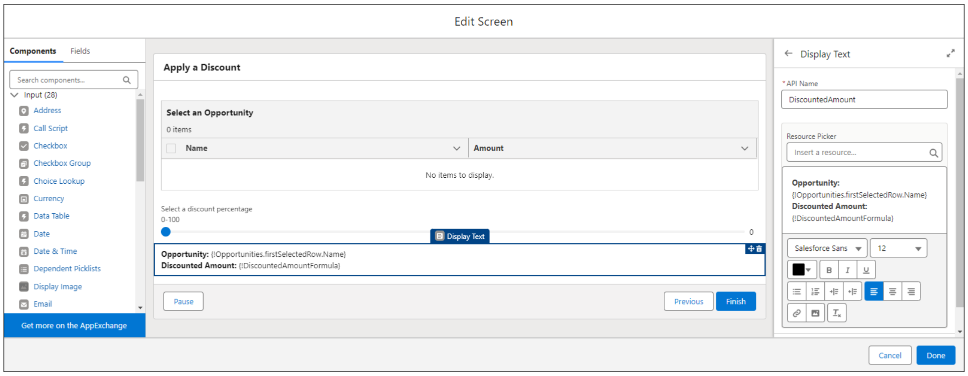
✓ Valid

c. Configure the component with these values.

- API Name—DiscountedAmount
- Canvas—`Opportunity: {!Opportunities.firstSelectedRow.Name}`

  `Discounted Amount: {!DiscountedAmountFormula}`

  This code displays the name of the first opportunity that the user selects in the Data Table component and calculates the discounted amount based on the Slider component value.



Edit Screen

Apply a Discount

Select an Opportunity

0 items

□	Name	Amount
	No items to display.	

Select a discount percentage

0-100

Opportunity: {!Opportunities.firstSelectedRow.Name}
Discounted Amount: {!DiscountedAmountFormula}

Display Text

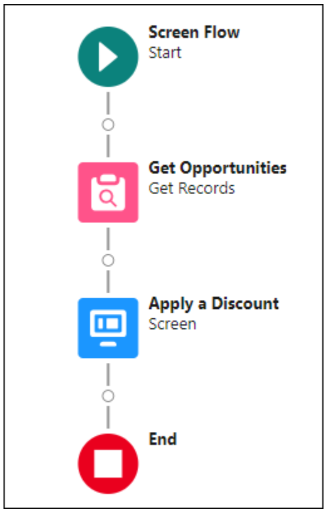
* API Name: DiscountedAmount

Opportunity:
{!Opportunities.firstSelectedRow.Name}
Discounted Amount:
{!DiscountedAmountFormula}

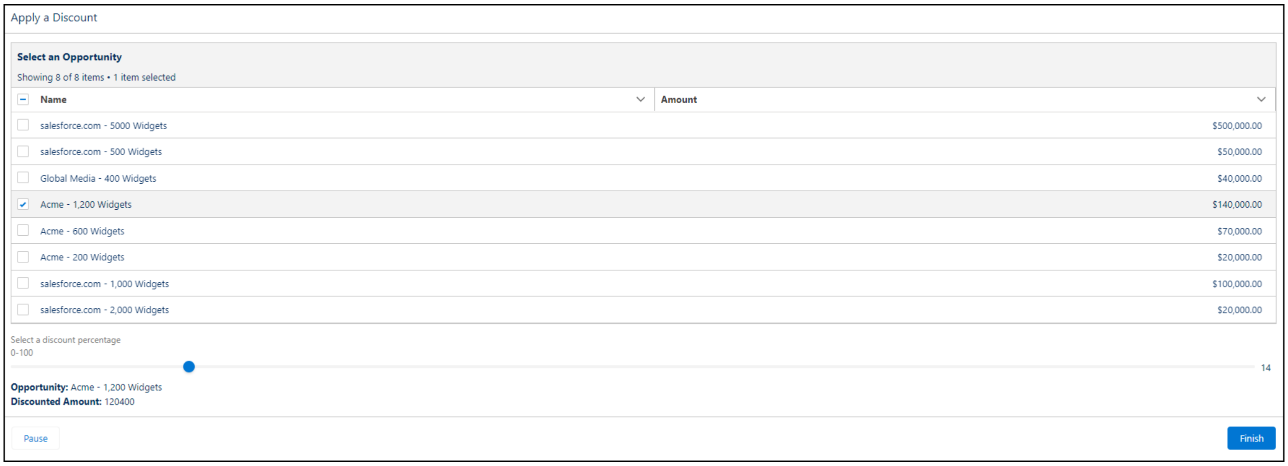
Salesforce Sans 12

7. Save and run the flow.

Note: The screen flow on the canvas includes two nodes.



Note: At runtime, the flow updates the Display Text component as the user selects opportunities and moves the slider.


Apply a Discount

Select an Opportunity

Showing 8 of 8 items • 1 item selected

⊟	Name	Amount
□	salesforce.com - 5000 Widgets	$500,000.00
□	salesforce.com - 500 Widgets	$50,000.00
□	Global Media - 400 Widgets	$40,000.00
☑	Acme - 1,200 Widgets	$140,000.00
□	Acme - 600 Widgets	$70,000.00
□	Acme - 200 Widgets	$20,000.00
□	salesforce.com - 1,000 Widgets	$100,000.00
□	salesforce.com - 2,000 Widgets	$20,000.00

Select a discount percentage
0-100

14

Opportunity: Acme - 1,200 Widgets
Discounted Amount: 120400

Flow Example: Retrieve Opportunities for an Account that a User Selects

Create a screen flow that enables a user to search for and select an Account record, and then click a button on the same screen to launch an autolaunched flow that retrieves all the Opportunity records associated with the Account record. Display the Opportunity records in a table on the same screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

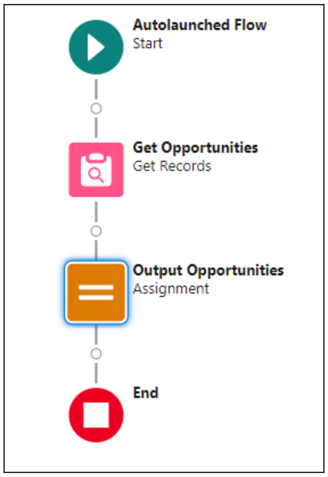
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

1. In Flow Builder, create an autolaunched flow.
2. To configure the autolaunched flow to retrieve all the Opportunity records associated with an Account record and save the Opportunity records to a record collection variable, create three variables:
   a. `inputAccountId`—A Text variable available for input.
   b. `inputAccountIds`—A Text collection variable available for input.
   c. `outputOpps`—A Record collection variable of Opportunity objects available for output.
3. Add a Get Records element to the flow with these values.
   - Label—`Get Opportunities`
   - API Name—`Get_Opportunities`
   - Object—Opportunity
   - Condition Requirements—Any Condition Is Met (OR)
   - Conditions—`AccountID` **Equals** `{!inputAccountId}` OR `AccountId` **In** `{!inputAccountIds}`
   - How Many Records to Store—All records
   - How To Store Record Data—Automatically store all fields
4. Add an Assignment element to the flow with these values.
   - Label—`Output Opportunities`
   - API Name—`Output_Opportunities`
   - Set Variable Values—`outputOpps` **Equals** `{!Get_Opportunities}`



5. In Flow Builder, create a screen flow.
6. Add a Screen element to the flow, and then add a Lookup component to the Screen element.

7. To retrieve the Account Ids of the accounts the user selects, configure the Lookup component with these values.
   - API Name—`lookupAccount`
   - Field API Name—`AccountId`
   - Label—`Account`
   - Object API Name—`Opportunity`
   - Maximum Selections—`50`

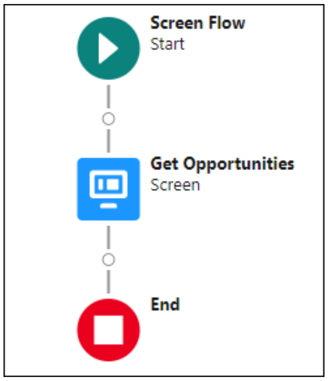
8. Add an Action Button component to the Screen element.

9. To run the autolaunched flow that you created in a previous step with the IDs of the records that the user selects as input, configure the Action Button component with these values.
   - API Name—`getOppsButton`
   - Label—`Get Opportunities`
   - Action—Auto_Get_Opps
   - Action Label—`Autolaunched Get Opps`
   - Action API Name—`Autolaunched_Get_Opps`
   - inputAccountId—`{!lookupAccount.recordId}`
   - inputAccountIds—`{!lookupAccount.recordIds}`

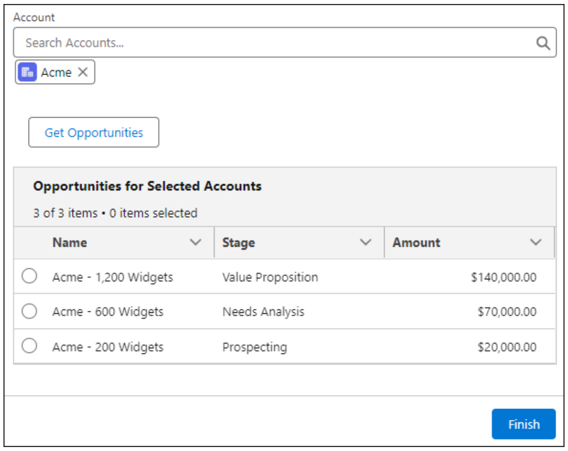
10. Add a Data Table component to the Screen element.

11. To display the list of Opportunity records returned by the Action Button component, configure the Data Table component with these values.
    - API Name—`dtOpps`
    - Label—`Opportunities for Selected Accounts`
    - Use Label as the table title—Selected
    - Source Collection—`{!Autolaunched_Get_Opps.Results.outputOpps}`
    - Configure Columns—**Name**, **Stage**, and **Amount**

12. Click **Done** in the Screen element.



13. Save and run the flow.



Opportunities for Selected Accounts

3 of 3 items • 0 items selected

	Name	Stage	Amount
○	Acme - 1,200 Widgets	Value Proposition	$140,000.00
○	Acme - 600 Widgets	Needs Analysis	$70,000.00
○	Acme - 200 Widgets	Prospecting	$20,000.00

Flow Example: Create a Contact for Each Beneficiary on a Policy

Add a Repeater component to a screen, and then add child components to represent a beneficiary on a policy. Loop over the output Repeater component to create a collection of Contact records. Then use the collection to create a Contact record for each beneficiary.

1. In Flow Builder, create a screen flow.
2. Create two text Choice resources with these values.
   - Child
   - Spouse
3. Add a Screen element to the flow with these values.
   - **Label**—`Beneficiaries`
   - **API Name**—`Beneficiaries`
4. Add a Repeater component to the Screen element.
5. In the API Name field, enter `AddBeneficiary`.
6. Add child components to the Repeater.
   a. Add a Text component with these values.
      - **Label**—`First Name`
      - **API Name**—`First_Name`
      - **Require**—Selected

   b. Add a Text component with these values.
      - **Label**—`Last Name`
      - **API Name**—`Last_Name`
      - **Require**—Selected

   c. Add a Date component with these values.
      - **Label**—`Date of Birth`
      - **API Name**—`Date_of_Birth`

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:
- Manage Flow

- **Require**—Selected

d. Add a Checkbox component with these values.

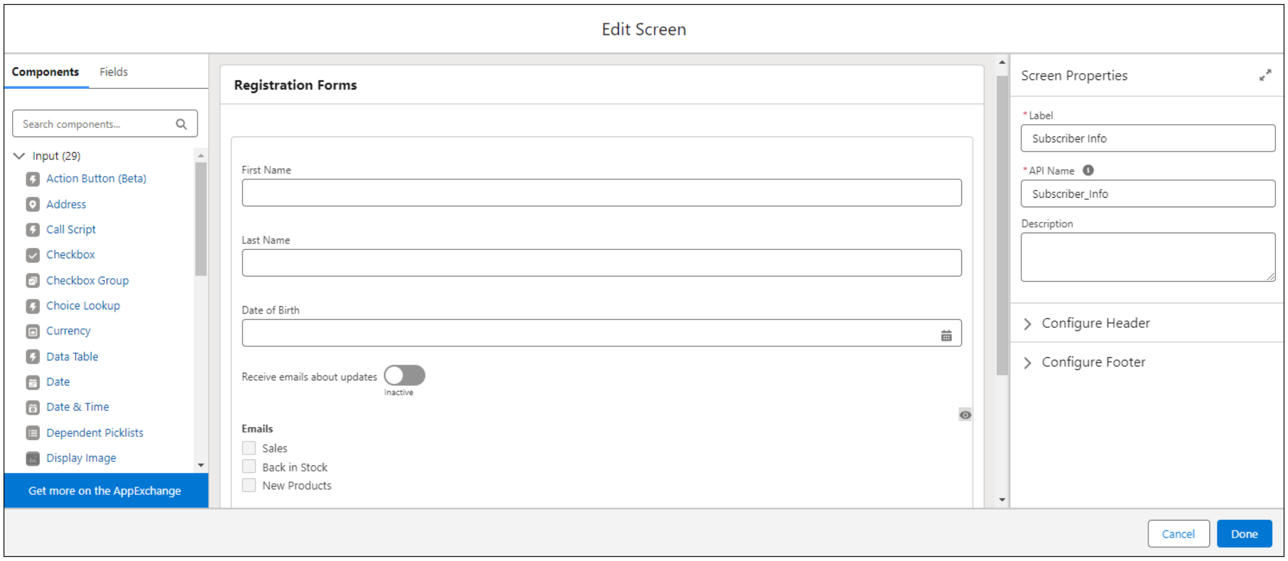
- **Label**—*`Subscriber`*
- **API Name**—*`Subscriber`*

e. Add a Picklist component with these values.

- **Label**—*`Relationship to Subscriber`*
- **API Name**—*`Relationship_to_Subscriber`*
- **Choice**—The text Choice resources that you created in a previous step
- **Set Component Visibility**—All Conditions are Met (AND) `{!AddBeneficiary.Subscriber} Equals {!$GlobalConstant.False}`

  This configuration specifies that the Picklist component appears only when the Checkbox component output is false.

f. Click **Done**.



*Label: Subscriber Info

*API Name: Subscriber_Info

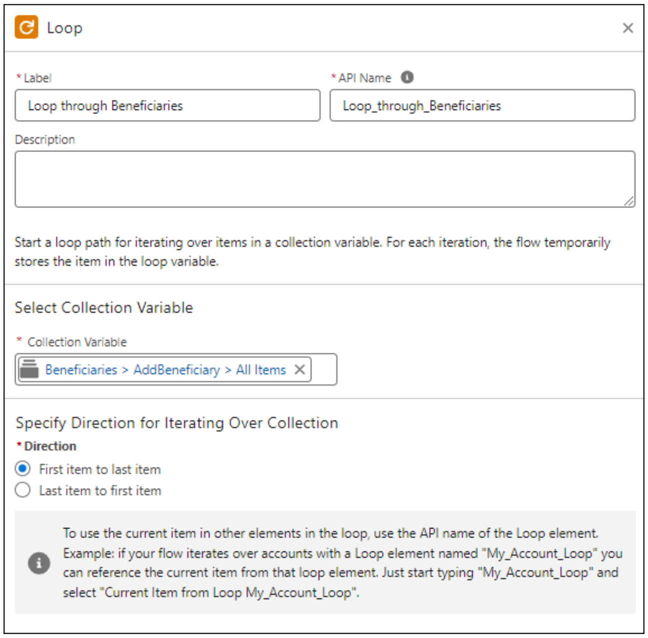
7. In the Flow Builder canvas, after the Screen element, add a Loop element to loop through the Repeater component output.

To use the Repeater component output elsewhere in the flow, you must loop over the output and save relevant data in a collection variable.

User input is stored in the `AllItems` attribute of the Repeater component. For example, if the API Name of a Repeater component is *`AddBeneficiary`*, you can access the user input in the `{!AddBeneficiary.AllItems}` resource.

8. Configure the Loop element with these values.

- **Label**—*`Loop through Beneficiaries`*
- **API Name**—*`Loop_through_Beneficiaries`*
- **Collection Variable**—*`{!AddBeneficiary.AllItems}`*



Loop

* Label
Loop through Beneficiaries

* API Name
Loop_through_Beneficiaries

Description

Start a loop path for iterating over items in a collection variable. For each iteration, the flow temporarily stores the item in the loop variable.

Select Collection Variable

* Collection Variable
Beneficiaries > AddBeneficiary > All Items

Specify Direction for Iterating Over Collection

* **Direction**

● First item to last item

○ Last item to first item

To use the current item in other elements in the loop, use the API name of the Loop element. Example: if your flow iterates over accounts with a Loop element named "My_Account_Loop" you can reference the current item from that loop element. Just start typing "My_Account_Loop" and select "Current Item from Loop My_Account_Loop".

9. Create a record variable with these values.

- **API Name**—`contactRecord`
- **Data Type**—Record
- **Object**—Contact

10. Create a record collection variable with these values.

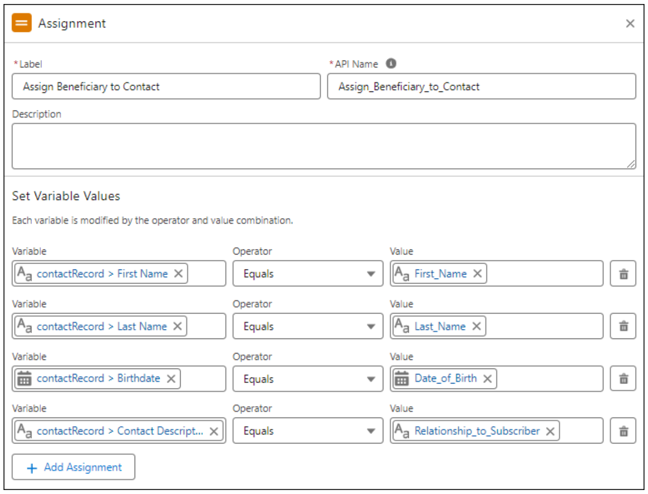
- **API Name**—`contacts`
- **Data Type**—Record
- **Allow Multiple Values**—Selected
- **Object**—Contact

11. After the For Each node of the Loop element, add two Assignment elements.

The first Assignment element assigns the values for each beneficiary to a Contact record variable. The second Assignment element adds the Contact records to a collection.

12. Configure the first Assignment element with these values.

- **Label**—`Assign Beneficiary to Contact`
- **API Name**—`Assign_Beneficiary_to_Contact`
- **Set Variable Values**—
  - `{!contactRecord.FirstName}` Equals `{!Loop_through_Beneficiaries.First_Name}`
  - `{!contactRecord.LastName}` Equals `{!Loop_through_Beneficiaries.Last_Name}`
  - `{!contactRecord.Birthdate}` Equals `{!Loop_through_Beneficiaries.Date_of_Birth}`
  - `{!contactRecord.Description}` Equals `{!Loop_through_Beneficiaries.Relationship_to_Subscriber}`



Assignment

*Label: Assign Beneficiary to Contact

*API Name: Assign_Beneficiary_to_Contact

Description

Set Variable Values

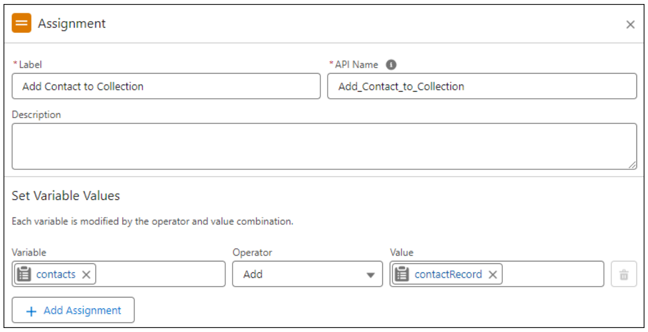
Each variable is modified by the operator and value combination.

Variable	Operator	Value
contactRecord > First Name	Equals	First_Name
contactRecord > Last Name	Equals	Last_Name
contactRecord > Birthdate	Equals	Date_of_Birth
contactRecord > Contact Descript...	Equals	Relationship_to_Subscriber

This configuration maps the values provided for each beneficiary to the values of a record variable.

13. Configure the second Assignment element with these values.

- **Label**—`Add Contact to Collection`
- **API Name**—`Add_Contact_to_Collection`
- **Set Variable Values**—`{!contacts}` Add `{!contactRecord}`



Assignment

*Label: Add Contact to Collection

*API Name: Add_Contact_to_Collection

Description

Set Variable Values

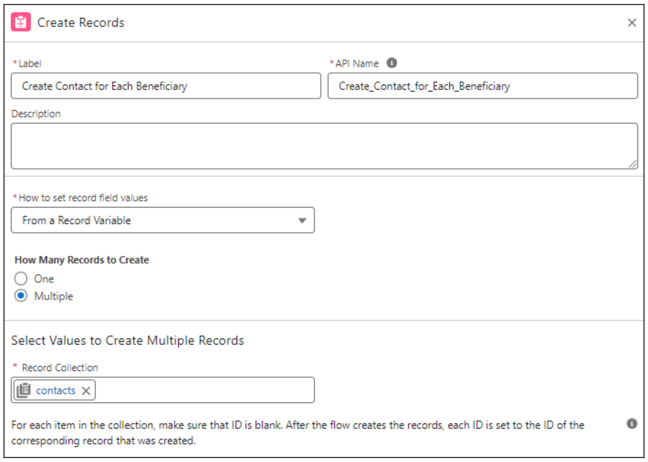
Each variable is modified by the operator and value combination.

Variable	Operator	Value
contacts	Add	contactRecord

This configuration adds the record variables from the previous step into a record collection variable.

14. To create a Contact record for each beneficiary, add a Create Records element to the flow with these values.

- **Label**—`Create Contact for Each Beneficiary`
- **API Name**—`Create_Contact_for_Each_Beneficiary`
- **How Many Records to Create**— Multiple
- **Record Collection**— `{!contacts}`



Create Records

*Label: Create Contact for Each Beneficiary

*API Name: Create_Contact_for_Each_Beneficiary

Description:

*How to set record field values: From a Record Variable

How Many Records to Create

○ One

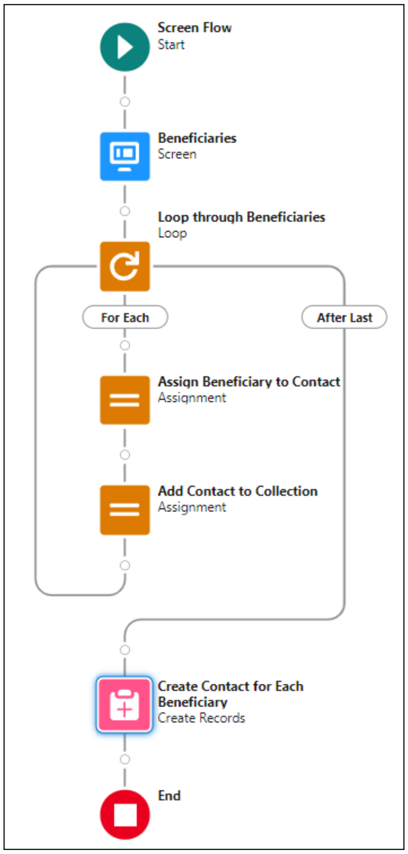
● Multiple

Select Values to Create Multiple Records

* Record Collection: contacts

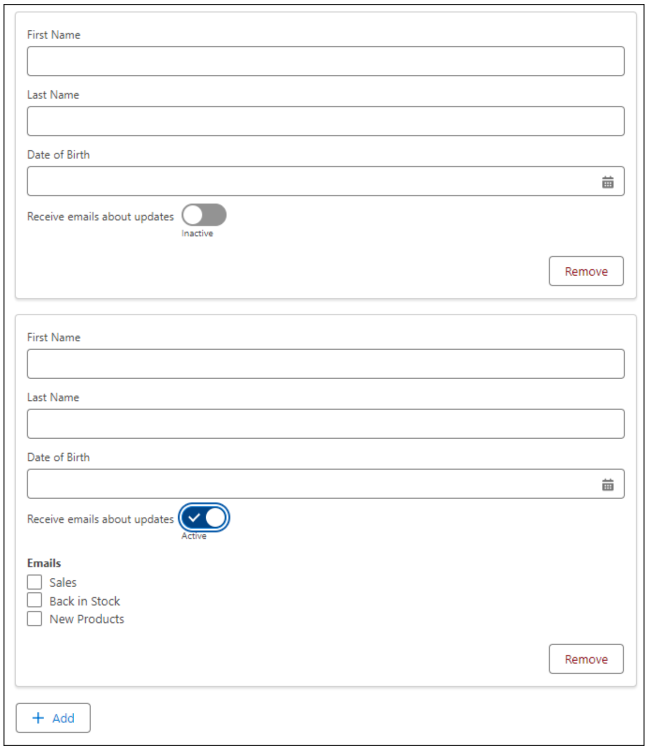
For each item in the collection, make sure that ID is blank. After the flow creates the records, each ID is set to the ID of the corresponding record that was created.

15. Save your work.



Screen Flow
Start
Beneficiaries
Screen
Loop through Beneficiaries
Loop
For Each
After Last
Assign Beneficiary to Contact
Assignment
Add Contact to Collection
Assignment
Create Contact for Each Beneficiary
Create Records
End

16. Run the flow.



First Name

Last Name

Date of Birth

Receive emails about updates Inactive

Remove

First Name

Last Name

Date of Birth

Receive emails about updates Active

Emails

□ Sales

□ Back in Stock

□ New Products

Remove

+ Add

Use Flows with Slack

Build a screen flow that's designed to run in a Slack conversation or direct message group. Use another flow, a Slack shortcut, a slash command, or a button in a Slack view to run a flow from Slack.

Create a Screen Flow to Run in Slack

Create a screen flow to run in Slack using supported flow screen components and text input variables.

Run Active Screen Flows from Slack

You can invoke an active screen flow using an action, a Slack shortcut in a custom Slack app, or a button in a Slack view.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flows in Slack

Create a Screen Flow to Run in Slack

Create a screen flow to run in Slack using supported flow screen components and text input variables.

We recommend that you create and test flows in a sandbox before deploying them in production.

1. Create a screen flow and add a Screen element to it.
2. Click **Save**.
3. In the Save the flow dialog box, fill in the fields, and then click **Advanced**.
4. Select **Make Available in Slack** and save the flow.
   The Screen element window displays only the screen components that can run in Slack.
5. Build your flow.

   Flows in Slack don't support field-level validation for screen components.
6. If you want the flow to be passed input when it's run, add text variables that are available for input.

   Flows in Slack support only text variables as input parameters.
7. Save and test your flow.
8. Activate the flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:
- Manage Flow

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flows in Slack

Run Active Screen Flows from Slack

You can invoke an active screen flow using an action, a Slack shortcut in a custom Slack app, or a button in a Slack view.

Send a Flow to Run in Slack

An active screen flow with the Make Available in Slack advanced setting can run in the flow default environment. To send a Slack message that contains a button to run that flow to a Slack channel or group direct message, create another flow that contains an action.

Run a Screen Flow from a Button in a Slack View (Beta)

Build a Slack view to invoke an active screen flow when a user clicks a button in Slack.

Run a Screen Flow from a Slack Shortcut (Beta)

Create a custom Slack app to invoke an active screen flow from a global shortcut, a message shortcut, or a slash command in Slack.

Send a Flow to Run in Slack

An active screen flow with the Make Available in Slack advanced setting can run in the flow default environment. To send a Slack message that contains a button to run that flow to a Slack channel or group direct message, create another flow that contains an action.

The Slack Send Message to Launch Flow action work with only official Salesforce Slack apps.

1. Edit or create a flow that supports Slack Send Message to Launch Flow, and add an Action element.

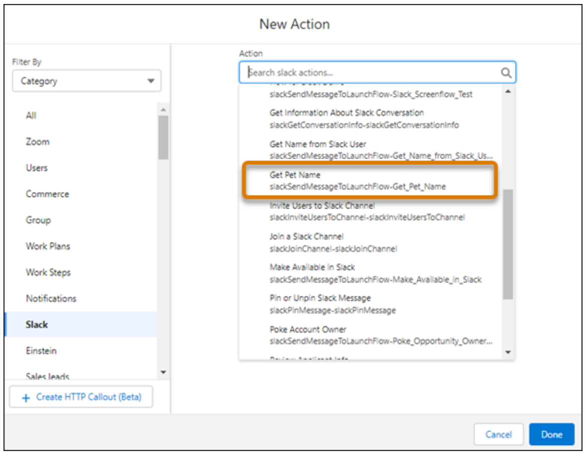
   These flow types don't support the Slack Send Message to Launch Flow action: CMS Orchestrator, EvaluationFlow, Journey, Orchestrator, Survey, SurveyEnrich, and TransactionSecurityFlow.

2. In the Action element dialog box, select **Slack** in the Categories panel.

3. In the Action field, select the version of the slackSendMessageToLaunchFlow action that has the name of the active flow that you created.

   The Slack Send Message to Launch Flow action shows versions for only active screen flows that have the Make Available in Slack advanced option selected.

   The version of slackSendMessageToLaunchFlow to select for the Get Pet Name flow is Get Pet Name. The API name is slackSendMessageToLaunchFlow - Get_Pet_Name.


New Action
Filter By
Category
All
Zoom
Users
Commerce
Group
Work Plans
Work Steps
Notifications
Slack
Einstein
Sales leads
+ Create HTTP Callout (Beta)
Action
Search slack actions...
slackSendMessageToLaunchFlow-Slack_Screenflow_Test
Get Information About Slack Conversation
slackGetConversationInfo-slackGetConversationInfo
Get Name from Slack User
slackSendMessageToLaunchFlow-Get_Name_from_Slack_Us...
Get Pet Name
slackSendMessageToLaunchFlow-Get_Pet_Name
Invite Users to Slack Channel
slackInviteUsersToChannel-slackInviteUsersToChannel
Join a Slack Channel
slackJoinChannel-slackJoinChannel
Make Available in Slack
slackSendMessageToLaunchFlow-Make_Available_in_Slack
Pin or Unpin Slack Message
slackPinMessage-slackPinMessage
Poke Account Owner
slackSendMessageToLaunchFlow-Poke_Opportunity_Owner...
Cancel
Done

4. In the New Action dialog, fill in the fields, and then click **Done**.

   You can select only an official Salesforce Slack app for the Slack app field.

   The Get Pet Name version of the Slack Send Message to Launch Flow action has one input value specific to Get Pet Name: petOwner. All other inputs are standard to the Slack Send Message to Launch Flow action.

EDITIONS

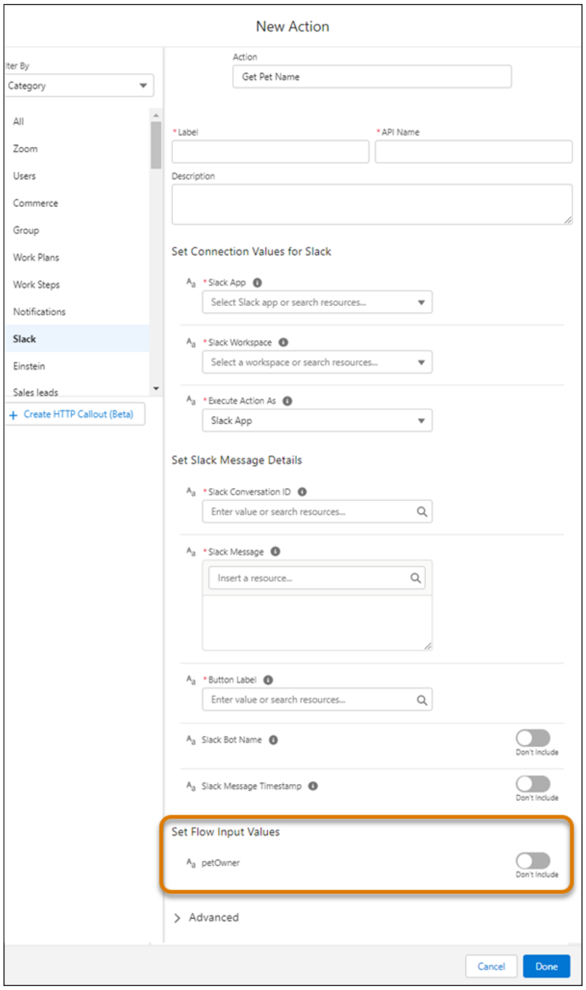
Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow



New Action

Action: Get Pet Name

Execute Action As: Slack App

5. Save and activate the flow.

SEE ALSO:

Flow Core Actions for Slack: Send Message to Launch Flow

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flows in Slack

Run a Screen Flow from a Button in a Slack View (Beta)

Build a Slack view to invoke an active screen flow when a user clicks a button in Slack.

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Use the Apex SDK for Slack (Beta) to build and deploy views in Slack.

1. Create a view definition.
2. Add a button component to invoke an active screen flow with the Make available in Slack advanced setting enabled.
3. Test the button in your view in Slack.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Apex SDK for Slack (Beta): Define a View

Apex SDK for Slack (Beta): Interactive Components

Run a Screen Flow from a Slack Shortcut (Beta)

Create a custom Slack app to invoke an active screen flow from a global shortcut, a message shortcut, or a slash command in Slack.

Note: This feature is a Beta Service. Customer may opt to try such Beta Service in its sole discretion. Any use of the Beta Service is subject to the applicable Beta Services Terms provided at Agreements and Terms.

Custom Slack apps don't have access to Slack Flow Core Actions.

Use the Apex SDK for Slack (Beta) to create and deploy global shortcuts, message shortcuts, or slash commands in Slack.

1. Create a custom Slack app.
2. Add the shortcuts and slash commands to your custom app configuration. using the Create an app tool from the slack api website.
3. Deploy the custom Slack app's metadata with Salesforce Developer Experience.
4. Connect the shortcut or slash command with your org.
5. Test the shortcuts and slash commands in Slack.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: This example shows the metadata to add for the /getPetName slash command that runs the Get_Pet_Name flow from Slack.

```
description: App configuration for running flows in slack
commands:
  /getPetName:
    action:
      definition: flow__requestHandler
      properties:
        flowApiName: Get_Pet_Name
    title: Launch flow command
    description: Launches a flow in a modal from slack
```

SEE ALSO:

Apex SDK for Slack (Beta): Define an App

Apex SDK for Slack (Beta): Shortcuts and Slash Commands

Apex SDK for Slack (Beta): Sample Apex SDK for Slack App

Use Flows with MuleSoft RPA

Register a MuleSoft RPA process as an external service. Then, build a flow that starts and checks the status of the MuleSoft RPA process.

With MuleSoft Robotic Process Automation (RPA), you can automate business processes that typically require human input and interact with systems that don't have an API. Use MuleSoft RPA Manager to publish REST APIs for your RPA processes to Anypoint Exchange, which you can then register as external services in Salesforce and invoke from a flow.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Make a MuleSoft RPA Process Available to Flows

Configure the connection between MuleSoft Anypoint platform and your org to import MuleSoft RPA APIs. Then, register the APIs as an external service to use them in flows.

Invoke a MuleSoft RPA Process in a Flow

After you make a MuleSoft RPA process available to flows, you can start and check the status of the process from a flow.

Make a MuleSoft RPA Process Available to Flows

Configure the connection between MuleSoft Anypoint platform and your org to import MuleSoft RPA APIs. Then, register the APIs as an external service to use them in flows.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Before you can make a MuleSoft RPA process available to flows:

- In MuleSoft RPA Manager, publish the RPA process as a REST API to Anypoint Exchange.
- In MuleSoft RPA Manager, create an API key for yourself and store it in a secure place. Optionally, copy the API key to use it in the creation of an external credential.
- In Salesforce, create and assign a permission set that enables users to invoke this MuleSoft RPA process in a flow.

1. Connect Salesforce to MuleSoft Anypoint Exchange from Setup, create the initial connection between MuleSoft Anypoint platform and your org.
2. Create an external credential for authenticating to MuleSoft RPA Manager with these values. The external credential details how Salesforce authenticates to MuleSoft RPA Manager. You link the external credential to a user's permission set and to the named credential that specifies the URL for MuleSoft RPA Manager.

Section	Value
Label	A user-friendly name for the external credential that's displayed in the Salesforce user interface, such as in list views.

Section	Value
Name	A unique identifier that's used to refer to this external credential from callout definitions and through the API. The name can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Authentication Protocol	**Custom**
Principals	Create a principal with these values. **Sequence Number** *1* **Authentication Parameters** Add an authentication parameter whose value is the API key that you created before in RPA Manager. You use the name of the authentication parameter in the custom header for this external credential.

Custom Headers	Create a custom header with these values.
	Name
	`x-apikey`
	This value is case-sensitive.
	Value
	`{!$Credential.External_Credential_Name.RPA_API_Key}`
	`External_Credential_Name` is the name of the external credential.
	`RPA_API_Key` is the name of the authentication parameter that you created to store the API key that you created before in RPA Manager.
	Sequence Number
	1

3. Enable the external credential principals using the permission set you created to enable users to invoke this RPA process.
4. Create a named credential that stores the URL for MuleSoft RPA Manager.

   You link the named credential to the external credential you created in the previous step.

Section	Value
Label	A user-friendly name for the named credential that's displayed in the Salesforce user interface, such as in list views.
Name	A unique identifier that's used to refer to this named credential from callout definitions and through the API. The name can contain only underscores and alphanumeric characters. It must be unique, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Section	Value
URL	Your MuleSoft RPA Manager URL, for example, `https://myOrg.rpa.mulesoft.com`.
External Credential	Select the external credential that you created in the previous step.
Generate Authorization Header	Deselect this option.
Allow Formulas in HTTP Header	Select this option.

5. Import the MuleSoft RPA API into Salesforce and use the named credential you created in the previous steps in the Import Service window.
6. Select the `startProcess` and `getProcessExecutionStatus` operations for the MuleSoft RPA API.

Use the MuleSoft RPA process in a flow.

SEE ALSO:

Salesforce Help : Permission Sets

MuleSoft RPA: Publishing Automation Assets to Anypoint Exchange

Invoke a MuleSoft RPA Process in a Flow

After you make a MuleSoft RPA process available to flows, you can start and check the status of the process from a flow.

You add invocable actions for the `startProcess` and `getProcessExecutionStatus` operations that you selected when you registered the RPA process as an external service.

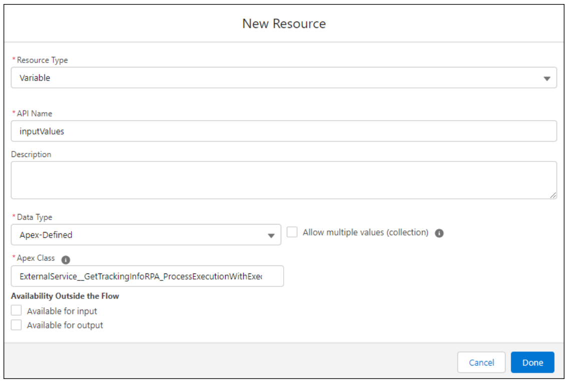
1. In Flow Builder, create a flow.
2. To store the input variables for the RPA process that you want to invoke, create a resource with these values.

Field	Value
Resource Type	Variable
Data Type	Apex-Defined
Apex Class	*ExternalService_External_Service_Name_ProcessExecutionWithExecutionId* `External_Service_Name` is the name that you provided when you registered the MuleSoft RPA API as an external service.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



New Resource

* Resource Type: Variable

* API Name: inputValues

Description:

* Data Type: Apex-Defined □ Allow multiple values (collection)

* Apex Class: ExternalService__GetTrackingInfoRPA_ProcessExecutionWithExe

Availability Outside the Flow

□ Available for input

□ Available for output

3. Assign values to the input variables.

   a. On the Flow Builder canvas, add an assignment node.

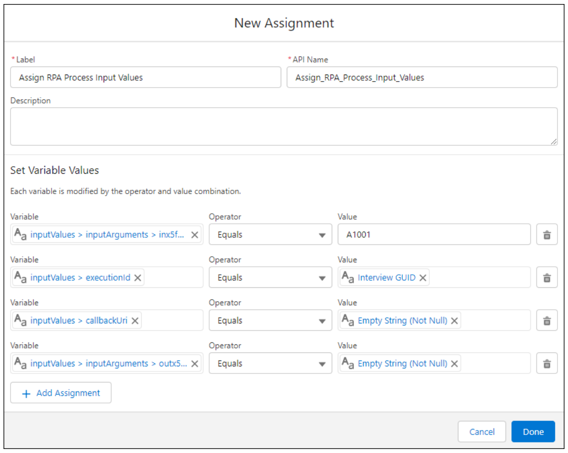
   b. Assign values to the variable that you created in the previous step.

      A variable's input values differ for each RPA process. In this example, the process requires an order number and a tracking number.

      Every input requires a value, even if the value is an empty string. If you don't specify a value for each input, the RPA process invocation generates an error.

      Each RPA process invocation requires a unique `executionId`. If your flow starts only one RPA process, you can use the `$Flow.InterviewGuid` global variable, which is the flow interview ID.

      Callbacks for MuleSoft RPA processes aren't supported in flows. Specify the `$GlobalConstant.EmptyString` global variable as the value of the `callbackUri` variable.



New Assignment

* Label: Assign RPA Process Input Values

* API Name: Assign_RPA_Process_Input_Values

Description:

Set Variable Values

Each variable is modified by the operator and value combination.

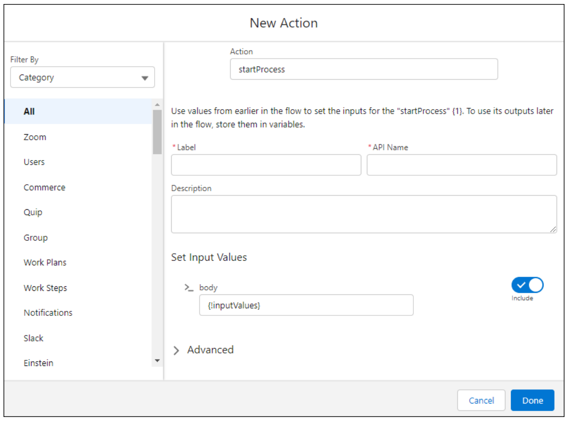
Variable	Operator	Value
inputValues > inputArguments > inx5f...	Equals	A1001
inputValues > executionId	Equals	Interview GUID
inputValues > callbackUri	Equals	Empty String (Not Null)
inputValues > inputArguments > outx5...	Equals	Empty String (Not Null)

4. Add an action for the `startProcess` operation to the flow.

   a. On the Flow Builder canvas, add an action node.

   b. In the Action field, type *`startProcess`*, and select the **startProcess** action.

   c. Configure the action, setting the body to the resource that you created to store the input values.



New Action

Action: startProcess

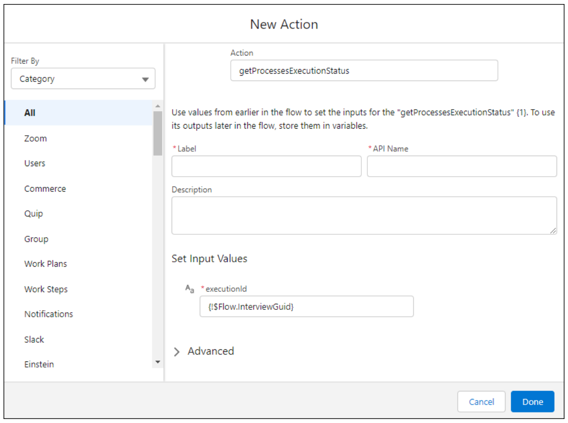
Use values from earlier in the flow to set the inputs for the "startProcess" {1}. To use its outputs later in the flow, store them in variables.

Set Input Values

body: {!inputValues}

   d. Click **Done**.

5. Add an action for the `getProcessExecutionStatus` operation to the flow.
   a. On the Flow Builder canvas, add an action node.
   b. In the Action field, type `getProcessExecutionStatus` operation, and select the **getProcessExecutionStatus** action.
   c. Configure the action, setting the executionId to the value of the `executionID` parameter that you specified for starting the MuleSoft RPA process.



New Action

Action: getProcessesExecutionStatus

Use values from earlier in the flow to set the inputs for the "getProcessesExecutionStatus" {1}. To use its outputs later in the flow, store them in variables.

Set Input Values

* executionId: {!$Flow.InterviewGuid}

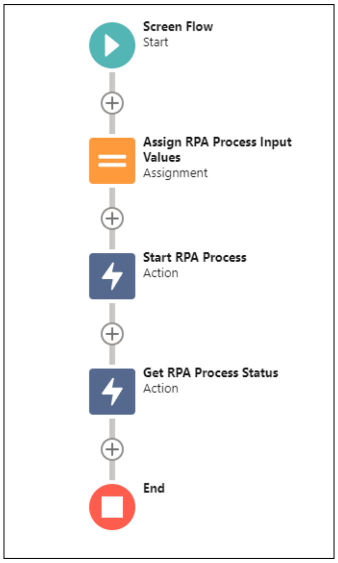
   d. Click Done.

6. Save and run your flow.

   The output of the `startProcess` action indicates whether the process started successfully.

   The output of the `getProcessExecutionStatus` action indicates the status of the RPA process.

Example: This simple screen flow invokes a MuleSoft RPA process called Get Tracking Info. It includes an Assignment node for input variables and actions that start and check the status of the process.



SEE ALSO:

- Build a Flow
- Flow Element: Assignment
- Flow Resource: Variable

Extend Your Flow-Building Options

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Sometimes your flow must do more than what Flow Builder provides out of the box. You can extend flows by calling Apex classes or adding Lightning components.

Extend Flows with Apex
With Apex, create custom functionality in flows.

Extend Flows with Lightning Components
To build a richer flow screen, add Aura components or Lightning web components to your flow. To perform an action without going through the server, add Aura components to your flow.

Extend Flows with Apex

With Apex, create custom functionality in flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Let Flows Execute Apex Actions
Flow Builder comes with much functionality, but sometimes your flow must do more than the default elements allow. In that case, call an Apex class from your flow by using an Apex action.

Extend Flows with the Apex-Defined Data Type
Using a combination of Apex, flow, and Lightning components, you can automate business processes that involve complex data objects typically returned from calls to web services. For example, get external product data about a web product and then create records in Salesforce.

Let Flows Execute Apex Actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Builder comes with much functionality, but sometimes your flow must do more than the default elements allow. In that case, call an Apex class from your flow by using an Apex action.

Developers have two options when they're trying to make an Apex class available as an Action element for a flow. While the `Process.Plugin` interface supports customizing how the class appears in the palette, the `@InvocableMethod` annotation provides more functionality. The following table describes the features supported by each option.

Tip: We recommend using the `@InvocableMethod` annotation instead of the `Process.Plugin` interface.

Important: Legacy Apex actions aren't supported in auto-layout in Flow Builder. Legacy Apex actions are only available to be added in free-form in Flow Builder. Existing actions can be edited in both auto-layout and free-form mode.

	`Process.Plugin` **Interface**	`@InvocableMethod` **Annotation**
Apex data type support	Doesn't support: • Blob • Collection • sObject • Time	Doesn't support: • Generic Object • Sets • Maps • Enums Flow Builder doesn't support mapping an Apex method's input or output parameters to a record collection variable.
Bulk operations	Not supported	Supported
Custom icons	Not supported	Supported
Element type in Flow Builder	Action	Action
Element name in Flow Builder	Class name or the value of the `name` property.	Class name

	`Process.Plugin` **Interface**	`@InvocableMethod` **Annotation**
Reusability	Classes with this interface implemented are available in flows	Classes with this annotation implemented are available in: • Flows • Processes • Rest API
More Details in *Apex Developer Guide*	Passing Data to a Flow Using the `Process.Plugin` Interface	`InvocableMethod` Annotation and `InvocableVariable` Annotation

Example: To illustrate the difference between these two implementation methods, here are two classes that do the same thing: get an account name from a flow and return that account's ID.

This class implements the `@InvocableMethod` annotation.

```
global class lookUpAccountAnnotation {
   @InvocableMethod
   public static List<String> getAccountIds(List<String> names) {
      List<Id> accountIds = new List<Id>();
      List<Account> accounts = [SELECT Id FROM Account WHERE Name in :names];
      for (Account account : accounts) {
         accountIds.add(account.Id);
      }
      return accountIds;
   }
}
```

Note: If a flow invokes Apex, the running user must have the corresponding Apex class assignment in their profile or permission set.

This class implements the `Process.Plugin` interface.

```
global class lookUpAccountPlugin implements Process.Plugin {

   global Process.PluginResult invoke(Process.PluginRequest request) {
      String name = (String) request.inputParameters.get('name');
      Account account = [SELECT Id FROM Account WHERE Name = :name LIMIT 1][0];

      Map<String,Object> result = new Map<String,Object>();
      result.put('accountId', account.Id);
      return new Process.PluginResult(result);
   }

   global Process.PluginDescribeResult describe() {
      Process.PluginDescribeResult result = new Process.PluginDescribeResult();
      result.Name = 'Look Up Account ID By Name';
      result.Tag = 'Account Classes';
      result.inputParameters = new
         List<Process.PluginDescribeResult.InputParameter>{
            new Process.PluginDescribeResult.InputParameter('name',
            Process.PluginDescribeResult.ParameterType.STRING, true)
         };
      result.outputParameters = new
         List<Process.PluginDescribeResult.OutputParameter>{
            new Process.PluginDescribeResult.OutputParameter('accountId',
            Process.PluginDescribeResult.ParameterType.STRING)
                };
      return result;
   }
}
```

Notice that `lookupAccountAnnotation` is less than half the length (11 lines) of `lookupAccountPlugin` (28 lines). In addition, because the annotation supports bulk operations, `lookupAccountAnnotation` performs one query per batch of interviews. `lookupAccountPlugin` performs one query per interview.

SEE ALSO:

Flow Elements

Extend Flows with the Apex-Defined Data Type

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Using a combination of Apex, flow, and Lightning components, you can automate business processes that involve complex data objects typically returned from calls to web services. For example, get external product data about a web product and then create records in Salesforce.

A call to a REST endpoint returns a JSON response with product data.

```
{
   "model": "Vintage Cruiser Jacket",
   "brand": "Acme",
   "identifiers": [
     {
       "SKU": "A-J001"
     },
     {
       "SKU": "A-J002"
     },
     {
       "SKU": "A-J003"
     }
   ],
   "price": {
     "amount": {
       "currencyValue": "500.00",
       "name": "Amount",
       "currency": "USD"
     },
     "salesUnit": {
       "code": "EA",
       "name": "Each"
     }
   }
}
```

Model JSON Objects with an Apex Class

Define a complex data object in an Apex class. For example, you could create a matching Apex class for a JSON object. The class acts as a template and converts the JSON into objects that a flow can use.

Create an Apex Action

After you create a mechanism to translate JSON data into flow resources, you can create the Apex class that retrieves the JSON data. Use the `@InvocableMethod` annotation to define the class as an Apex action for flows.

Configure an Apex Action

Create a screen flow and add an Apex action to it.

Add a Custom Icon to an Apex-Defined Action

Add custom icons to your Apex-defined invocable actions to make them easier to find on the Flow Builder canvas.

SEE ALSO:

Considerations for the Apex-Defined Data Type

Flow Resource: Variable

Apex Developer Guide : `AuraEnabled` Annotation

Apex Developer Guide : Invoking HTTP Callouts

Apex Developer Guide : JSON Parsing

Model JSON Objects with an Apex Class

Define a complex data object in an Apex class. For example, you could create a matching Apex class for a JSON object. The class acts as a template and converts the JSON into objects that a flow can use.

After you define the Apex classes for a web product, you can create an Apex action to get the data from a web service using a REST call. Store the returned data in flow variables that use the Apex-defined data type. Then operate on the data in flows, Apex, and Lightning components.

Example: Here's an example of a WebProduct Apex template. It contains the main `WebProduct` class and classes for `Price` and `Identifiers`. `WebProduct` has three types of data.

- The `model` and `brand` fields use the primitive String data type.
- The `price` field references the `Price` class. Make sure that you save each class in separate class files because inner classes aren't supported in the Apex-defined data type for flows.
- The `identifiers` field is similar to `price`, but is represented as a List. The `Identifiers` class lets a flow access JSON arrays to represent a list of three specific SKU numbers.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- The `parse` method carries out the conversion of the JSON to the flow-accessible Apex objects.
- Fields in an Apex class for flows require the `@AuraEnabled` annotation.

```
public class WebProduct {

@AuraEnabled
public String model;

@AuraEnabled
public String brand;

//Because the identifiers information is an array in the JSON schema, we create a
separate class called Identifiers and define a property that contains a List.
@AuraEnabled
public List<Identifiers> identifiers;

//This field has an Apex-defined data type in the Price Apex class.
@AuraEnabled
public Price price;

//This method is used as part of the Create Apex Action section.
public static WebProduct parse(String json) {
return (WebProduct) System.JSON.deserialize(json, WebProduct.class);
}
```

The `Identifiers` Apex class defines the `SKU` field.

```
public class Identifiers {

@AuraEnabled
public String SKU;

}
```

The `Price` Apex class defines the `amount` and `salesUnit` fields.

```
public class Price {

//This field has an Apex-defined data type in the Amount Apex class.
@AuraEnabled
public Amount amount;


//This field has an Apex-defined data type in the SalesUnit Apex class.
@AuraEnabled
public SalesUnit salesUnit;

}
```

The `Amount` Apex class defines the `currencyValue`, `name`, and `currency` fields.

```
public class Amount {

@AuraEnabled
public String currencyValue;
@AuraEnabled
public String name;

@AuraEnabled
public String currency;

}
```

The `SalesUnit` Apex class defines the `code` and `name` fields.

```
public class SalesUnit {

@AuraEnabled
public String code;

@AuraEnabled
public String name;

}
```

Tip: You can use third-party tools to convert JSON to Apex.

Create an Apex Action

After you create a mechanism to translate JSON data into flow resources, you can create the Apex class that retrieves the JSON data. Use the `@InvocableMethod` annotation to define the class as an Apex action for flows.

The `GetWebProduct` Apex class defines an Apex action. It gets product data from a REST endpoint and parses the JSON response into the `WebProduct` Apex object.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

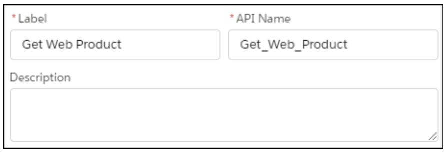
Example:

```
global with sharing class GetWebProduct {

    @InvocableMethod
    public static List<Results> GetWebProduct(List<Requests>
 requests)
       {
            Http http = new Http();
            HttpRequest request = new HttpRequest();
            String endpoint = requests[0].endpoint;
            String submittedId = requests[0].submittedId;
            request.setEndpoint(endpoint + submittedId );
            request.setMethod('GET');
            HttpResponse response = http.send(request);

            Results curResult = new Results();
            String responseJSON = response.getBody();
            WebProduct curProduct = WebProduct.parse(responseJSON);
            curResult.WebProduct = curProduct;

            List<Results> resultsList = new List<Results>();
            resultsList.add(curResult);
            return resultsList;
        }

    global class Requests {
        @InvocableVariable
        global String submittedId;

        @InvocableVariable
        global String endpoint;
    }

    global class Results {
        @InvocableVariable
        global WebProduct WebProduct;
    }

}
```

Configure an Apex Action

Create a screen flow and add an Apex action to it.

Example: The Get Web Product Apex action stores the web product data in {!Get_Web_Product} automatically.



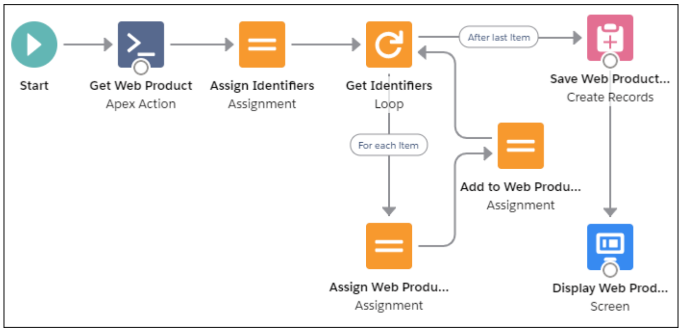
* Label	* API Name
Get Web Product	Get_Web_Product
Description	

Now you can operate on the data in flows, Apex, and Lightning components. For example, build a flow to generate the product data by looping over the identifiers and assembling a collection of products. Save the collection of records to Salesforce, and display the data on a flow screen component.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Add a Custom Icon to an Apex-Defined Action

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Add custom icons to your Apex-defined invocable actions to make them easier to find on the Flow Builder canvas.

The custom icon can be an SVG file or an existing Salesforce Lightning Design System (SLDS) icon.

SVG files must meet these requirements.

- The `<svg>` element in the file includes the `id`, `xmlns`, and `viewBox` attributes.
- The `<svg>` element in the file doesn't include the `style`, `height`, and `width` attributes.
- The file doesn't include a `<clipPath>` element.
- Each `<path>` element in the file includes a `fill` attribute.

For example, this SVG file is supported as a custom icon.

```
<svg id="top" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 48 48"><path fill="#673ab7"

d="M12,44c-1.657,0-3-1.343-3-3V7c0-1.657,1.343-3,3-3h16l11,11v26c0,1.657-1.343,3-3,3H12z"/><path
 fill="#5e35b1" d="M39 15L28.883 14.125 39 24.124z"/><path fill="#b39ddb"
d="M39,15h-8c-1.657,0-3-1.343-3-3V4L39,15z"/><path fill="#f5f5f5" d="M22 23H32V25H22zM22
28H32V30H22zM22 33H32V35H22zM17.5 22.5A1.5 1.5 0 1 0 17.5 25.5 1.5 1.5 0 1 0 17.5 22.5zM17.5
 27.5A1.5 1.5 0 1 0 17.5 30.5 1.5 1.5 0 1 0 17.5 27.5zM17.5 32.5A1.5 1.5 0 1 0 17.5 35.5
1.5 1.5 0 1 0 17.5 32.5z"/></svg>
```

Only Standard Object and Custom Object SLDS icons are supported as custom icons for Apex-defined invocable actions.

1. If you're using an SVG file, upload the file as a static resource in your org.
2. Set the `iconName` attribute of the `InvocableMethod` annotation for the invocable action to the name of the icon.

For SVG icons, specify the name as `resource:`***`namespace__iconName`***`:`***`svgID`***. For example:

```
public class CustomSvgIcon {

@InvocableMethod(iconName='resource:google:top')
public static void run() {}

}
```

```
public class CustomSvgIcon {

@InvocableMethod(label='myIcon' iconName='resource:myPackageNamespace__google:top')
global static List<Integer> myMethod(List<Integer> request) {
List<Integer> results = new List<Integer>();
results.add(1);
return results;

}
```

- *`iconName`* is the name that you specified when you uploaded the icon.
- *`svgID`* is the value of the `id` attribute of the <svg> element in the SVG file.
- *`namespace`* is the namespace of the package that includes the invocable action to customize. If the invocable action is in a managed package, *`namespace__`* is a required prefix. Otherwise, it's optional. To use a custom icon for an invocable action in a managed package, declare the method that defines the invocable action as `global`.

For SLDS icons, specify the name as `slds:`***`category`***`:`***`name`***. For example:

```
public class CustomSldsIcon {

@InvocableMethod(iconName='slds:standard:choice')
public static void run() {}

}
```

- *`category`* is the name of the category of the icon.
- *`name`* is the name of the file in SLDS.

When a user adds the invocable action in Flow Builder, the custom icon appears on the Flow Builder canvas to represent it.

SEE ALSO:

Define Static Resources

Apex Developer Guide: InvocableMethod Annotation

Extend Flows with Lightning Components

To build a richer flow screen, add Aura components or Lightning web components to your flow. To perform an action without going through the server, add Aura components to your flow.

Build Rich Screens with Custom Screen Components

Use screen components to unlock the look, feel, and functionality of your flow screens.

Let Flows Execute Local Browser Actions

With local actions, you can perform actions in the browser rather than going through the Salesforce server.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Build Rich Screens with Custom Screen Components

Use screen components to unlock the look, feel, and functionality of your flow screens.

You can install screen components from AppExchange or have a developer build Lightning components for you. With custom screen components, you can do these actions and more:

- Customize the navigation in your flow
- Display data in a table
- Use a branded header instead of the default flow header

Note: Before you add custom screen components to a flow screen, make sure that:

- The flow's type is Screen Flow.
- Users can run the flow only in Lightning runtime. For example, don't distribute the flow using a Visualforce component.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When you have the appropriate screen component in your org, add it to your flow screen.

1. In Flow Builder, open an existing screen, or add a Screen element to the canvas.
2. From the list of screen components, drag the appropriate component onto the screen canvas.

   If a screen component is built from a Lightning component, it has a lightning bolt icon next to its name. If the custom screen component includes an API name and label, the label is shown on the screen canvas.
3. Use values from the flow to set the screen component's attributes.
4. To use values from the screen component as resources later in the flow, start typing the API name of the screen component.

   We automatically store all the attribute values in a variable with the same name as the screen component's API name.

   To store the action's output values manually, select **Manually assign variables (advanced)**.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

SEE ALSO:

Available Screen Components

Lightning Web Components Dev Guide: Configure a Component for Flow Screens

Lightning Aura Components Developer Guide: Customize Flow Screens Using Aura Components

Extend Flows with Lightning Components

Let Flows Execute Local Browser Actions

With local actions, you can perform actions in the browser rather than going through the Salesforce server.

Either install a local action from an external library, such as Unofficial SF, or have a developer build one for you. For example, with local actions, a flow can:

- Open a related article in another window or tab.
- When the flow creates a record, open the record in another browser tab.
- When the flow finishes, close the browser or console tab.
- Get data from a database behind your firewall.

Note: To add local actions to a flow, make sure that:

- The flow’s type is Screen Flow.
- Users can run the flow only in Lightning runtime. For example, don’t distribute the flow using a Visualforce component.

To add a local action to a flow:

1. Add an Action element to the canvas.
2. In the `Action` field, search for and select the local action.
3. To pass data between the flow and the local action, set the action’s inputs and store its outputs.

SEE ALSO:

Lightning Aura Components Developer Guide: Create Flow Local Actions Using Aura Components

Extend Flows with Lightning Components

Add and Edit Elements

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Testing Your Flow

Before you activate a data cloud-triggered flow or a record-triggered flow, test it to quickly verify its expected results and identify flow run-time failures.

Automated testing for flows saves you time and improves the flow's reliability. When you debug a flow, you manually configure the debug parameters and inputs each time you start to debug. With a flow test, you configure the test parameters and inputs one time. Each time you run the test, it uses the same configuration to evaluate the flow. Also, you're not stuck with the same test per flow. You can edit the test or create more tests for different scenarios. We recommend that you create a test for every path that the flow can take.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To run a flow in Flow Builder or from the flow detail page:

- Manage Flow

To open, edit, or create a flow test in Flow Builder:

- Manage Flow

To view test run details in Flow Builder:

- View All Data

Flow Test Limits and Limitations

Understand how many tests you can create per flow and the limitations of a flow test.

Limits

The maximum number of tests per flow is 200.

Limitations

- Flow tests are available only for record-triggered and data cloud-triggered flows.
- Flow tests don't support flows that run when a record is deleted.
- Flow tests don't support flow paths that run asynchronously.
- Flow tests don't count towards flow test-coverage requirements.
- Formulas aren't supported for setting test data. Use fixed values. For example, if a flow test that verifies a date field is set to the relative date `Today`, you must update the field manually before running the test. For example, if you create and run the test on August 3, 2022, the date is set to the relative date `August 3, 2022`. If you run the same test tomorrow, the value of the date field is still `August 3, 2022`.

Test a Flow

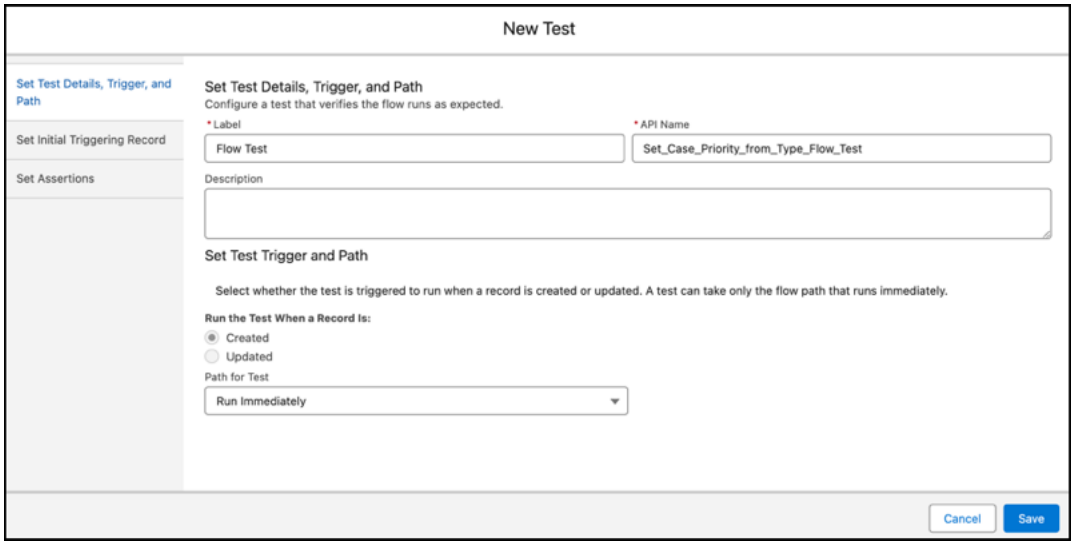
In Flow Builder, you can create, save, and run flow tests only for data cloud-triggered flows and record-triggered flows. Each time you modify the flow, you can run the test.

We recommend that you test all possible paths through the flow so that you can find errors before they affect your users. For example, if users or inputs don't provide all the data that your flow requires, the flow fails. Add fault connectors to provide paths for users or the flow logic to correct the data so that the flow can finish successfully.

1. Open Flow Builder.
   - From Setup, in the Quick Find box, enter *`Flows`*, and then select **Flows**.
   - From the Automation Lightning app, select the **Flows** tab.
   - From the **Flows** tab in any Lightning app.
2. Open the flow version that you want to test.
   - From the flow list screen in Setup, click ▾ for the flow that you want to test, select **View Details and Versions**, and then click **Open**.

- From the flow list screen in the Automation or any Lightning app, click the flow’s **Related** tab. For the flow version that you want to test, click ▾ , and then select **Open Flow**.

3. Click **View Tests**, and then click **Create**.
4. In the Set the Test Details, Trigger, and Path window, configure the test.
   a. Enter a label, API name, and description for the test.
   b. For Run the Test When a Record Is, select **Created** or **Updated**.


New Test
Set Test Details, Trigger, and Path
Set Initial Triggering Record
Set Assertions
Cancel
Save

Set Test Details, Trigger, and Path

Configure a test that verifies the flow runs as expected.

* Label: Flow Test

* API Name: Set_Case_Priority_from_Type_Flow_Test

Description

Set Test Trigger and Path

Select whether the test is triggered to run when a record is created or updated. A test can take only the flow path that runs immediately.

Run the Test When a Record Is:

● Created

○ Updated

Path for Test: Run Immediately

5. Click **Set Initial Triggering Record**, and enter the values for the record that initially triggers the test.
6. If the test is triggered to run when a record is updated, click **Set Updated Trigger Record**, and then enter the record values.
7. Click **Set Assertions**, and set conditions and custom failure messages for each assertion.
8. Save your changes, and then click **View Tests**.

   The owner of the flow changes to the last person who edits the test.

9. Select the test that you configured, and then click **Run**.

   Flow Builder runs the test and shows the result in the Result column.

Interpret Flow Test Results

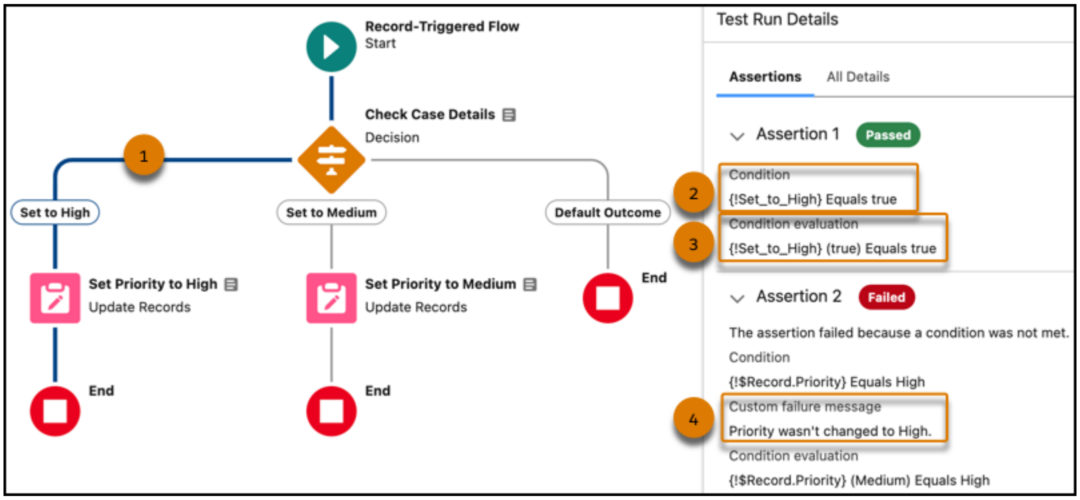
Understand how to interpret the flow test results after the flow test run completes.

We recommend interpreting the results after testing a flow to ensure that the flow operates as intended under various conditions.

1. To view the test run details, click ▾ , and select **Run Test and View Details**.

   The All Details tab under Test Run Details shows the results for the entire test run. If a flow fails, troubleshoot the flow errors.

2. To check the status of the flow tests, under Test Run Details, select the **Assertions** tab, and then expand each assertion.



Test Run Details

Assertions All Details

Assertion 1 Passed

Condition
{!Set_to_High} Equals true

Condition evaluation
{!Set_to_High} (true) Equals true

Assertion 2 Failed

The assertion failed because a condition was not met.

Condition
{!$Record.Priority} Equals High

Custom failure message
Priority wasn't changed to High.

Condition evaluation
{!$Record.Priority} (Medium) Equals High

The Flow Builder highlights the path of the test run (1). In the Test Run Details, each assertion tells you what condition it was looking for (2), the values that those conditions were checked against (in parentheses) (3), and the custom error message that you defined for that assertion (4).

3. Interpret the failed flow test.

   a. Check the condition and the condition evaluation of the failed assertion.

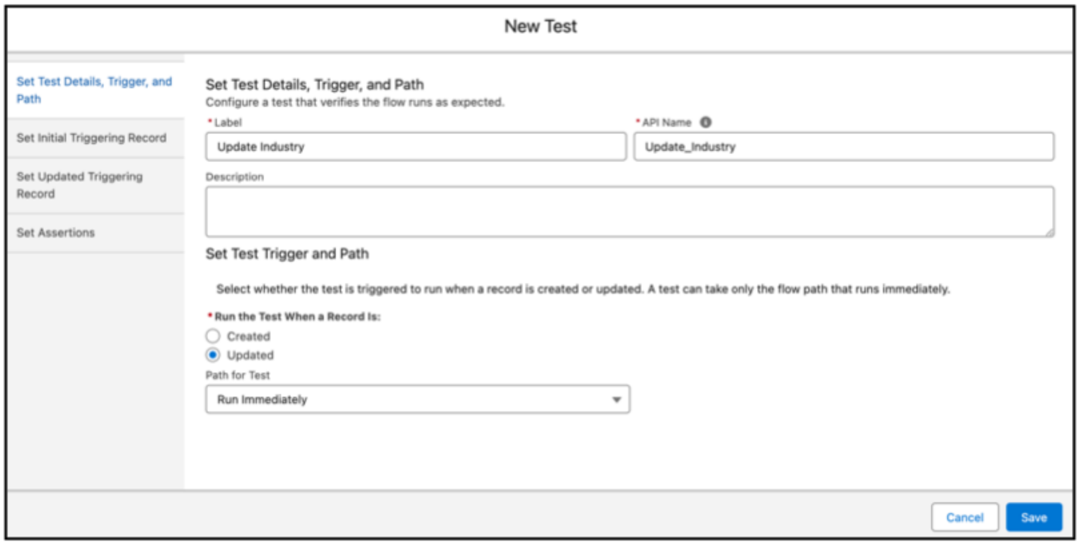
      In this example, for the Set Priority to High element, the failed assertion looks for the value of the field $Record's Priority to be set to High.

   b. To change the value, click the element, and then click **Edit Element**.

      In this example, to fix the failed assertion, edit the Set Priority to High element and change the priority for $Record to High.

   c. Save the flow, and then run the flow test that failed.

Example: This test verifies the flow when an object is updated and the flow takes the path that runs immediately.


New Test
Set Test Details, Trigger, and Path
Set Initial Triggering Record
Set Updated Triggering Record
Set Assertions
Cancel
Save

Set Test Details, Trigger, and Path

Configure a test that verifies the flow runs as expected.

*Label: Update Industry

*API Name: Update_Industry

Description:

Set Test Trigger and Path

Select whether the test is triggered to run when a record is created or updated. A test can take only the flow path that runs immediately.

*Run the Test When a Record Is:

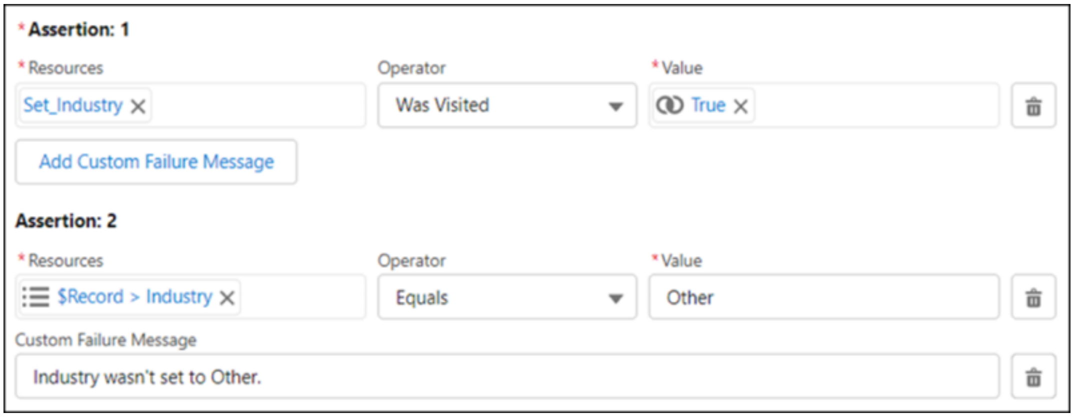
○ Created

● Updated

Path for Test: Run Immediately

The test includes fields to set the initial and updated values for the triggering record, which is the same record that triggers the flow to run. The test creates a copy of the record to use only for the test. The record isn't saved to the database.

The test evaluates each assertion to verify that the flow runs as expected. A test can evaluate only whether a flow element was executed and whether flow resource values are set as expected. The test evaluates the flow based on the existing data in your Salesforce database and customizations such as rules and restrictions.



*Assertion: 1

*Resources	Operator	*Value
Set_Industry ✕	Was Visited	True ✕

Add Custom Failure Message

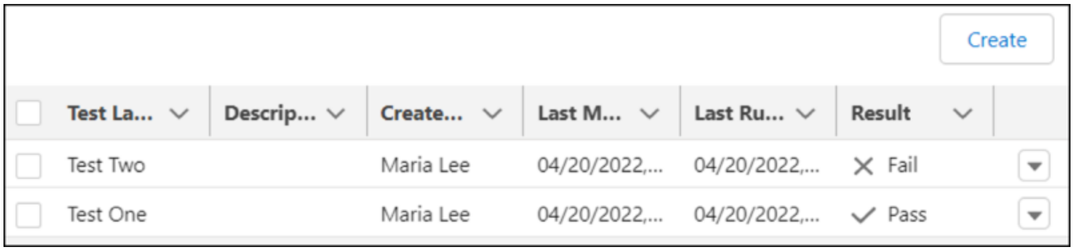
Assertion: 2

*Resources	Operator	*Value
$Record > Industry ✕	Equals	Other

Custom Failure Message

Industry wasn't set to Other.

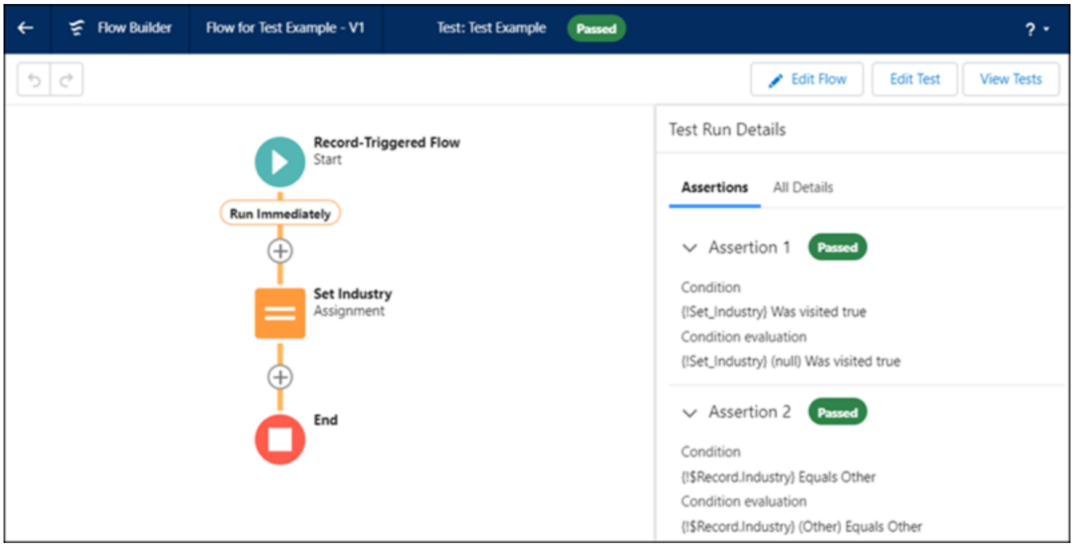
When a flow includes tests, the Tests list view shows all tests for a flow and associated test results.



Create

□	Test La...	Descrip...	Create...	Last M...	Last Ru...	Result
□	Test Two		Maria Lee	04/20/2022,...	04/20/2022,...	✕ Fail
□	Test One		Maria Lee	04/20/2022,...	04/20/2022,...	✓ Pass

After a test is run, Test Run Details shows how each assertion was evaluated. If a condition evaluates to false, the assertion and test fail. If all assertions pass, the test passes.



Test Run Details

Assertions All Details

Assertion 1 Passed

Condition

{!Set_Industry} Was visited true

Condition evaluation

{!Set_Industry} (null) Was visited true

Assertion 2 Passed

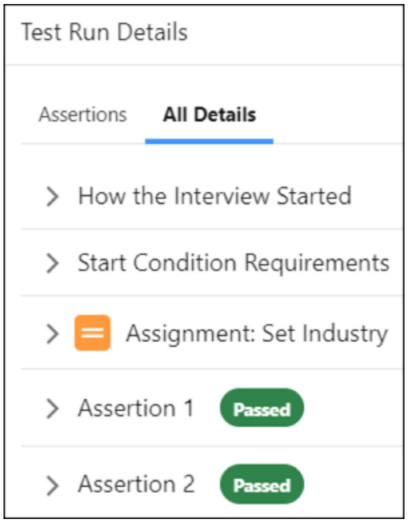
Condition

{!$Record.Industry} Equals Other

Condition evaluation

{!$Record.Industry} (Other) Equals Other

The All Details tab shows the results for the entire test run. Assertions are evaluated at the end of the test run.


Test Run Details

Assertions **All Details**

- How the Interview Started
- Start Condition Requirements
- Assignment: Set Industry
- Assertion 1 Passed
- Assertion 2 Passed

When you're confident that your flow is working as expected, activate the version that you tested and distribute the flow.

SEE ALSO:

Trailhead: Create Flow Tests

Troubleshoot Flow Errors

Distribute a Flow

After you've designed and tested your flow, it's time to put it to work! Flows can be executed in several ways, depending on who the flow is designed for. Internal users, external users, or systems can run a flow, or a flow can be deployed for another organization.

Activate or Deactivate a Flow Version

You can have multiple versions of a flow in Salesforce, but only one version of each flow can be active at a time. You can activate or deactivate a flow right in Flow Builder or from the flow's detail page in Setup.

Flow Distribution Concepts

Understand which users can run your flow, what data your flow can access, and how your flow looks in Classic and Lightning Experience.

Distribute Flows to Users in Your Org

Enable your internal users to run your flow through a custom action, the flow URL, a Lightning page, a Visualforce page, or a custom Aura component.

Distribute Flows to Users Outside Your Org

Let external users run your flow by adding the flow to an Experience Builder site, an external app or page, or an Embedded Service deployment. For finer control over how your flow behaves in external contexts, use a custom Aura component or Visualforce page. Flows in custom Aura components use Lightning runtime, and flows in Visualforce pages use Classic runtime.

Distribute Flows to Automated Systems

Some flows don't require any user interaction to start. To enable a system to automatically launch a flow, use the `start` Apex method, a process, or a workflow action.

Distribute Flows to Other Orgs

Flows can be included in Lightning Bolt Solutions, change sets, and packages. The recipient org of the solution, change set, or package must have flows enabled.

Activate or Deactivate a Flow Version

You can have multiple versions of a flow in Salesforce, but only one version of each flow can be active at a time. You can activate or deactivate a flow right in Flow Builder or from the flow's detail page in Setup.

When you activate a flow version, the previously activated version (if one exists) is deactivated. Any running flow interview continues to run using the version that it started with.

If you activate your flow in a sandbox, when you move it to production the **Activate** button is disabled until you save the flow as a new version. Since sandbox orgs and production orgs can have differences including properties and IDs, after you move your flow from a sandbox to production, always debug on page 234 the flow using rollback mode.

1. Open Flow Builder.
    - From Setup, in the Quick Find box, enter *`Flows`*, and then select **Flows**.
    - From the Automation Lightning app, select the **Flows** tab.
    - From the **Flows** tab in any Lightning app.
2. Open the flow version you want to activate or deactivate.
3. Activate or deactivate your flow.
    - To activate your flow, click **Activate** in the button bar.
    - To deactivate your flow, click **Deactivate** in the button bar.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To activate or deactivate a flow:
- Manage Flow

To activate an autolaunched flow that has a trigger:
- View All Data

SEE ALSO:

Distribute a Flow

Triggers for Autolaunched Flows

Flow Distribution Concepts

Understand which users can run your flow, what data your flow can access, and how your flow looks in Classic and Lightning Experience.

Limit User Access to Execute Flows

Limit which users can execute flows based on their user record, profile, or permission set. To apply more granular security to an individual flow, override the default behavior, then grant access to that specific flow based on user profile or permission set.

Flow Run Context

Flows run in user context or system context. For a flow running in user context, the running user's profile and permission sets determine the object permissions and field-level access of the flow. For a flow that runs in system context, the flow access is determined by whether the flow runs in system context with sharing or without sharing.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning Runtime vs. Classic Runtime for Flows

Depending on how a flow is distributed, users see either the Classic runtime or Lightning runtime UI when they run the flow. Like its name suggests, Lightning runtime looks and feels like Lightning Experience.

Limit User Access to Execute Flows

Limit which users can execute flows based on their user record, profile, or permission set. To apply more granular security to an individual flow, override the default behavior, then grant access to that specific flow based on user profile or permission set.

Configure User Profile or Permission Set Access to a Flow

For individual flows that you create or clone, you can override default behavior and restrict access to users with enabled profiles or permission sets only.

Configure Guest User Profile Access to a Flow with Flow Interview Sharing Rules

By default, only the user who started the flow interview can resume it. To grant access to a flow interview to someone other than the user who started it, use flow interview sharing rules. You can control access to specific flows for the Guest User profile only through the profile page.

SEE ALSO:

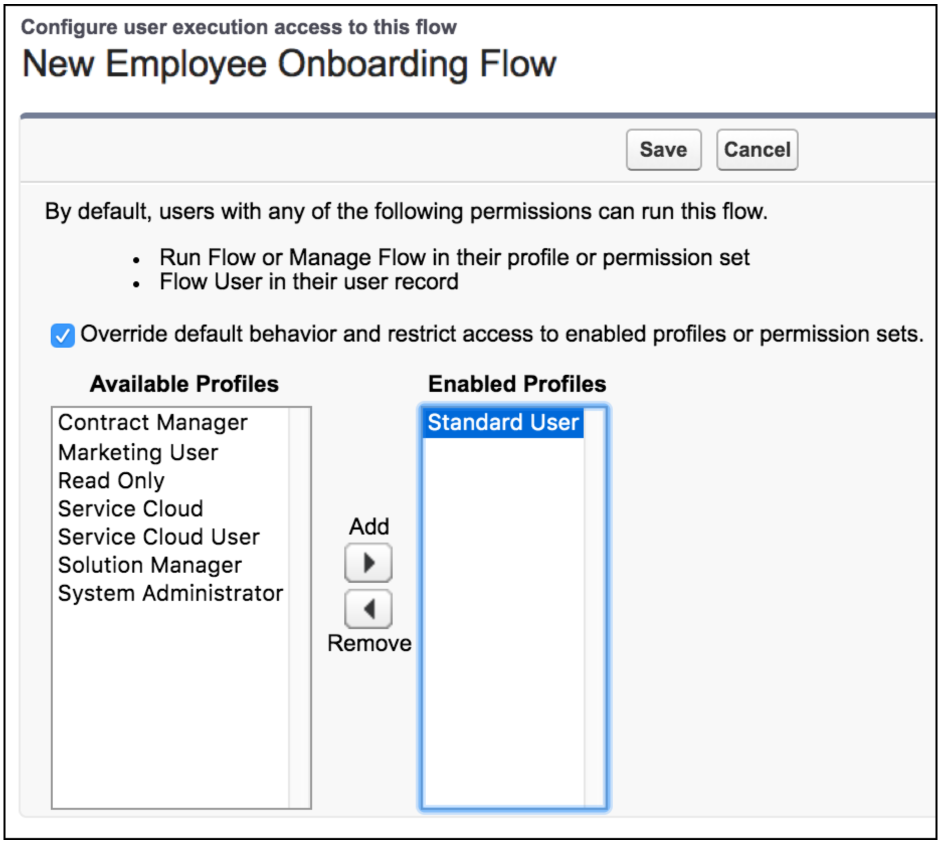
Data Safety When Running Screen and Autolaunched Flows in System Context

Configure User Profile or Permission Set Access to a Flow

For individual flows that you create or clone, you can override default behavior and restrict access to users with enabled profiles or permission sets only.

If you select Override default behavior and restrict access to enabled profiles or permission sets, the resuming user must have permission to the individual flow also. To override the default behavior and restrict access, follow these guidelines.

1. From Setup, in the Quick Find box, enter `Flows`, and then select **Flows**.
2. From the actions menu ▼ for the flow that you want to edit, click **Edit Access.**
3. In Apps, click **Flow Access**.
4. Click **Edit**.
5. Select **Override default behavior and restrict access to enabled profiles or permission sets**.
6. Add the available profile to the enabled profiles.
7. Save your work.



Configure user execution access to this flow

New Employee Onboarding Flow

By default, users with any of the following permissions can run this flow.

- Run Flow or Manage Flow in their profile or permission set
- Flow User in their user record

☑ Override default behavior and restrict access to enabled profiles or permission sets.

Available Profiles: Contract Manager, Marketing User, Read Only, Service Cloud, Service Cloud User, Solution Manager, System Administrator

Enabled Profiles: Standard User

Configure Guest User Profile Access to a Flow with Flow Interview Sharing Rules

By default, only the user who started the flow interview can resume it. To grant access to a flow interview to someone other than the user who started it, use flow interview sharing rules. You can control access to specific flows for the Guest User profile only through the profile page.

You can control access to specific flows for the Guest User profile through the profile page only.

Permission to access a flow is checked only at the entry points, such as when a flow starts or resumes.

Permission for a flow is checked only at the top level. For example, flow A calls flow B. User X has a profile that can access flow A but not flow B. User X can execute flow B, but only through flow A. User X can't execute flow B directly.

1. From Experience Builder for the site associated with the Guest User profile, go to **Settings** > **General**.
2. Under Guest User Profile, click the profile name.
3. Click **Enabled Flow Access** > **Edit**.
4. Add or remove flows from the Guest User Profile.
5. Save your work.

SEE ALSO:

Sharing Rules

Flow Run Context

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flows run in user context or system context. For a flow running in user context, the running user's profile and permission sets determine the object permissions and field-level access of the flow. For a flow that runs in system context, the flow access is determined by whether the flow runs in system context with sharing or without sharing.

The context that the flow runs in impacts what the following flow elements can do with Salesforce data:

- Action
- Create Records
- Delete Records
- Get Records
- Subflow
- Update Records
- Any other flow element that accesses fields from a record

By default, a flow runs in user context or system context, depending on how the flow is launched. When a flow runs in system context, it runs in that context for all users, including users with the Guest User profile.

Flow Launch Method	Default Context
Apex	Depends on code
Experience Cloud site	User
Embedded as a visual component inside a custom Aura component	User
Embedded as a visual component inside a Visualforce page	User
Custom button	User
Custom link	User
Direct link	User
Flow action	User
Lightning page	User
Platform event	System context without sharing
Process built in Process Builder	System context without sharing
Record-triggered	System context without sharing
Rest API	User
Run from an Apex method of a custom Aura component controller	Depends on code
Run from an Apex method of a Visualforce controller	Depends on code
Schedule-triggered	System context without sharing
Web tab	User

Running User of a Flow

The running user of a flow is the user that launched the flow. The running user determines what a flow that runs in user context can do with Salesforce data.

For a flow running in user context, the running user's profile and permission sets determine the object permissions and field-level access of the flow. When a flow attempts to create, read, edit, or delete Salesforce data, it enforces the running user's permissions and field-level access. For example, if the running user doesn't have the edit permission for the Account object, and the flow attempts to update account records, an error occurs. If the running user doesn't have permission to edit the Rating field on the Account object, and the flow attempts to update that field, an error occurs.

Org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories also impact what data is accessible to flows that run in user context. If the org-wide default of the Opportunity object is private, and no opportunity records have been shared with the running user, the flow can't read or edit any opportunity records.

Limitations

- Lightning components such as screen components and local actions always run in user context when retrieving data directly through LWC supported APIs.
- If the context depends on code, Apex uses the `with sharing` and `without sharing` keywords to specify whether to enforce org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories. A flow called by Apex always ignores object and field-level access permissions.
- When a record-triggered flow, schedule-triggered flow, or process invokes an Apex invocable method from an Apex class with the inherited sharing declaration, the invocable method runs in system context with sharing. The flows and process run in system context without sharing.
- If a process triggers the flow to launch, the user that triggered the process sometimes requires other permissions. For example, if a process launches a flow that attempts to save permission set license assignments, and the running user doesn't have the Assign Permission Sets permission, an error occurs.
- A flow always runs in user context when it executes the Post to Chatter action.

Data Safety When Running Screen and Autolaunched Flows in System Context

Screen flows and autolaunched flows with no trigger can sometimes share more data than intended. Screen flows on Experience Cloud sites are primarily impacted because external users access them, but other screen flows and autolaunched flows with no trigger can be affected, too. Take the time to configure user permissions for field-level and record-level security, instead of running flows in system context as a shortcut. If you must run a screen flow or autolaunched flow with no trigger in system context, use the minimum number of fields and records needed to get the job done.

Change the Flow Run Context

You can set some flow versions to always run in system context, overriding the context it normally runs in. If you choose to run the flow in system context with sharing, the flow respects org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories. But it doesn't respect object permissions, field-level access, or other permissions of the running user.

SEE ALSO:

Change the Flow Run Context

Data Safety When Running Screen and Autolaunched Flows in System Context

Screen flows and autolaunched flows with no trigger can sometimes share more data than intended. Screen flows on Experience Cloud sites are primarily impacted because external users access them, but other screen flows and autolaunched flows with no trigger can be affected, too. Take the time to configure user permissions for field-level and record-level security, instead of running flows in system context as a shortcut. If you must run a screen flow or autolaunched flow with no trigger in system context, use the minimum number of fields and records needed to get the job done.

To control what data your users can access, follow these guidelines and suggested workarounds.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Restrict User Access to Data in Screen Flows

Run screen flows in user context whenever possible because it enforces permissions that restrict user access to data. When you run a screen flow in system context, especially system context without sharing, it doesn't enforce permissions for users so it increases the chance of unintended data sharing. There are other, safer ways to grant access to users without running a flow in system context.

Instead of setting the entire screen flow to run in system context, use the Subflow element to launch a flow that performs the actions that require permissions beyond what the running user has.

Example: Let's say that you want to assign a permission set to a user as part of a screen flow, which requires permissions the running user doesn't have. Build an autolaunched flow with no trigger that runs in system context and assigns the permission set to the user. Then, build your screen flow running in user context and add a Subflow element to launch the autolaunched flow.

Store Specific Fields in Get Records Elements

When you override an autolaunched flow with no trigger or screen flow to run in system context and use a Get Records element to pass records into a screen component, action, or another flow, specify which fields to store by selecting **Choose fields and let Salesforce do the rest** instead of selecting **Automatically store all fields**. When passing in the entire record variable or record collection from a Get Records element to a screen component, action, or another flow, the flow doesn't know what fields are needed, so it retrieves all the fields the user can access. When you select **Choose fields and let Salesforce do the rest** and specify fields, Salesforce stores only the fields you need, reducing the number of fields that are exposed to users.

Example: In a screen flow, when getting records for a record collection used in a Data Table component, in the associated Get Records element, store only the fields that you want to show in the Data Table component or reference elsewhere on a screen.

Specify Fields and Records in Flows on Experience Cloud Sites

When you run in system context and use a Create Records, Update Records, or Delete Records element in a screen flow on an Experience Cloud site, limit the number of fields and records only to those that you intend the user to edit.

- For Create Records elements, for How to set record field values, select **Manually** and specify each field to use when creating the record.
- For Update Records elements, for How to Find Records and Set Their Values, select **Specify conditions to identify records, and set fields individually** and use filter conditions to find the records to update and then specify the fields to update.
- For Delete Records elements, for How to Find Records to Delete, select **Specify conditions** and use filter conditions to find the records to delete.

If you must use a record variable or record collection in a Create Records, Update Records, or Delete Records element, use a Transform element first to filter out the fields that you don't want edited. See Transform Data in a Flow. Then, in a Create Records, Update Records, or Delete Records element, select the record variable or record collection that the Transform element generates.

Review Screen Actions Triggered by Action Button Components

When using an Action Button component to trigger a screen action, follow these recommendations.

- Check the flow run context of the autolaunched flow that's called by an action button. The autolaunched flow can run in a different context than your screen flow, potentially exposing more data than you want. Consider keeping the autolaunched flow running in user context. If the autolaunched flow must run in system context, use the previous suggestions to modify the flow and keep your data safe.
- Use an autolaunched flow called by an action button only to find data or perform calculations. If you must create, update, or delete records, use a Create Records, Update Records, or Delete Records element in the screen flow or a different flow that's launched by a Subflow element.
- Reference a screen action only in the Screen element where the screen action is configured. If you must reference the screen action's output somewhere else in the flow, run the screen action's autolaunched flow again in a Subflow element and reference the Subflow element's outputs instead.

Check Inputs and Outputs of Flows Launched by Subflow Elements

Follow these guidelines when you configure a flow launched by a Subflow element that's running in system context.

- Make variables available for input and output only if you plan on using them in the flow that launches the referenced flow. Unused input and output variables can expose users to unnecessary data.
- Avoid passing in inputs to a referenced flow that are easy to guess. For example, if you pass in a company email address of a record, and the company email addresses follow the pattern `firstname.lastname@company`.com, it's easy to guess another record's email address and have access to that other record.
- Make sure a referenced flow includes only output values that are directly used in the flow that launched it. For example, if you only need a collection of contact record birthdays, don't also return all of the information about the contacts or the contact's accounts.

SEE ALSO:

Flow Run Context

Salesforce Help: Securely Share Your Experience Cloud Sites with Guest Users

Medium Blog for Salesforce Architects: Building Secure Screen Flows For External User Access

Change the Flow Run Context

You can set some flow versions to always run in system context, overriding the context it normally runs in. If you choose to run the flow in system context with sharing, the flow respects org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories. But it doesn't respect object permissions, field-level access, or other permissions of the running user.

1. Open the flow in Flow Builder.
2. Click ⚙.
3. Click **Show Advanced**.
4. In the How to Run the Flow dropdown, select a context.

   > Note: If the dropdown isn't available, you can't change the flow run context for the type of flow.

   - User or system context—The context is determined by how the flow is launched.
   - System Context with Sharing—The flow respects org-wide default settings, role hierarchies, sharing rules, manual sharing, teams, and territories. But it doesn't respect object permissions, field-level access, or other permissions of the running user.
   - System Context without Sharing—The flow can access all data.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:
- Manage Flow

SEE ALSO:

Flow Run Context

Lightning Runtime vs. Classic Runtime for Flows

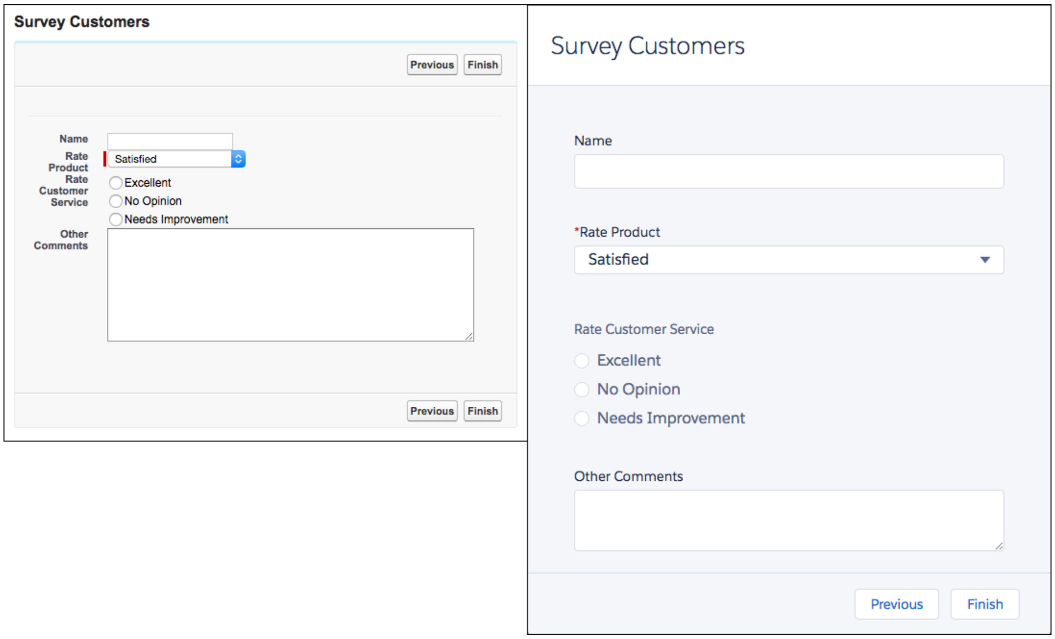
Depending on how a flow is distributed, users see either the Classic runtime or Lightning runtime UI when they run the flow. Like its name suggests, Lightning runtime looks and feels like Lightning Experience.

Here's the same flow rendered in Classic runtime (left) and Lightning runtime (right).

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Survey Customers

Previous Finish

Name

Rate Product: Satisfied

Rate Customer Service: ○ Excellent ○ No Opinion ○ Needs Improvement

Other Comments

Previous Finish

Survey Customers

Name

*Rate Product: Satisfied

Rate Customer Service

○ Excellent

○ No Opinion

○ Needs Improvement

Other Comments

Previous Finish

Which Runtime Experience Do My Users See?

Flows that run from a Visualforce component always use Classic runtime. Flows that run from a Lightning page, flow action, or custom Aura component always use Lightning runtime. All other methods depend on whether Lightning runtime has been enabled in your org's Process Automation settings.

This table summarizes which runtime experience your users see based on how you distribute the flow.

Flow Distribution Method	When Lightning Runtime for Flows is	
	Not selected	Selected
Visualforce component	Classic runtime	Classic runtime
Custom button	Classic runtime	Lightning runtime
Custom link	Classic runtime	Lightning runtime
Web tab	Classic runtime	Lightning runtime
Direct link	Classic runtime	Lightning runtime
Flow action	Lightning runtime	Lightning runtime
Lightning page	Lightning runtime	Lightning runtime
Custom Aura component	Lightning runtime	Lightning runtime

SEE ALSO:

Set the Runtime Experience for URL-Based Flows

Distribute Flows to Users in Your Org

Enable your internal users to run your flow through a custom action, the flow URL, a Lightning page, a Visualforce page, or a custom Aura component.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Add a Flow to a Lightning Page

To easily distribute a flow to Lightning Experience or Salesforce app users, embed it in a Lightning page.

Create an Object-Specific Quick Action That Launches a Flow

To easily distribute a flow to Lightning Experience or Salesforce app users, create a flow action and add it to the appropriate page layout. Flows aren't supported for global actions.

Add a Flow to the Actions & Recommendations Component

Want to use your flow to guide users through complex business processes in Lightning console or standard navigation apps? Associate the flow with records by using a process to create a RecordAction object, or by creating an Actions & Recommendations deployment. Then add the Actions & Recommendations component to your Lightning pages using the Lightning App Builder.

Add a Flow to a Utility Bar

Want your flow to be accessible from any page in your app? Add it to the utility bar in your Lightning app. The utility bar gives your users quick access to commonly used tools.

Share the Flow URL or Create a Custom Button, Link, or Web Tab

Users in your org who don't need a customized look and feel can run the flow via its URL. Distribute a flow URL directly or through a custom button, link, or web tab.

Embed a Flow in a Custom Aura Component

To customize how your flow gets and receives data, add it to a custom Aura component. Then distribute that component through a custom action, Lightning tab, or Lightning page.

Embed a Flow in a Visualforce Page

To customize your flow's look and feel for internal users, add the flow to a Visualforce page. Then distribute that page through a Visualforce tab, custom button, or custom link.

Prepare Your Org for Paused Flow Interviews

A *flow interview* is a running instance of a flow. Not every flow interview can be completed in one go. Add the Pause button to your flows, so that users can pause flow interviews for later. Update the sharing model for flow interviews, so that other users can resume a paused interviews. And make it easy for users to resume interviews by adding a component to their Home page.

Add a Flow to a Lightning Page

To easily distribute a flow to Lightning Experience or Salesforce app users, embed it in a Lightning page.

Available in: Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create and save Lightning pages in the Lightning App Builder:

- Customize Application

If you're not yet familiar with the types of Lightning pages you can customize, check out the Lightning App Builder module in Trailhead. If your org uses the Outlook or Gmail integrations you can also create custom email application panes.

1. Open a Lightning page in the Lightning App Builder.
2. From the Lightning Components pane on the left, drag the Flow component onto the Lightning page canvas.
3. Configure the component.

Component	Description
Flow	Only active screen flows are available. Flows that were built in the Desktop Flow Designer aren't supported.
Layout	By default, flows display in one column. If a screen contains a Section screen component, that screen ignores the Layout property. • Starting in Winter '23, two-column flow layouts are ignored. For a better layout option, add Section components to your flow screens. Each Section component lets you organize record fields and screen components in up to four adjustable-width columns. • If a flow screen contains a Section component, that screen ignores the Layout property.
Input variables	If you see other properties, they're the flow's input variables. Variables appear only if they allow input access.
Pass record ID into this variable	This option is available only for Text input variables in Record pages. For simplicity, we recommend passing the ID to only one variable. For example, when this component is embedded in an Opportunity Record page, at run time the component passes the opportunity's ID into the selected input variable.

4. Save the page.
5. Hang on, you're not finished yet! To make your page available to your users, activate it. You can activate the page from the Save dialog when you save it for the first time, or later using the **Activation** button.
6. Test that the flow is working correctly, and then roll the Lightning page out to your users.

SEE ALSO:

Two-Column Flow Considerations

Flow Screen Output Component: Section

Create an Object-Specific Quick Action That Launches a Flow

To easily distribute a flow to Lightning Experience or Salesforce app users, create a flow action and add it to the appropriate page layout. Flows aren't supported for global actions.

Available in: Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create actions:
- Customize Application

1. From the management settings for the object for which you want to create an action, go to Buttons, Links, and Actions.
2. Click **New Action**.
3. For Action Type, select **Flow**.
4. Select the flow to use in your action.

   The flow must be active and of type "Screen Flow" or "Field Service Mobile Flow".
5. Enter a label for the action.

   Users see this label, rather than the flow name, as the name of the action. We recommend entering the name of the flow as the action label.
6. If necessary, change the name of the action.
   This name is used in the API and managed packages. It must begin with a letter and use only alphanumeric characters and underscores, and it can't end with an underscore or have two consecutive underscores. Unless you're familiar with working with the API, we suggest not editing this field.
7. Type a description for the action.
   The description appears on the detail page for the action and in the list on the Buttons, Links, and Actions page. The description isn't visible to your users. If you're creating several actions on the same object, we recommend using a detailed description.
8. Optionally, change the action icon to a static resource in your org.

   Custom images used for action icons must be less than 1 MB in size.
9. Save the action.
10. Hang on, you're not done yet! To make your action available to your users, add it to a page layout.

Want the action to send the record's ID to your flow? Make sure that the flow has a variable with these settings.

Variable Setting	Value
API Name	`recordId` (case sensitive)
Data Type	Text
Available for input	Selected

Note: If you delete an action, the action is removed from all layouts that it's assigned to. If you deactivate a flow referenced in an action, the action doesn't appear at run time.

Add a Flow to the Actions & Recommendations Component

Want to use your flow to guide users through complex business processes in Lightning console or standard navigation apps? Associate the flow with records by using a process to create a RecordAction object, or by creating an Actions & Recommendations deployment. Then add the Actions & Recommendations component to your Lightning pages using the Lightning App Builder.

When users open a record with this component, they can launch flows and other actions from the list. The flows start in a subtab in a console app, or in a popup window in a standard navigation app. The Actions & Recommendations component is great for call scripts or chat interactions.

SEE ALSO:

Lightning Flow for Service Developer Guide (English only)

Add a Flow to a Utility Bar

Want your flow to be accessible from any page in your app? Add it to the utility bar in your Lightning app. The utility bar gives your users quick access to commonly used tools.

Available in: Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

1. From Setup, in the Quick Find box, enter *App*, then select **App Manager**.
2. Edit an existing Lightning App or click **New Lightning App**. You can also upgrade a custom Classic App to a Lightning app.
3. Under App Settings, click **Utility Items**.
4. Click **Add Utility Item**, then select **Flow**.
5. Configure the utility item properties and the component properties.
   - Flow
   - Layout

   Tip: If a Section component has three or more columns, give the flow's utility item enough space to display the flow's columns. To do so, increase the value of the Panel Width property on the flow's utility item.
6. Save your changes.

To verify your changes, from the App Launcher find and open the app that you added the flow to.

SEE ALSO:

Two-Column Flow Considerations

Flow Screen Output Component: Section

Share the Flow URL or Create a Custom Button, Link, or Web Tab

Users in your org who don't need a customized look and feel can run the flow via its URL. Distribute a flow URL directly or through a custom button, link, or web tab.

1. From Setup, enter `Flows` in the Quick Find box, then select **Flows**.
2. For the flow that you want to distribute, click ▾ and select **View Details and Versions**.

   If ▾ isn't available for each flow, click the name of the flow that you want to distribute.
3. Verify that the flow has an active version.

   Only users with the Manage Flow permission can run inactive flows. If the flow contains subflow elements, the referenced flows must also have an active version.
4. Copy the flow URL, and append it to your instance.
   For example:

   ```
   https://yourDomain.my.salesforce.com/flow/MyFlowName
   ```

   If the flow was installed from a managed package, include the namespace prefix in the flow URL. For example:

   ```
   https://yourDomain.my.salesforce.com/flow/namespace/MyFlowName
   ```
5. To set the initial values of your flow's variables, append `?variable1=value1&variable2=value2` to the URL.
6. Distribute the flow URL.
   Here are some examples:
   - Create a custom button or link, and add it to a page layout.
   - Create a web tab, and add it to the appropriate profiles.

Set the Runtime Experience for URL-Based Flows

Are you distributing a flow via a URL? That includes things like direct URLs, custom buttons, and links in Setup. You can flip one switch to upgrade all those flows to Lightning runtime.

Customize a Flow URL to Control Finish Behavior

By default, when a flow interview that uses screens finishes, a new interview for that flow begins, and the user is redirected to the first screen. If you want to redirect users to another page within Salesforce when they click **Finish**, use the `retURL` parameter in the flow URL.

Customize a Flow URL to Set Variable Values

When you distribute a flow using a URL, you can set variables within that flow using parameters in the URL.

Customize a Flow URL to Render Two-Column Screens

When you distribute a flow using a URL, you can control whether to display the screens with one column or two. Two-column screens are supported only for orgs that have enabled Lightning runtime.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view flow detail pages:
- Manage Flow

Set the Runtime Experience for URL-Based Flows

Are you distributing a flow via a URL? That includes things like direct URLs, custom buttons, and links in Setup. You can flip one switch to upgrade all those flows to Lightning runtime.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

User Permissions Needed	
To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

We have two flavors of runtime experience for your flow users. *Classic runtime* looks more like a standard Visualforce page. *Lightning runtime* fits right in with Lightning Experience. To compare the two runtime experiences, check out Lightning Runtime vs. Classic Runtime for Flows.

To render all URL-based flows in Lightning runtime:

1. From Setup, enter `Process Automation Settings` in the Quick Find box, then select **Process Automation Settings**.
2. Select **Enable Lightning runtime for flows**.
3. Save your changes.

This setting also controls whether a flow appears in one or two columns when you distribute the flow via a URL or via a Lightning page.

When enabled, flows use Lightning runtime when they're run from:

- A direct link
- A custom button or link
- Flow Builder
- Flow detail pages or list views

SEE ALSO:

Lightning Runtime vs. Classic Runtime for Flows

Customize a Flow URL to Render Two-Column Screens

Customize a Flow URL to Control Finish Behavior

By default, when a flow interview that uses screens finishes, a new interview for that flow begins, and the user is redirected to the first screen. If you want to redirect users to another page within Salesforce when they click **Finish**, use the `retURL` parameter in the flow URL.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Format

To redirect users to a specific page in Salesforce after they click **Finish**:

```
/flow/flowName?retURL=url
```

where *url* is a relative URL (the part of the URL that comes after https://*MyDomainName*.my.salesforce.com/ or https://*MyDomainName*.lightning.force.com/).

URL Options

You can't redirect flow users to a URL that's external to your Salesforce org.

Tip: Use Salesforce Classic URLs. Lightning Experience URLs always redirect to the home page in Lightning Experience.

- For Salesforce Classic URLs, Salesforce redirects your users to the right page in whichever Salesforce experience they've enabled — Lightning Experience or Salesforce Classic. If the page doesn't exist in Lightning Experience, Salesforce redirects the user to the page in Salesforce Classic.
- For Lightning Experience URLs, Salesforce always redirects your users to the home page in Lightning Experience (lightning/page/home), even if the user has Salesforce Classic enabled. Users who don't have permission to access Lightning Experience see an error message.
- If your URL redirects users to a web tab, Salesforce renders the web tab in Salesforce Classic.
- Web tabs in Lightning Experience can redirect only to Visualforce pages.

Redirect Destination	Relative URL	Example
Chatter	_ui/core/chatter/ui/ChatterPage	_ui/core/chatter/ui/ChatterPage
Home page	home/home.jsp	home/home.jsp
List view	*objectCode*?fcf=*listViewId*	006?fcf=00BD0000005Iwec
Object home page, such as Accounts home	*objectCode*/o	001/o
Specific record, such as a contact, report, dashboard, user, profile, or Chatter post	*recordId*	0D5B000000SKZ7V
Visualforce page	apex/*pageName*	apex/myVisualforcePage
Web tab	servlet/servlet.Integration?lid=*webTabId*	servlet/servlet.Integration?lid=01rD0000000A88h

Limitations

- You can't use a flow variable as the value for the retURL parameter. If you want to use a flow variable to redirect a user, such as to a specific record, distribute the flow by using Visualforce.
- retURL can cause nested top and side navigation bars to render on the destination page.
- retURL is case-sensitive. If you use retUrl, the URL doesn't redirect the user.

Examples

This flow URL redirects users to Accounts home, which exists in both Lightning Experience and Salesforce Classic.

```
/flow/myFlow?retURL=001/o
```

When Lightning Experience users finish the flow interview, Salesforce redirects them to http://***MyDomainName***.lightning.force.com/lightning/o/Account/home. When Salesforce Classic users finish the flow interview, Salesforce redirects them to http://***MyDomainName***.my.salesforce.com/001/o. Either way, Salesforce redirects users to Accounts home in their respective experience.

This flow URL redirects users to a Visualforce page that exists only in Salesforce Classic.

```
/flow/myFlow?retURL=apex/myPage
```

When users finish the flow interview, Salesforce redirects them to `http://`***`MyDomainName`***`.my.salesforce.com/apex/myPage` in Salesforce Classic. When they navigate away from the Visualforce page, Salesforce reverts to their original experience.

For instance, after viewing the Visualforce page, users navigate to the home page. For Lightning Experience users, Salesforce renders the Lightning Experience home page (`http://`*`MyDomainName`*`.lightning.force.com/lightning`). For Salesforce Classic users, Salesforce renders the Salesforce Classic home page (`http://`***`MyDomainName`***`.my.salesforce.com/home/home.jsp`).

This flow URL sets the *`varUserFirst`* and *`varUserLast`* variables (both of type Text) to the running user's `FirstName` and `LastName` field values. When the flow interview finishes, the user is redirected to the home page for whichever Salesforce experience is enabled.

```
/flow/User_Info?varUserFirst={!$User.FirstName}
   &varUserLast={!$User.LastName}&retURL=home/home.jsp
```

SEE ALSO:

- Distribute a Flow
- Troubleshooting Flow URLs
- Customize a Flow URL to Set Variable Values

Customize a Flow URL to Set Variable Values

When you distribute a flow using a URL, you can set variables within that flow using parameters in the URL.

You can't set the values for record variables and record collection variables using URL parameters. The variable must allow input access.

These steps assume you're using a relative URL in your org to direct users to the flow.

1. From Setup, enter *`Flows`* in the Quick Find box, then select **Flows**.
2. Click ▾ for the flow you want to customize and select **View Details and Versions**.
3. Copy the URL for the flow.
4. At the end of the flow URL, append `?`***`name`***`=`***`value`***, where *`name`* is the unique name of the variable or collection variable in the flow that you want to set and *`value`* is the value you want to set it to. If you want to set multiple variable values, append `?`***`name1`***`=`***`value1`***`&`***`name2`***`=`***`value2`*** to the end of the flow URL. If you want to set the values for multiple items in the same collection variable, append `?`***`name`***`=`***`value1`***`&`***`name`***`=`***`value2`***.
   For example:

   ```
   /flow/MyFlow?varNumber=100&varString=Hello
   ```

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view a flow's detail page:
- View Setup and Configuration

To create and edit custom buttons, links, and Web tabs:
- Customize Application

Here are the valid values for each flow variable and collection variable based on its data type.

Variable Type	Acceptable Values
Boolean	• Merge field of type Checkbox • True values: `true` or `1` • False values: `false` or `0`
Currency	Merge field of type Number or a numeric value
Date	Merge field of type Date or `YYYY-MM-DD`
DateTime	Merge field of type Date/Time or `YYYY-MM-DDThh:mm:ssZ`
Multi-Select Picklist	Merge field of any type or a string in this format: `value1; value2`
Number	Merge field of type Number or a numeric value
Picklist	Merge field of any type or a string
Text	Merge field of any type or a string

Note: When you distribute a flow, don't pass a currency field value from a Salesforce record into a flow currency variable with a URL parameter. When a currency field is referenced through a merge field (such as `{!Account.AnnualRevenue}`), the value includes the unit of currency's symbol (for example, $). Flow currency variables can accept only numeric values, so the flow fails at run time. Instead, pass the record's ID to a text variable with a URL parameter. Then in the flow, use the ID to look up that record's value for the currency field.

Example: The following example is a flow URL that is used in a custom button on a case page layout. When a user clicks that button, the flow launches with the varID variable (of type Text) set to the case record's `CaseNumber` field value.

```
/flow/Case_Management?varID={!Case.CaseNumber}
```

The following example sets the varUserFirst and varUserLast variables (both of type Text) to the running user's `FirstName` and `LastName` field values.

```
/flow/User_Info?varUserFirst={!$User.FirstName}&varUserLast={!$User.LastName}
```

The following example is a flow URL that is used in a custom button on a contact page layout. When a user clicks that button, the flow launches and adds text values from the contact as items in the {!collNames} text collection variable.

```
/flow/Contact_Info?collNames={!Contact.FirstName}&collNames={!Contact.LastName}
```

SEE ALSO:

Customize a Flow URL to Control Finish Behavior

Distribute a Flow

Troubleshooting Flow URLs

Customize a Flow URL to Render Two-Column Screens

When you distribute a flow using a URL, you can control whether to display the screens with one column or two. Two-column screens are supported only for orgs that have enabled Lightning runtime.

Note: Starting in Winter '23, two-column flow layouts are ignored, including this URL customization. For a better layout option, add Section components to your flow screens. Each Section component lets you organize record fields and screen components in up to four adjustable-width columns.

Prerequisites:

Enable Lightning runtime so that your flows respect the specified layout.

1. From Setup, go to Process Automation Settings.
2. Select `Enable Lightning runtime for flows`.

Format:

To display a flow's screens in two columns:

```
/flow/flowName?flowLayout=twoColumn
```

Examples:

This example displays a "Case Management" flow in two columns.

```
/flow/Case_Management?flowLayout=twoColumn
```

This example displays a "User Info" flow in two columns and sets the varUserFirst and varUserLast variables (both of type Text) to the running user's FirstNameand LastName field values.

```
/flow/User_Info?varUserFirst={!$User.FirstName}&varUserLast={!$User.LastName}&flowLayout=twoColumn
```

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

- Set the Runtime Experience for URL-Based Flows
- Two-Column Flow Considerations
- Flow Screen Output Component: Section

Embed a Flow in a Custom Aura Component

To customize how your flow gets and receives data, add it to a custom Aura component. Then distribute that component through a custom action, Lightning tab, or Lightning page.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To embed a flow in your Aura component, add the `lightning:flow` component to it, for example:

```
<aura:component>
  <aura:handler name="init" value="{!this}" action="{!c.init}" />
  <lightning:flow aura:id="flowData" />
</aura:component>
```

In the JavaScript controller, identify which flow to start, for example:

```
({
  init : function (component) {
    // Find the component whose aura:id is "flowData"
    const flow = component.find("flowData");
    // In that component, start your flow. Reference the flow's API Name.
    flow.startFlow("myFlow");
  },
})
```

Embed a Flow in a Visualforce Page

To customize your flow's look and feel for internal users, add the flow to a Visualforce page. Then distribute that page through a Visualforce tab, custom button, or custom link.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, and set version settings for Visualforce pages:

- Customize Application

1. Find the flow's API name.
   a. From Setup, enter *`Flows`* in the Quick Find box, then select **Flows**.
   b. Click the name of the flow.
   c. Copy the API name of the flow.
2. From Setup, enter *`Visualforce Pages`* in the `Quick Find` box, then select **Visualforce Pages**.
3. Define a new Visualforce page, or open an existing one.
4. Add the `<flow:interview>` component somewhere between the `<apex:page>` tags.
5. Set the `name` attribute to the API name of the flow.

   For example:

   ```
   <apex:page>
   <flow:interview name="flowAPIName"/>
   </apex:page>
   ```

   If the flow is from a managed package, the `name` attribute must be in this format: `namespace.flowuniquename`.
6. Click **Save**.
7. Restrict which users can access the Visualforce page.
   a. Click **Visualforce Pages**.
   b. Click **Security** next to your Visualforce page.
   c. Move all the appropriate profiles from Available Profiles to Enabled Profiles by using the add and remove buttons.

d. Click **Save**.

8. Add the Visualforce page to your app by using a custom button, link, or Visualforce tab.

Customize a Visualforce Component to Control the Flow's Finish Behavior
By default, users who click **Finish** start a new interview and see the first screen of the flow. After you embed a flow in a Visualforce page, configure the `finishLocation` attribute to route users to another page in Salesforce.

Customize a Visualforce Component to Control the Flow's Finish Behavior

By default, users who click **Finish** start a new interview and see the first screen of the flow. After you embed a flow in a Visualforce page, configure the `finishLocation` attribute to route users to another page in Salesforce.

Set **`finishLocation`** with the **`URLFOR`** Function

Note:

- You can't redirect flow users to a URL that's external to your Salesforce org.
- Don't call the `Auth.SessionManagement.finishLoginFlow` method and the `finishLocation` attribute in the same flow. `Auth.SessionManagement.finishLoginFlow` indicates the end of a Visualforce page login flow. If `finishLocation` is in the same flow, `finishLocation` executes when the flow starts, giving users full access to the session.

To route users to a relative URL or a specific record or detail page, using its ID, use the `URLFOR` function.

This example routes users to the Salesforce home page.

```
<apex:page>
    <flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/home/home.jsp')}"/>
</apex:page>
```

This example routes users to a detail page with an ID of 001D000000IpE9X.

```
<apex:page>
    <flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/001D000000IpE9X')}"/>
</apex:page>
```

For details about `URLFOR`, see Functions in the *Visualforce Developer's Guide*.

Set **`finishLocation`** with the **`$Page`** Variable

To route users to another Visualforce page without using `URLFOR`, set `finishLocation` to the name of the destination page with the format `{!$Page.pageName}`.

```
<apex:page>
    <flow:interview name="MyUniqueFlow" finishLocation="{!$Page.MyUniquePage}"/>
</apex:page>
```

For details about `$Page`, see Global Variables in the *Visualforce Developer's Guide*.

Set **`finishLocation`** with a Controller

You can set `finishLocation` in a few ways with a custom controller.

This sample controller configures a flow's finish behavior in three different ways.

```
public class myFlowController {

    public PageReference getPageA() {
        return new PageReference('/300');
    }

    public String getPageB() {
        return '/300';
    }

    public String getPageC() {
        return '/apex/my_finish_page';
    }
}
```

Here’s a sample Visualforce page that references the controller and sets the flow finish behavior to the first option.

```
<apex:page controller="myFlowController">
    <h1>Congratulations!</h1> This is your new page.
    <flow:interview name="flowname" finishLocation="{!pageA}"/>
</apex:page>
```

If you use a standard controller to display a record on the same page as the flow, users who click **Finish** start a new flow interview. They see the first screen of the flow, without the record, because the `id` query string parameter isn't preserved in the page URL. If needed, configure the `finishLocation` to route users back to the record.

Prepare Your Org for Paused Flow Interviews

A *flow interview* is a running instance of a flow. Not every flow interview can be completed in one go. Add the Pause button to your flows, so that users can pause flow interviews for later. Update the sharing model for flow interviews, so that other users can resume a paused interviews. And make it easy for users to resume interviews by adding a component to their Home page.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Let Users Pause Flow Interviews

When users can’t finish a flow interview, give them the option to pause it for later by customizing your org’s process automation settings. For example, a customer service representative can pause an interview when the customer doesn’t have all the necessary information.

Add Record Context to Your Flows

All it takes to associate your org’s paused interviews with a record is setting the $Flow.CurrentRecord global variable in your flow. That way, you can find all the paused flow interviews related to that record. For example, in the Change Address flow, set `$Flow.CurrentRecord` to *`{!recordId}`* so that all Change Address interviews are associated with the relevant contact.

Make It Easy for Users to Find Their Paused Flow Interviews

Give your users an instant view of their paused flow interviews by customizing the Home page or Salesforce mobile app navigation menu.

Make It Easy for Users to Find Paused Flow Interviews for a Record

From a record page, display a list of all paused flow interviews that are associated with that record with this custom Aura component.

Customize Who Has Access to Paused Flow Interviews

By default, users can resume paused flow interviews as long as they have edit access. To control who has edit access, build a sharing model for the Flow Interview object. Configure the org-wide default access level, and build sharing rules to override that default for specific users or groups.

Restrict Who Can Resume Shared Flow Interviews

By default, users with the Run Flows permission or a Flow User feature license can resume any paused flow interview that they have edit access to. You can set Salesforce to only allow each interview to be resumed by the interview owner, or by an admin with Manage Flow permission and view access to the interview.

Delete a Paused Flow Interview

Delete long running or paused flow interviews to take actions on them, for example, to update or delete the flow version.

Let Users Pause Flow Interviews

When users can't finish a flow interview, give them the option to pause it for later by customizing your org's process automation settings. For example, a customer service representative can pause an interview when the customer doesn't have all the necessary information.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

1. From Setup, enter `Automation` in the `Quick Find` box, then select **Process Automation Settings**.
2. Select **Let users pause flows**.
3. Click **Save**.

Screens automatically display the Pause button after **Let Users Pause Flows** is enabled.

SEE ALSO:

Flow Element: Screen

Add Record Context to Your Flows

All it takes to associate your org's paused interviews with a record is setting the $Flow.CurrentRecord global variable in your flow. That way, you can find all the paused flow interviews related to that record. For example, in the Change Address flow, set `$Flow.CurrentRecord` to `{!recordId}` so that all Change Address interviews are associated with the relevant contact.

When a user pauses an interview or an interview executes a Wait element, the interview is associated with the record through the FlowRecordRelation object.

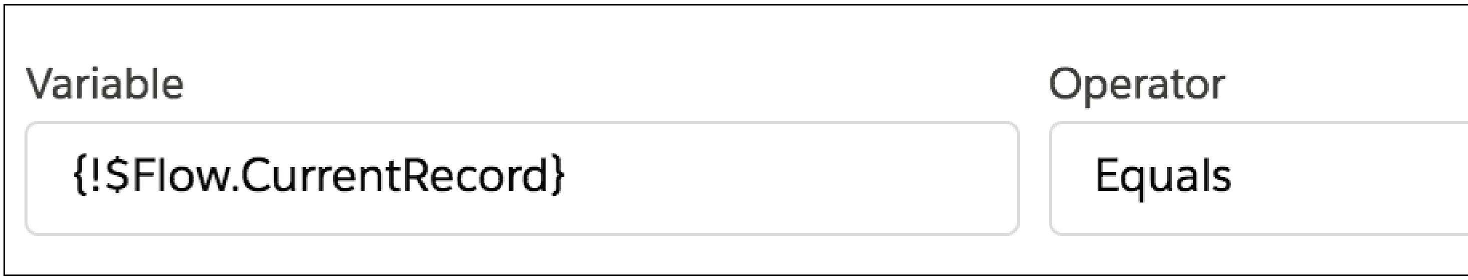
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

1. At the beginning of your flow, add an Assignment element.
2. For Variable, select **$Flow.CurrentRecord**.
3. For Operator, leave **equals** selected.
4. For Value, select a variable that contains the appropriate ID.

   Make sure that the variable contains only one ID.



Variable

{!$Flow.CurrentRecord}

Operator

Equals

Make It Easy for Users to Find Their Paused Flow Interviews

Give your users an instant view of their paused flow interviews by customizing the Home page or Salesforce mobile app navigation menu.

- Lightning Experience—Add the Paused Flow Interviews component to the appropriate Home pages. This component is available only for Home pages in the Lightning App Builder. It displays paused interviews that the user has read access to.
- Experience Builder Site—Add the Paused Flows component to a site page. This component is available for most pages in Experience Builder, except ones like login pages and error pages. The component displays paused interviews that the user has read access to.
- Salesforce mobile app—Add the Paused Flows item to the navigation items of any Lightning app.
- Salesforce Classic—Add the Paused Flow Interviews related list to the appropriate home page layouts. This component displays only interviews that the user paused.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Set Up the Lightning Experience Home Page

Salesforce Classic Home Tab Page Layouts

Make It Easy for Users to Find Paused Flow Interviews for a Record

From a record page, display a list of all paused flow interviews that are associated with that record with this custom Aura component.

Example: This example uses the Apex controller to get a list of interviews that are associated with the record. The component then displays the interviews in a table. For each interview, the component displays an action menu from which the user can resume or delete the interview.

When the user clicks **Resume**, the helper fires the navigateFlow action to resume the interview. When the user clicks **Delete**, the Apex controller deletes the interview.

`c:interviewsByRecord` Component

```
<aura:component controller="interviewsByRecordController"
implements="flexipage:availableForRecordHome,force:hasRecordId" access="global" >
    <aura:attribute name="columns" type="List" default=""/>
    <aura:attribute name="Interviews" type="Object" />
    <aura:attribute name="recordId" type="Id" />
    <aura:attribute name="ContextRecord" type="Object" />
    <aura:attribute name="overlay" type="Aura.Component"/>
    <aura:handler name="init" value="{!this}" action="{!c.init}" />
    <aura:handler event="force:refreshView" action="{!c.init}" />
```

```
<force:recordData aura:id="contextRecord" recordId="{!v.recordId}"
   targetFields="{!v.ContextRecord}" layoutType="FULL"/>

<lightning:overlayLibrary aura:id="overlayLib" />
<lightning:card iconName="standard:flow" class="slds-card_boundary">
    <aura:set attribute="title">
        <span class="slds-card__header-link">Paused Flow Interviews</span>
    </aura:set>
    <aura:set attribute="actions">
        <lightning:buttonIcon iconName="utility:refresh" onclick="{!c.init}"
                              alternativeText="Refresh the list of interviews" />

    </aura:set>
 <table class="slds-table slds-table--bordered slds-table--cell-buffer
    slds-table_fixed-layout">
   <thead>
      <tr class="slds-text-heading--label">
         <th scope="col"><div class="slds-truncate">Interview Label</div></th>
         <th scope="col"><div class="slds-truncate">Pause Reason</div></th>
         <th scope="col"><div class="slds-truncate">Paused Date</div></th>
         <th scope="col"><div class="slds-truncate">Current Element</div></th>
         <th scope="col"><div class="slds-truncate">Owner</div></th>
         <th scope="col" style="width: 3.25rem;"><div class="slds-truncate"/>
             <div class="slds-th__action">
                <span class="slds-assistive-text">Actions</span>
             </div>
          </th>
      </tr>
   </thead>
   <tbody>
      <!-- Use the Apex controller to fetch interviews associated
        with this record -->
      <aura:iteration items="{!v.Interviews}" var="interview">
         <tr>
             <th scope="row">
                <div class="slds-truncate" title="{!interview.InterviewLabel}">
                    {!interview.InterviewLabel}
                 </div>
             </th>
             <td role="gridcell">
                 <div class="slds-truncate" title="{!interview.PauseLabel}">
                     {!interview.PauseLabel}
                 </div>
             </td>
             <td role="gridcell">
                 <div class="slds-truncate" title="{!interview.PausedDate}">
                     <ui:outputDateTime value="{!interview.PausedDate}"
                        format="M/d/y h:m a"/>
                 </div>
             </td>
             <td role="gridcell">
                 <div class="slds-truncate" title="{!interview.CurrentElement}">
                     {!interview.CurrentElement}
                     </div>
                 </td>
                 <td role="gridcell">
                     <div class="slds-truncate" title="{!interview.PausedBy}">
                         {!interview.PausedBy}
                     </div>
                 </td>
                 <td role="gridcell">
                     <!-- Display Resume and Delete actions in a menu at the
                        end of each row -->
                     <div class="slds-shrink-none">
                         <lightning:buttonMenu iconSize="x-small"
                            class="paused-interview-card-row-menu"
                             alternativeText="Actions for this interview"
                             onselect="{! c.handleMenuSelect }">
                             <lightning:menuItem aura:id="{!interview.Id + 'resume'}"

                                label="Resume" value="{!interview.Id + '.resume'}" />

                             <lightning:menuItem aura:id="{!interview.Id + 'delete'}"

                                label="Delete" value="{!interview.Id + '.delete'}"/>
                         </lightning:buttonMenu>
                      </div>
                </td>
             </tr>
          </aura:iteration>
       </tbody>
    </table>
    </lightning:card>
</aura:component>
```

Apex Controller

```
public class interviewsByRecordController {

    @AuraEnabled
    public static List<FlowRecordRelation> getInterviews(Id recordId) {
        return [ SELECT
                    ParentId, Parent.InterviewLabel, Parent.PauseLabel,
                    Parent.CurrentElement, Parent.CreatedDate, Parent.Owner.Name
                FROM FlowRecordRelation
                WHERE RelatedRecordId = :recordId ];
    }

    @AuraEnabled
    public static FlowInterview deleteInterview(Id interviewId) {
       FlowInterview interview = [Select Id from FlowInterview Where Id = :interviewId];

        delete interview;
        return interview;
    }
}
```

`c:interviewsByRecord` JavaScript Controller

```
({
   init : function(component, event, helper) {
      helper.populateTable(component, event, helper);
   },

   handleMenuSelect: function(component, event, helper) {
      // Figure out which action was selected
      var interviewAction = event.getParam("value").split(".");
      if(interviewAction.includes("resume")) {
         helper.handleShowModal(component, interviewAction[0]);
      } else if(interviewAction.includes("delete")) {
         helper.handleDelete(component, event, helper, interviewAction[0]);
      }
   },

   statusChange: function(component, event) {
      // When the interview finishes, close the overlay
      if(event.getParam("status").includes("FINISHED")) {
         component.get("v.overlay").close();
      }
   }
})
```

`c:interviewsByRecord` Helper

```
({
    populateTable : function(component, event, helper) {
        var action = component.get("c.getInterviews");
        action.setParams({
            recordId: component.get("v.recordId")
        });
        action.setCallback(this, $A.getCallback(function (response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                // Push interviews fetched by the Apex controller to the component
                var recordRelations = response.getReturnValue();
                var interviews = [];
                for (var i = 0; i < recordRelations.length; i++) {
                    interviews.push(
                        {
                            Id: recordRelations[i].ParentId,
                            InterviewLabel: recordRelations[i].Parent.InterviewLabel,

                            PauseLabel: recordRelations[i].Parent.PauseLabel,
                            CurrentElement: recordRelations[i].Parent.CurrentElement,

                            PausedDate: recordRelations[i].Parent.CreatedDate,
                            PausedBy: recordRelations[i].Parent.Owner.Name
                        });
                }
                component.set('v.Interviews', interviews);
            } else if (state === "ERROR") {
                var errors = response.getError();
                console.error(errors);
            }
        }));
        $A.enqueueAction(action);
    },

    handleShowModal: function (component, id) {
        // On resume, render the interview in a modal
        $A.createComponent("lightning:flow", {"onstatuschange":
component.get("c.statusChange")},
            function (content, status) {
                if (status === "SUCCESS") {
                    component.find('overlayLib').showCustomModal({
                        body: content,
                        showCloseButton: true,
                        closeCallback: function () {
                            $A.get('e.force:refreshView').fire();
                        }
                    }).then(function(overlay) {
                        // Use to close the modal later
                        component.set("v.overlay", overlay);
                    });
                    content.resumeFlow(id);
                }
            });
    },

    handleDelete: function (component, event, helper, id) {
        // On delete, pass the interview ID to the Apex controller
        var action = component.get("c.deleteInterview");
        action.setParams({
            interviewId: id
        });
        action.setCallback(this, $A.getCallback(function (response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                // Automatically refresh the table
                helper.populateTable(component, event, helper);
            } else if (state === "ERROR") {
                var errors = response.getError();
                console.error(errors);
            }
        }));
        $A.enqueueAction(action);
    }
})
```

Customize Who Has Access to Paused Flow Interviews

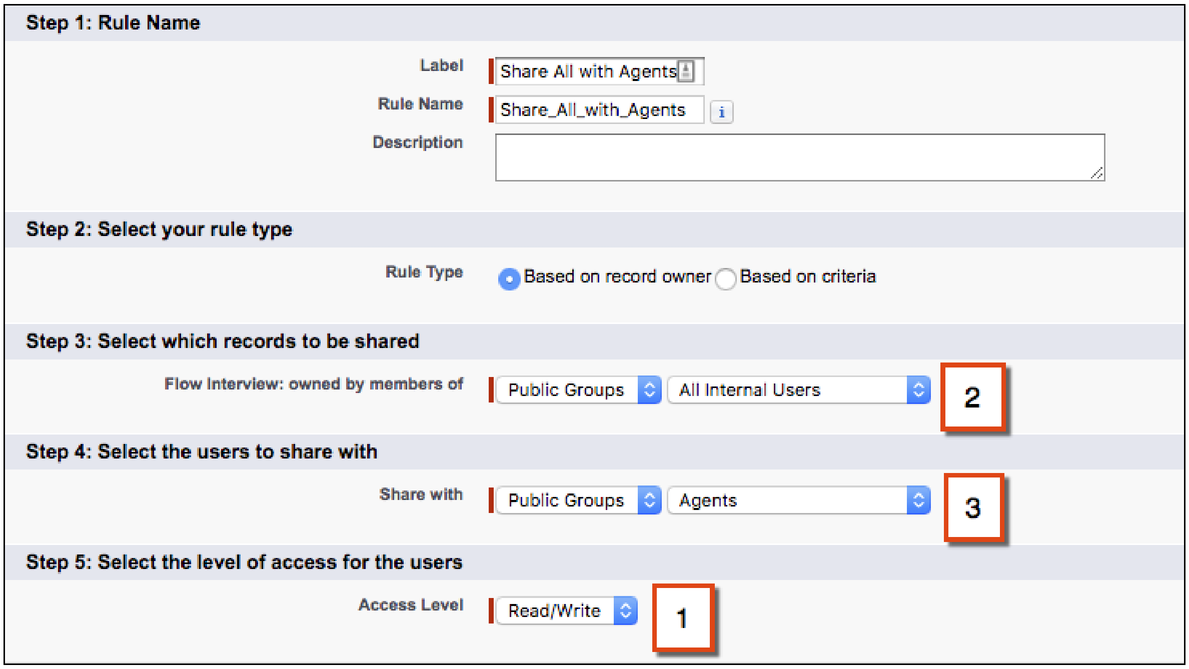
By default, users can resume paused flow interviews as long as they have edit access. To control who has edit access, build a sharing model for the Flow Interview object. Configure the org-wide default access level, and build sharing rules to override that default for specific users or groups.

Note:

- The default sharing model for interviews is Private, which means that users inherit edit access from users lower in the role hierarchy. If your org uses a role hierarchy, users can resume all interviews that users lower in the hierarchy own or have edit access to.
- Users with the CEO role have read/write access to all flow interviews in the org, even if the interview owner isn't part of the hierarchy.

Example: To let all agents in your org resume any interview:

1. Add all agents to the Agents public group.
2. For Flow Interview, leave the organization-wide default set to Private.
3. In a flow interview sharing rule, give read/write access (1) for interviews owned by internal users (2) to the Agents public group (3).



Step 1: Rule Name

Label	Share All with Agents
Rule Name	Share_All_with_Agents
Description	

Step 2: Select your rule type

Rule Type ● Based on record owner ○ Based on criteria

Step 3: Select which records to be shared

Flow Interview: owned by members of Public Groups All Internal Users

Step 4: Select the users to share with

Share with Public Groups Agents

Step 5: Select the level of access for the users

Access Level Read/Write

SEE ALSO:

Restrict Who Can Resume Shared Flow Interviews

Sharing Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Restrict Who Can Resume Shared Flow Interviews

By default, users with the Run Flows permission or a Flow User feature license can resume any paused flow interview that they have edit access to. You can set Salesforce to only allow each interview to be resumed by the interview owner, or by an admin with Manage Flow permission and view access to the interview.

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: You can configure a flow to override default behavior and restrict access to enabled profiles or permission sets. For such a flow, the resuming user must also have access to that flow by a permission set or their profile.

To require the resuming user to be the interview owner, or an admin with the Manage Flow permission and view access to the interview, complete these steps.

1. From Setup, enter *`Automation`* in the `Quick Find` box, then select **Process Automation Settings**.
2. Deselect `Let users resume shared flow interviews`.
3. Click **Save**.

SEE ALSO:

Customize Who Has Access to Paused Flow Interviews

Limit User Access to Execute Flows

Delete a Paused Flow Interview

Delete long running or paused flow interviews to take actions on them, for example, to update or delete the flow version.

1. From Setup, enter *`Flows`* in the Quick Find box, then select **Flows**.
2. For each interview that you want to delete, click **Del**, or click ▾ and select **Delete**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Distribute Flows to Users Outside Your Org

Let external users run your flow by adding the flow to an Experience Builder site, an external app or page, or an Embedded Service deployment. For finer control over how your flow behaves in external contexts, use a custom Aura component or Visualforce page. Flows in custom Aura components use Lightning runtime, and flows in Visualforce pages use Classic runtime.

For example, set up a self-service tool for your site to help visitors generate custom sales quotes.

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Add a Screen Flow to an Experience Builder Site Page

To add a screen flow to a page on your public Experience Builder website, use an Experience Builder Flow component. Before unauthenticated web visitors can see your flow, you must grant the guest user profile access to it. If web visitors log in before they see the screen flow, you don't need to grant permissions to their profiles.

Embed a Flow in a Visualforce Page for External Users

Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

Add a Screen Flow to an Experience Builder Site Page

To add a screen flow to a page on your public Experience Builder website, use an Experience Builder Flow component. Before unauthenticated web visitors can see your flow, you must grant the guest user profile access to it. If web visitors log in before they see the screen flow, you don't need to grant permissions to their profiles.

This solution includes multiple roles and permission sets, which are listed in the linked topics that provide the instructions to complete the steps in Flow Builder, Experience Cloud, and Setup. To refer to this topic, bookmark this page or keep it open in a separate browser tab.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note:

- Flows in Experience Builder sites are supported through the Flow and Suggested Actions components.
- Flow creators can overwrite error messages with their own content.

1. In Flow Builder, create and activate a screen flow.

   See Build a Flow and Screen Flows in Trailhead.

2. In Experience Builder, add the Flow component to the section of the page where you want the screen flow to appear.

   See Flow (Experience Builder Component).

3. In the property editor, configure the Flow component.

   a. From the Flow dropdown, select the active screen flow that you want to show on the page.

   b. If the flow includes variables that are available for input, the variable fields appear in the property editor. You can enter a value for each variable or leave the fields blank.

      If the flow is on a record detail page, you can pass the record ID to the flow by selecting **Pass record ID into this variable** for the record ID input variable. If you have a Test input variable on a Record page and select the **Pass record ID into this variable** option, we recommend that you pass the ID to only one variable.

4. If you want unauthenticated web visitors to complete the screen flow, grant guest user profile permission to the flow and to the Experience Builder site.

   See Allow Guest Users to Access Flows and Guest User Setup Checklist.

5. If your screen flow includes a data element, you must give object-level and field-level permissions to the guest user profile in Setup.

   See Edit Object Permissions in Profiles and Set Field Permissions in Permission Sets and Profiles.

6. Save your work.

7. In Experience Builder, publish your site and test your flow on your site.

See Publish Your Experience Builder Site Customizations.

SEE ALSO:

Limit User Access to Execute Flows

Sharing Settings

Trailhead: Display a Flow on a Page Outside Your Salesforce Org

Trailhead: Grant Flow Permissions and Context

Flow Screen Output Component: Section

Custom Login Flows

Trailhead: Create a Self-Registration Flow for an Experience Cloud Site

Embed a Flow in a Visualforce Page for External Users

Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

USER PERMISSIONS

To create, edit, and set version settings for Visualforce pages:

- Customize Application

For example, you can set up a self-service tool for your public Salesforce site to help visitors generate custom sales quotes. Because the flow is embedded in a Visualforce page, you can customize the appearance of the flow so that it uses your company's branding and style.

> Note: When you make a flow available to site or portal users, point them to the Visualforce page that contains the embedded flow, not the flow itself. Site and portal users aren't allowed to run flows directly.

To add a flow to a Visualforce page, embed it by using the `<flow:interview>` component.

1. Find the flow's API name.
    a. From Setup, enter *`Flows`* in the Quick Find box, then select **Flows**.
    b. Click the name of the flow.
    c. Copy the API name of the flow.
2. From Setup, enter *`Visualforce Pages`* in the `Quick Find` box, then select **Visualforce Pages**.
3. Define a new Visualforce page, or open an existing one.
4. Add the `<flow:interview>` component somewhere between the `<apex:page>` tags.
5. Set the `name` attribute to the API name of the flow.

    For example:

    ```
    <apex:page>
    <flow:interview name="flowAPIName"/>
    </apex:page>
    ```

    If the flow is from a managed package, the `name` attribute must be in this format: `namespace.flowuniquename`.
6. Click **Save**.
7. Restrict which users can access the Visualforce page.

    Any external users with access to the Visualforce page can run the embedded flow.

    a. Click **Visualforce Pages**.
    b. Click **Security** next to your Visualforce page.

c. Move all the appropriate profiles from Available Profiles to Enabled Profiles by using the add and remove buttons.

d. Click **Save**.

8. To distribute your Visualforce page, add the Visualforce page to your Salesforce site. Or use the Visualforce page to define a custom Visualforce tab, and then add that tab to your portal or community.

SEE ALSO:

Lightning Aura Components Developer Guide : Configure Components for Experience Builder

Lightning Aura Components Developer Guide : Add Aura Components to Any App with Lightning Out (Beta)

Distribute Flows to Automated Systems

Some flows don't require any user interaction to start. To enable a system to automatically launch a flow, use the `start` Apex method, a process, or a workflow action.

Most of these methods can be used only with an autolaunched flow. A *flow* can be launched without user interaction, such as from a process or the Apex `interview.start` method. Autolaunched flows run in bulk and without user interaction. They can't contain steps, screens, choices, or dynamic choices in the active or latest flow version. When a flow user invokes an autolaunched flow, the active flow version runs. If there's no active version, the latest version runs. When a flow admin invokes a flow, the latest version always runs.

Launch a Flow from a Process

Just like workflow rules, processes start when a certain object's records are created or edited. Add a flow action to give a process even more functionality. For example, create a process that checks if a new feed item is a question. If it is, wait a day and then use a flow to check whether a Best Comment has been selected or not. If it hasn't, use that question to create a case.

Launch a Flow from a Workflow Action—Pilot

Create a flow trigger workflow action to launch a flow from workflow rules. With flow triggers, you can automate complex business processes—create flows to perform logic, and have events trigger the flows via workflow rules—without writing code. For example, your flow looks up and assigns the relevant entitlement for a case. Create a flow trigger to launch the flow whenever a case is created, so that all new cases are automatically set with a default entitlement.

Launch a Flow from REST API

To invoke an autolaunched flow from REST API, use the invocable action endpoint. For example you can run a flow from an application outside of Salesforce or using code. The flow retrieves data from Salesforce and returns it as an output variable in your REST API call.

SEE ALSO:

Actions Developer Guide: Flow Actions

Apex Reference Guide: Interview Class

Launch a Flow from a Process

Just like workflow rules, processes start when a certain object's records are created or edited. Add a flow action to give a process even more functionality. For example, create a process that checks if a new feed item is a question. If it is, wait a day and then use a flow to check whether a Best Comment has been selected or not. If it hasn't, use that question to create a case.

1. Create and activate the autolaunched flow for the process to launch.

2. Create the process that you plan to launch this flow from.

   For details, see "Create a Process" in Salesforce Help.

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

3. Add a “Flows” action to the process.
   a. For `Flow`, search for and select the flow that you created.
   b. Optionally, click **Add Row** to set values for the flow’s variables.
4. Activate the process.

Launch a Flow from a Workflow Action—Pilot

Create a flow trigger workflow action to launch a flow from workflow rules. With flow triggers, you can automate complex business processes—create flows to perform logic, and have events trigger the flows via workflow rules—without writing code. For example, your flow looks up and assigns the relevant entitlement for a case. Create a flow trigger to launch the flow whenever a case is created, so that all new cases are automatically set with a default entitlement.

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Before you begin, review the special behavior and limitations of flow triggers. See Flow Trigger Considerations (Pilot).

To set up a workflow rule to launch a flow:

1. Create and activate the autolaunched flow to launch from this workflow action.
2. Create the workflow rule that you plan to add this workflow action to.
3. Define the flow trigger.
4. Associate the flow trigger to the workflow rule.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view workflow rules and actions:
- View Setup and Configuration

To create or change workflow rules and actions:
- Customize Application

Associate the Flow Trigger with a Workflow Rule
Add the flow trigger as an immediate action on your workflow rule.

Define a Flow Trigger—Pilot
After you create an autolaunched flow, create a flow trigger to launch that flow as part of a workflow rule.

Associate the Flow Trigger with a Workflow Rule

Add the flow trigger as an immediate action on your workflow rule.

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Before you begin, create:

- An autolaunched flow
- A workflow rule
- A flow trigger that launches the autolaunched flow

1. From Setup, enter *`Workflow Rules`* in the `Quick Find` box, then select **Workflow Rules**.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To select existing actions:
- Customize Application

2. Select the workflow rule.
3. Click **Edit** in the Workflow Actions section.
4. In the Immediate Workflow Actions section, click **Add Workflow Action** > **Select Existing Action**.

   Flow triggers aren't available as time-dependent workflow actions. You can add flow triggers to workflow rules only as immediate workflow actions.

5. In the Search dropdown list, select Flow Trigger.

   The Available Actions box lists all existing flow triggers.

6. Select the flow trigger to associate with this workflow rule. Move the flow trigger to Selected Actions by using the right arrow.
7. Click **Save**.

Define a Flow Trigger—Pilot

After you create an autolaunched flow, create a flow trigger to launch that flow as part of a workflow rule.

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

1. From Setup, enter *Flow Triggers* in the Quick Find box, then select **Flow Triggers**.
2. Click **New Flow Trigger**.
3. Select the same object as the workflow rule, and then click **Next**.
4. Configure the flow trigger.

Field	Description
Name	Name of the flow trigger.
Unique Name	Enter a unique name to refer to this component in the API. The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Protected Component	Reserved for future use.
Flow	Unique name of the autolaunched flow that this workflow action launches.
Set Flow Variables	Whether to pass values into the flow's variables.

5. If you select Set Flow Variables, specify their names and values.

   Click **Set Another Value** to set up to

6. To put the flow trigger in test mode, select Administrators run the latest flow version.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view workflow rules and actions:

- View Setup and Configuration

To create or change workflow rules and actions:

- Customize Application

When selected and an admin triggers the workflow rule, the flow trigger launches the latest version of the flow. For all other users, the flow trigger always launches the active version of the flow.

The same values are passed into the flow variables whether the flow trigger launches the active or latest flow version.

7. Click **Save**.

Don't forget to associate the flow trigger to a workflow rule.

Launch a Flow from REST API

To invoke an autolaunched flow from REST API, use the invocable action endpoint. For example you can run a flow from an application outside of Salesforce or using code. The flow retrieves data from Salesforce and returns it as an output variable in your REST API call.

Example: This example invokes the active version of the flow "Escalate_to_Case". See Flow Actions in the Actions Developer Guide.

```
POST /services/data/v33.0/actions/custom/flow/Escalate_to_Case
```

The request sets values for two of the flow's input variables: `CommentCount` and `FeedItemId`. When it's invoked, the flow checks if:

- A feed item has more than five comments
- A best comment hasn't been selected

```
{
"inputs" : [ {
   "CommentCount" : 6,
   "FeedItemId" : "0D5D0000000cfMY"
   } ]
}
```

Distribute Flows to Other Orgs

Flows can be included in Lightning Bolt Solutions, change sets, and packages. The recipient org of the solution, change set, or package must have flows enabled.

Distribute Flows via Lightning Bolt Solutions

To distribute automated business processes or bootstrap an Experience Builder site with a complete solution or new look, create a Lightning Bolt Solution. A Lightning Bolt Solution can include flows, custom Lightning apps, or Experience Builder templates or pages. Before you create a solution, group flows that you want to include in a flow category.

Deploy Processes and Flows as Active

By default, processes and flows that are active in a sandbox or non-production org are deployed to a production org as inactive. After deployment, manually reactivate the new versions. In production orgs, you can enable the setting to deploy a new active version of a process or flow via change sets or Metadata API. If you use a continuous integration and continuous delivery model to deploy metadata changes, enable the option to deploy processes and flows as active.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Lightning Bolt Solutions are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Change sets are available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Packages are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Distribute Flows via Lightning Bolt Solutions

To distribute automated business processes or bootstrap an Experience Builder site with a complete solution or new look, create a Lightning Bolt Solution. A Lightning Bolt Solution can include flows, custom Lightning apps, or Experience Builder templates or pages. Before you create a solution, group flows that you want to include in a flow category.

1. Add your flows to a flow category. You can only add active flows.
2. Create your Lightning Bolt Solution. Add one or more flow categories, custom Lightning apps, and Lightning Community templates or pages.
3. Package the solution to distribute it to your own orgs, or share or sell it on AppExchange.

SEE ALSO:

Add Flows to a Lightning Bolt Solution

Lightning Bolt for Salesforce: Build Once, Then Distribute and Reuse

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create a flow category:
- Customize Application

To create a Lightning Bolt Solution without an Experience Builder template:
- Customize Application AND View Setup and Configuration

To create a Lightning Bolt Solution with an Experience Builder template:
- Customize Application AND AND

Deploy Processes and Flows as Active

By default, processes and flows that are active in a sandbox or non-production org are deployed to a production org as inactive. After deployment, manually reactivate the new versions. In production orgs, you can enable the setting to deploy a new active version of a process or flow via change sets or Metadata API. If you use a continuous integration and continuous delivery model to deploy metadata changes, enable the option to deploy processes and flows as active.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

Note: This setting applies to processes and autolaunched flows that are deployed via change sets and Metadata API. This setting isn't available in developer, sandbox, or other non-production orgs because you can always deploy a new active version.

1. From Setup, in the Quick Find box, enter `Automation`, then select **Process Automation Settings**.
2. Select **Deploy processes and flows as active**.
3. Enter the flow test coverage percentage.
4. Save your changes.

Before you can deploy a process or autolaunched flow as active, make sure you meet flow test coverage requirements. At least one Apex test must cover the flow test coverage percentage of the active processes and autolaunched flows. Flow test coverage requirements don't apply to flows that have screens.

To calculate your flow test coverage, determine the number of all active flow versions with or without test coverage. Also determine the number of flow versions that are inactive, the latest version, and have test coverage. Here's a sample query.

```
SELECT count_distinct(Id)
FROM Flow
WHERE Status = 'Active' AND Id NOT IN (
SELECT FlowVersionId
FROM FlowTestCoverage
)
+
SELECT count_distinct(FlowVersionId)
FROM FlowTestCoverage
```

To determine the number of all latest flow versions that have test coverage, run all tests and use the Tooling API FlowTestCoverage object. Here's a sample query.

```
SELECT count_distinct(FlowVersionId)
FROM FlowTestCoverage
```

Divide the second number (number of all latest flow versions that have test coverage) by the first number (number of all active versions with or without test coverage and all of the latest inactive versions that have test coverage).

For example, you have a total of 10 flows. Flow A has two versions. The latest version is inactive with test coverage. The first version is active without test coverage. Flows B-E each have only one version, and each version is inactive with test coverage. Flows F-J each have only one version, and each version is active with test coverage. The flow test coverage is 90%.

Flow Label	Version	Status	Test Coverage
Flow A	2	Inactive	Yes
Flow A	1	Active	No
Flow B	1	Inactive	Yes
Flow C	1	Inactive	Yes
Flow D	1	Inactive	Yes
Flow E	1	Inactive	Yes
Flow F	1	Active	Yes
Flow G	1	Active	Yes
Flow H	1	Active	Yes
Flow I	1	Active	Yes
Flow J	1	Active	Yes

```
SELECT FlowVersion.Definition.DeveloperName
FROM FlowTestCoverage
GROUP BY FlowVersion.Definition.DeveloperName
```

Tip: To get the names of all active autolaunched flows and processes that don’t have test coverage, use this query.

```
SELECT Definition.DeveloperName
FROM Flow
WHERE Status = 'Active'
   AND (ProcessType = 'AutolaunchedFlow'
      OR ProcessType = 'Workflow'
      OR ProcessType = 'CustomEvent'
      OR ProcessType = 'InvocableProcess')
   AND Id NOT IN (SELECT FlowVersionId FROM FlowTestCoverage)
```

SEE ALSO:

Tooling API : FlowTestCoverage

Flow Interviews

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

A *flow interview* is a running instance of a flow. A *flow* is an application built by your administrator that asks you for inputs and does something in Salesforce based on those inputs.

For example, a flow could provide a call script for customer support calls and use the information you provide to create a case. What the flow does with the information you provide is entirely up to your administrator.

When you run a flow interview, whether through a link, button, or tab, you're running a single instance of a flow. If the terminology is confusing, consider the difference between a record and an object. You create an account *record*, which is a single instance of the Account *object* that your administrator customized.

Note: Keep these tips in mind when you run a flow.

- Don't use your browser's Back or Forward buttons to navigate through a flow. Doing so can result in inconsistent data between the flow and Salesforce.
- A single flow can have up to 50 different versions. When you run a flow, you see the active version, but your admin could have a more recent version.

Delete a Flow Interview

If you paused a flow interview and don’t plan to resume it, delete it. By removing unnecessary flow interviews, you make sure that your pending list includes only flow interviews that you still plan to act on.

Pause a Flow Interview

If your administrator has configured a given flow to do so, you can pause its interviews. Pausing is useful, for example, when a call with a customer drops or the customer can't find their account number and plans to call you back.

Resume a Flow Interview

If you paused a flow interview, resume it after you have all the necessary information.

SEE ALSO:

Automate Tasks with Flows

Delete a Flow Interview

If you paused a flow interview and don’t plan to resume it, delete it. By removing unnecessary flow interviews, you make sure that your pending list includes only flow interviews that you still plan to act on.

1. Go to a place that lists all of your paused flow interviews. Depending on your admin’s configurations, you can find all the flow interviews that you've paused in one of these places.
   - "Paused Flow Interviews" on the Home tab in Salesforce Classic
   - “Paused Flows” in the Salesforce mobile app navigation menu

   If you paused a flow interview and can't find the place where you're supposed to resume flow interviews from, contact your admin.
2. Delete the flow interview that’s no longer necessary.

SEE ALSO:

Flow Interviews

Resume a Flow Interview

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To delete a flow interview:
- Run Flows

  OR

  `Flow User` field enabled on the user detail page

Pause a Flow Interview

If your administrator has configured a given flow to do so, you can pause its interviews. Pausing is useful, for example, when a call with a customer drops or the customer can't find their account number and plans to call you back.

1. In an open flow interview, click **Pause**.
2. Explain why you had to pause the flow.

   This step is optional, but it helps differentiate between the different flow interviews that you've paused, especially if you paused multiple interviews of the same flow. This explanation can be up to 255 characters long.
3. Click **OK**.

The flow interview is saved until you resume or delete it later. Any valid values that you entered before you paused are saved with the interview, so you don't have to reenter that information when you resume.

SEE ALSO:

Flow Interviews

Resume a Flow Interview

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To pause a flow interview:
- Run Flows

  OR

  `Flow User` field enabled on the user detail page

Resume a Flow Interview

If you paused a flow interview, resume it after you have all the necessary information.

Note: If you pause a flow interview and the associated flow is updated before you resume, the resumed flow interview doesn't use the updated flow. Instead, it uses the version of the flow that was active when you paused.

1. Go to a place that lists all of your paused flow interviews. Depending on your administrator's configurations, you can find all the flow interviews that you've paused in one of these places.
   - "Paused Flows" on a page in an Experience Builder site
   - "Paused Flow Interviews" on the Home tab in the full Salesforce site
   - "Paused Flows" in the Salesforce mobile app navigation menu

   If you paused a flow interview and can't find the place where you're supposed to resume flow interviews from, contact your administrator.

2. Resume the appropriate flow interview.

   Note: If you entered values before you paused, all valid values are restored to those fields when you resume. If you don't see something that you entered before you paused, that value was invalid and the flow didn't save it. For example, if you enter "Acme, Inc." for a field that only accepts numbers and then pause, that field is blank when you resume the flow interview.

Warning: When you resume a flow interview, that flow interview is removed from your Paused Flow Interviews list. If you resume a flow interview and then change your mind, click **Pause**. If you close the flow interview before pausing, you can't resume the flow interview later.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To resume a flow interview:
- Run Flows

  OR

  `Flow User` field enabled on the user detail page

Monitor Flows and Processes

Monitor your org's usage of flows and processes. See a list of paused interviews and scheduled actions from processes. Control who can view and how they view monitoring information for flows and processes.

Control Your Views of Flows, Paused Interviews, and Scheduled Actions
You have two options for viewing flows, paused flow interviews, and scheduled actions from processes. We recommend using the enhanced option that you get by default. But you can switch to the other option, for example, to use a custom list view that you haven't recreated with the enhanced option.

View On-Canvas Insights and Flow Reports
For Data Cloud-triggered, segment-triggered, and form-triggered flows, element run data shows directly on the flow canvas. From an element's analytics tab, you can also access more detailed reports on the element's run data.

Analyze Flows and Processes (Beta)
Get a quick view of your most commonly used automation types. See your org's flow activity in charts, such as total errors and total started automations. Learn how much time it's taking users to complete screen flows, so that you can make changes as needed.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Require Access to Automation Home Charts (Beta)

By default, all users with the View Setup and Configuration permission can view all charts in Automation Home. To limit access, you can require that users have the Manage Flow permission to view all charts in Automation Home. Then users with View Setup and Configuration permission can view only the Total Started Automations by Process Type chart.

Flow Report: Screen Flows

See who's running the screen flows that you build. From the Reports tab, use the Sample Flow Report: Screen Flow to discover flow interview execution counts and screen duration times and to review the status of a flow interview.

Control Your Views of Flows, Paused Interviews, and Scheduled Actions

You have two options for viewing flows, paused flow interviews, and scheduled actions from processes. We recommend using the enhanced option that you get by default. But you can switch to the other option, for example, to use a custom list view that you haven't recreated with the enhanced option.

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: You can complete these steps in Salesforce Classic or Lightning Experience, but the changes affect only the pages and list views in Lightning Experience.

1. From Setup, enter `Automation Settings` in the Quick Find box, then select **Process Automation Settings**.
2. Select or deselect **In Lightning Experience, use the enhanced Flows page and separate Paused and Scheduled Automations page**.

   This setting affects where lists appear and what they contain.

When the setting is...	Selected	Deselected
The Flows page in Setup...	Lists both standard and custom flows	Lists only custom flows
The list of flows...	Can include the packages that flows are installed from	Never includes the packages that flows are installed from
The list of paused flow interviews and scheduled actions from processes...	Appears on its own Paused Flow Interviews page in Setup	Appears as a related list on the Flows page in Setup

SEE ALSO:

Monitor Your Processes' Pending Scheduled Actions

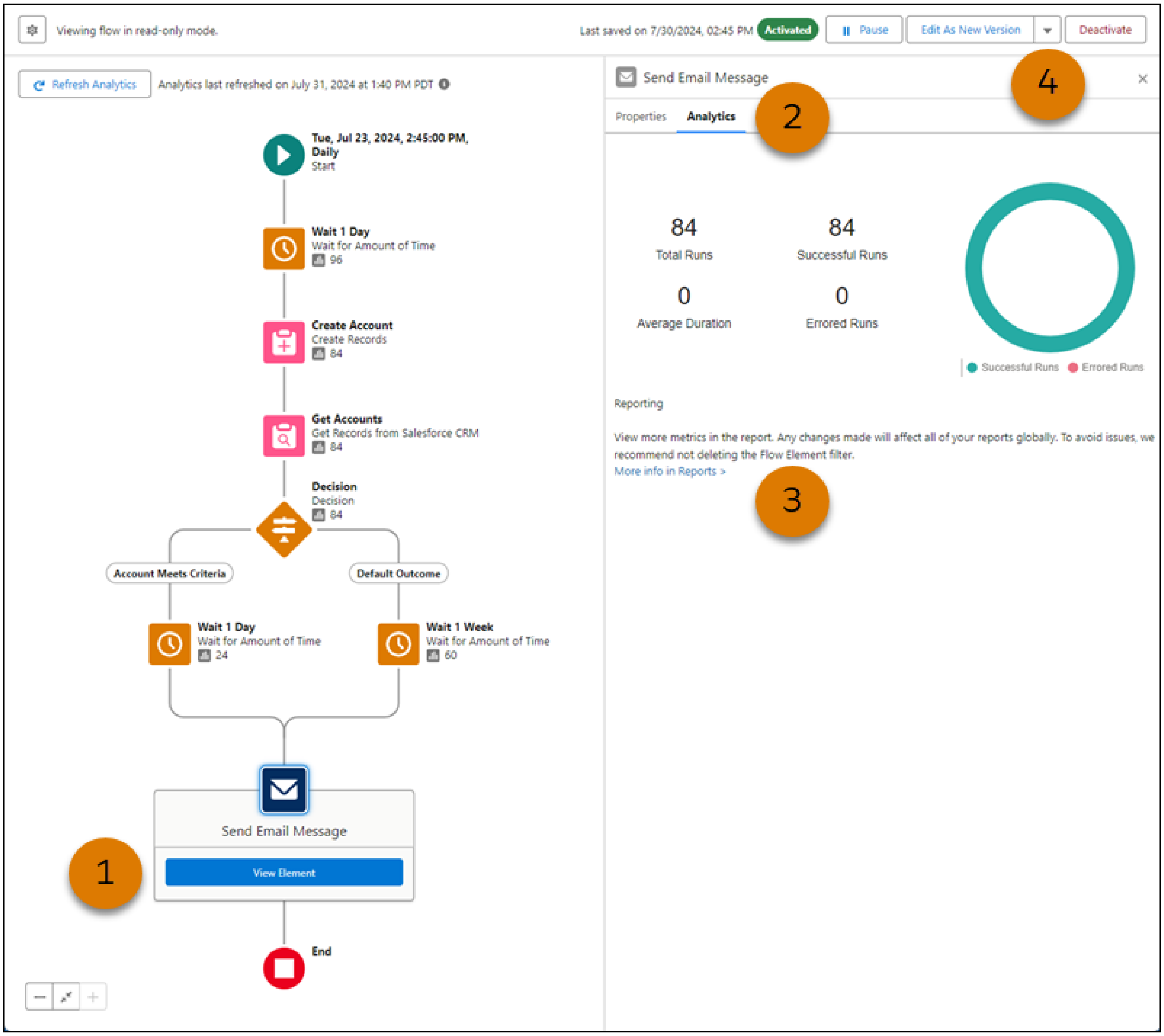
View On-Canvas Insights and Flow Reports

For Data Cloud-triggered, segment-triggered, and form-triggered flows, element run data shows directly on the flow canvas. From an element's analytics tab, you can also access more detailed reports on the element's run data.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions with **Marketing Cloud Advanced Edition**.



84 Total Runs

84 Successful Runs

0 Average Duration

0 Errored Runs

Reporting

View more metrics in the report. Any changes made will affect all of your reports globally. To avoid issues, we recommend not deleting the Flow Element filter.

More info in Reports >

Important: Metrics for on-canvas insights and reports aren't available for flow runs that completed prior to the Winter '25 release.

Active and previously run Data Cloud-triggered, segment-triggered, and form-triggered flows open in read-only mode, which displays analytics and insights for each flow element.

- To see an element's analytics, click the element and then click **View Element** (1). On the Analytics tab (2), you can see how many times an element was run, how many times it succeeded, how many times it errored, and its average duration.
- To view reports on your flow data, in an elements Analytics tab, click **More info in Reports** (3). To access reports, make sure your org has the Flow Reports Analytics Package installed.
- To get out of read-only mode, click **Edit as New Version** (4) or **Save As New Flow** in the dropdown menu.

For certain elements, like Send Email actions, a successful run means only that the element ran successfully. A successful run doesn't necessarily indicate that the email was sent or delivered. To see more details about the results of a Send Email action, create a Data Cloud report using one of the Message Engagement DMOs, filtered by the flow element ID.

Note: To access reports, install the Flow Reports Analytics Package in Marketing Cloud Setup. On-canvas insights aren't available for form-triggered flows that were created prior to Winter '25.

Flow Element Analytics Report Considerations

- Any changes made to a Flow Element Analytics report affects all reports of the same type. For example, editing the report from Flow A also makes those changes to the report from Flow B. To mitigate any issues, keep the Element ID filter as the report's first filter.
- The reports show raw data about the element's run, including information about the individual.

SEE ALSO:

Salesforce Help: Data Cloud Reports and Dashboards

Analyze Flows and Processes (Beta)

Get a quick view of your most commonly used automation types. See your org's flow activity in charts, such as total errors and total started automations. Learn how much time it's taking users to complete screen flows, so that you can make changes as needed.

Note: As a beta feature, Automation Home is a preview and isn't part of the "Services" under your Main Services Agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is and isn't supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. You can provide feedback and suggestions for Automation Home in the Trailblazer Community.

To monitor and analyze your flows and processes in Automation Home, from Setup, enter `Automation` in the Quick Find box, and select **Automation Home (Beta)**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view all charts in Automation Home:

- View Setup and Configuration

Require Access to Automation Home Charts (Beta)

By default, all users with the View Setup and Configuration permission can view all charts in Automation Home. To limit access, you can require that users have the Manage Flow permission to view all charts in Automation Home. Then users with View Setup and Configuration permission can view only the Total Started Automations by Process Type chart.

> Note: As a beta feature, Automation Home is a preview and isn't part of the "Services" under your Main Services Agreement with Salesforce. Use this feature at your sole discretion, and make your purchase decisions only on the basis of generally available products and features. Salesforce doesn't guarantee general availability of this feature within any particular time frame or at all, and we can discontinue it at any time. This feature is for evaluation purposes only, not for production use. It's offered as is and isn't supported, and Salesforce has no liability for any harm or damage arising out of or in connection with it. All restrictions, Salesforce reservation of rights, obligations concerning the Services, and terms for related Non-Salesforce Applications and Content apply equally to your use of this feature. You can provide feedback and suggestions for Automation Home in the Trailblazer Community.

To require that users have the Manage Flow permission to view all flow activity charts:

1. From Setup, enter `Process Automation` in the Quick Find box, and select **Process Automation Settings**.
2. Select **Require the Manage Flow permission to view all Automation Home charts**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view all charts in Automation Home:

- View Setup and Configuration

Flow Report: Screen Flows

See who's running the screen flows that you build. From the Reports tab, use the Sample Flow Report: Screen Flow to discover flow interview execution counts and screen duration times and to review the status of a flow interview.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

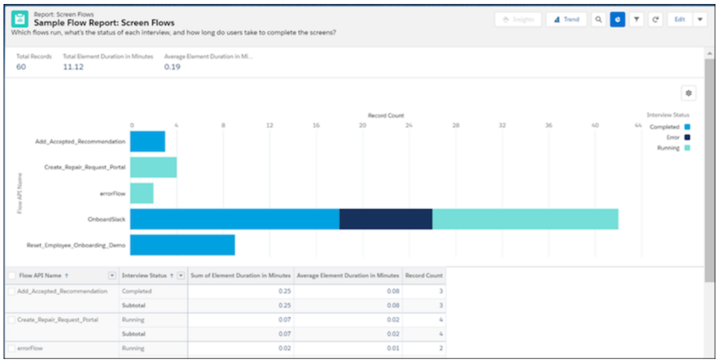
USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To view other users' flow log data:

- View All Data



Report: Screen Flows

Sample Flow Report: Screen Flows

Which flows run, what's the status of each interview, and how long do users take to complete the screens?

Total Records 60

Total Element Duration in Minutes 11.12

Average Element Duration in Mi... 0.19

Flow API Name ↑	Interview Status ↑	Sum of Element Duration in Minutes	Average Element Duration in Minutes	Record Count
Add_Accepted_Recommendation	Completed	0.25	0.08	3
	Subtotal	0.25	0.08	3
Create_Repair_Request_Portal	Running	0.07	0.02	4
	Subtotal	0.07	0.02	4
errorFlow	Running	0.02	0.01	2

The Sample Flow Report: Screen Flow tracks active flows that are still in progress and completed flows. Use this custom report to discover information about your screen flows. For example, you can see who's running a screen flow that you created and get a better understanding of its adoption. Also, you can compare how quickly teams complete their tasks with flows versus without them. Metrics like these provide insights into whether the business can benefit from more process automation.

To display the Sample Flow Report: Screen Flow report, Salesforce adds the Screen Flows custom report type and report to orgs that haven't reached their org limit for the number of custom report types. For more information on custom report types, see Custom Report Types.

Limits and Limitations

Understand the limits and limitations of the Screen Flows custom report.

Limits

- The Flow Interview Log Entries object generates a log record for each screen flow that's active in the last 31 days.
- The limit for the Flow Interview Log Entries object is 7 million log records a month, for the previous 31 days.

  Note: Orgs rarely reach the monthly limit. If they do, Salesforce stops logging metrics for new flows. However, if a flow interview is happening when the limit is reached, Salesforce tracks metrics for the flow interview. Salesforce automatically deletes all logs that are older than 31 days, even for orgs that haven't reached the limit.

Limitations

- By default, the Sample Flow Report: Screen Flows report is visible to all users. However, each user can only see information about the flows that they're the running user for. Only users with the View All Data permission can see other users' flow log data. The report uses data from the new Flow Interview Logs and Flow Interview Log Entries primary objects. To give users permission to see information about flows run by other users, use sharing settings to grant access to the Flow Interview Logs object. For more information, see Set Share Settings.
- Orgs with a custom report type with the same API name or a report with the same report type name don't receive the Screen Flows custom report type. If your org doesn't receive the report, create a custom report type by using the Flow Interview Logs and Flow Interview Log Entries objects. Users can then create custom reports by using the custom report type.

View Screen Flow Usage

Grant access to the Flow Interview Logs object through the sharing settings so that users can view information about flows run by other users.

1. From the App Launcher, in the Quick Find box, enter `report`, and then click **Reports**.

2. Click **Public Folders**.
3. Find and select **Sample Flow Report: Screen Flow**.

   The report displays flow interviews that ran within a specified date range for the Flow Interview Logs and Flow Interview Log Entries objects. By default, the results are sorted by flow API name and interview status. The results include the total time and the average time that users spent on a flow screen. The results also include the number of flow interview log records generated for a flow within the reporting period.

4. To change how the report is sorted or to report on different fields, edit the report. For more information, see Edit or Delete a Custom Report Type.
5. If you don't see all of your flows, change the date filter to a wider range.

   The default date range is the last 7 days, and the maximum range is the last 31 days.

6. To restrict view access to the report, move it to a different folder.

SEE ALSO:

Salesforce Help: Custom Report Types

Salesforce Help: Edit or Delete a Custom Report Type

Salesforce Help: Set Share Settings

Troubleshoot Flow Errors

If a flow interview fails, Salesforce sends an email to either the admin who last modified the associated flow or the Apex exception email recipients. The email includes the error message, details about each flow element that the interview executed, and a link to view the failed flow interview in Flow Builder.

- If the interview fails at multiple elements, or if failures occur in a batch of flow interviews, the recipients receive either multiple emails or one email with an error message for each failure.
- If a flow interview encounters an error and takes a fault path, the flow executes the elements in the fault path instead of sending a fault email.
- To view the failed flow interview in an interactive environment, click **Flow Error: Click here to debug the error in Flow Builder** in the flow fault email.
- To debug the flow and observe what happens as it runs, use the debug option in Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Process and flow error emails include the data that's involved in the process or flow, including user-entered data.

Example:

```
An error occurred at element Apex_Plug_in_1.
List index out of bounds: 0.

An error occurred at element Delete_1.
DELETE --- There is nothing in Salesforce matching your delete criteria.

An error occurred at element Email_Alert_1.
Missing required input parameter: SObjectRowId.
```

Debug a Flow in Flow Builder

If you're troubleshooting a flow that fails, the debug option in Flow Builder can be your best friend. See real-time details of what your flow does, set input variables, and restart the flow anytime to debug a different branch.

Troubleshooting Flow URLs

If you're distributing a flow and the custom button, custom link, or a direct flow URL isn't working as expected, verify the referenced flow. In addition, verify its variables if you're passing values into a flow from the URL.

Select Flow and Process Error Email Recipients

When a process or flow interview fails, a detailed email is sent to the admin who last modified the process or flow. However, sometimes the admin isn't the best person to act on the details of what was executed and what went wrong. In that case, you can send error emails to the Apex exception email recipients.

SEE ALSO:

Considerations for Flow Error Emails

Customize What Happens When a Flow Fails

Considerations for Troubleshooting Flows

What Happens When an Apex Exception Occurs?

Select Flow and Process Error Email Recipients

Send Alerts When a Screen Flow Fails

Debug a Flow in Flow Builder

If you're troubleshooting a flow that fails, the debug option in Flow Builder can be your best friend. See real-time details of what your flow does, set input variables, and restart the flow anytime to debug a different branch.

Warning: If you debug a flow without choosing to run the flow in rollback mode, the flow performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.

Debug isn't available with all flow types. See Considerations for Troubleshooting Flows on page 282.

1. Open the flow in Flow Builder.
2. Click **Debug**.
3. Set the debug options and input variables.

   The debug options vary depending on the flow type.

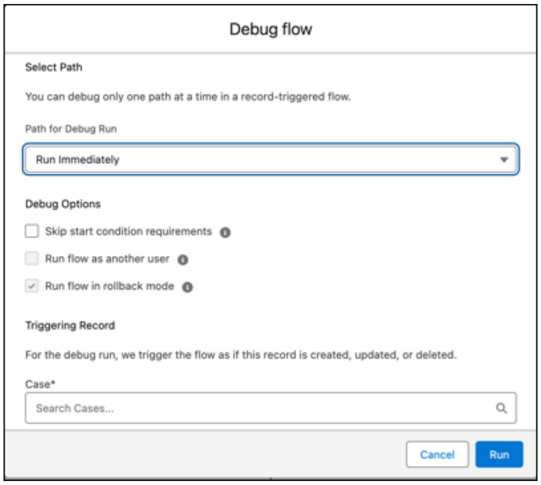
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To debug a flow in Flow Builder:
- View All Data



Debug flow

Select Path

You can debug only one path at a time in a record-triggered flow.

Path for Debug Run

Run Immediately

Debug Options

☐ Skip start condition requirements

☐ Run flow as another user

☑ Run flow in rollback mode

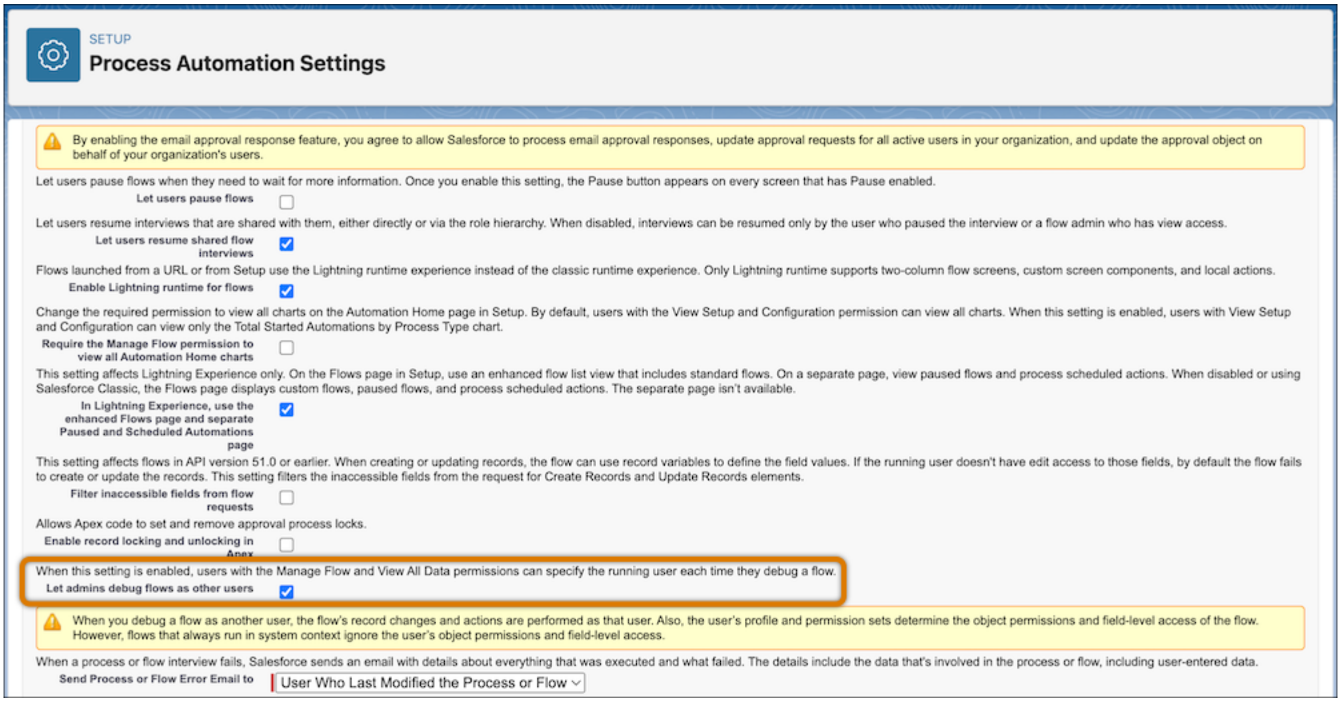
Triggering Record

For the debug run, we trigger the flow as if this record is created, updated, or deleted.

Case*

Search Cases...

4. To debug the flow as another user, enable debugging as another user.
   - a. From Setup, in the Quick Find box, enter `Process`, and then click **Process Automation Settings**.
   - b. Enable **Let admins debug flows as other users**.
   - c. Save your work.



SETUP

Process Automation Settings

By enabling the email approval response feature, you agree to allow Salesforce to process email approval responses, update approval requests for all active users in your organization, and update the approval object on behalf of your organization's users.

Let users pause flows when they need to wait for more information. Once you enable this setting, the Pause button appears on every screen that has Pause enabled.

Let users pause flows ☐

Let users resume interviews that are shared with them, either directly or via the role hierarchy. When disabled, interviews can be resumed only by the user who paused the interview or a flow admin who has view access.

Let users resume shared flow interviews ☑

Flows launched from a URL or from Setup use the Lightning runtime experience instead of the classic runtime experience. Only Lightning runtime supports two-column flow screens, custom screen components, and local actions.

Enable Lightning runtime for flows ☑

Change the required permission to view all charts on the Automation Home page in Setup. By default, users with the View Setup and Configuration permission can view all charts. When this setting is enabled, users with View Setup and Configuration can view only the Total Started Automations by Process Type chart.

Require the Manage Flow permission to view all Automation Home charts ☐

This setting affects Lightning Experience only. On the Flows page in Setup, use an enhanced flow list view that includes standard flows. On a separate page, view paused flows and process scheduled actions. When disabled or using Salesforce Classic, the Flows page displays custom flows, paused flows, and process scheduled actions. The separate page isn't available.

In Lightning Experience, use the enhanced Flows page and separate Paused and Scheduled Automations page ☑

This setting affects flows in API version 51.0 or earlier. When creating or updating records, the flow can use record variables to define the field values. If the running user doesn't have edit access to those fields, by default the flow fails to create or update the records. This setting filters the inaccessible fields from the request for Create Records and Update Records elements.

Filter inaccessible fields from flow requests ☐

Allows Apex code to set and remove approval process locks.

Enable record locking and unlocking in Apex ☐

When this setting is enabled, users with the Manage Flow and View All Data permissions can specify the running user each time they debug a flow.

Let admins debug flows as other users ☑

When you debug a flow as another user, the flow's record changes and actions are performed as that user. Also, the user's profile and permission sets determine the object permissions and field-level access of the flow. However, flows that always run in system context ignore the user's object permissions and field-level access.

When a process or flow interview fails, Salesforce sends an email with details about everything that was executed and what failed. The details include the data that's involved in the process or flow, including user-entered data.

Send Process or Flow Error Email to: User Who Last Modified the Process or Flow

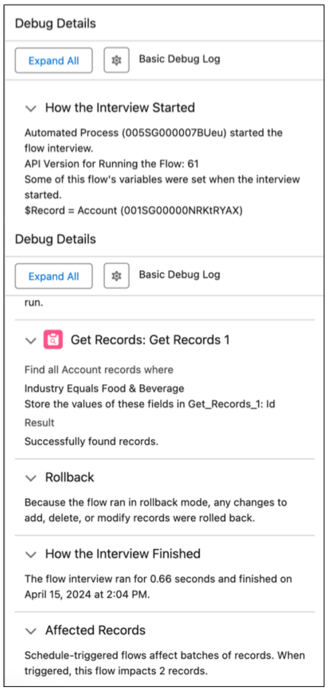
> Warning: When you debug a flow as another user, the flow's record changes and actions are performed as that user. Also, the user's profile and permission sets determine the object permissions and field-level access of the flow. However, flows that always run in system context ignore the user's object permissions and field-level access.

   - d. In Debug Options, select **Run flow as another user** and search for the user that you want to debug.

     You can debug a flow as another user only in a sandbox environment.

5. Click **Run**.

The debug details for the run appear in a panel on the right.



Debug Details

How the Interview Started

Automated Process (005SG000007BUeu) started the flow interview.
API Version for Running the Flow: 61
Some of this flow's variables were set when the interview started.
$Record = Account (001SG00000NRKtRYAX)

Debug Details

run.

Get Records: Get Records 1

Find all Account records where
Industry Equals Food & Beverage
Store the values of these fields in Get_Records_1: Id
Result
Successfully found records.

Rollback

Because the flow ran in rollback mode, any changes to add, delete, or modify records were rolled back.

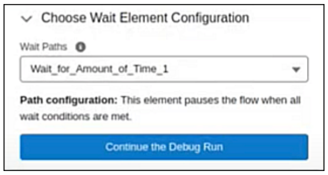
How the Interview Finished

The flow interview ran for 0.66 seconds and finished on April 15, 2024 at 2:04 PM.

Affected Records

Schedule-triggered flows affect batches of records. When triggered, this flow impacts 2 records.

6. If you selected Debug wait element behavior when you set the debug options, select a **Wait Path**, and then click **Continue the Debug Run** for each Wait element in the flow.
   The debug details for the run appear in a panel on the right.



Choose Wait Element Configuration

Wait Paths: Wait_for_Amount_of_Time_1

Path configuration: This element pauses the flow when all wait conditions are met.

7. (Optional) To restart the flow by using the same or different values for the input variables, click **Debug Again**.
8. (Optional) To convert the debug run to a test in a record-triggered flow only, click **Convert to Test**.

Flow Example: Debug a Screen Flow

Let’s debug a screen flow that creates a contact for each beneficiary on a policy by creating a registration form.

Flow Example: Debug a Template-Triggered Prompt Flow
Let’s debug a prompt flow that integrates with a prompt template for generating a list of events.

SEE ALSO:

Customize What Happens When a Flow Fails

Considerations for Troubleshooting Flows

Lightning Runtime vs. Classic Runtime for Flows

Flows in Transactions

Testing Your Flow

Trailhead: Flow Troubleshooting

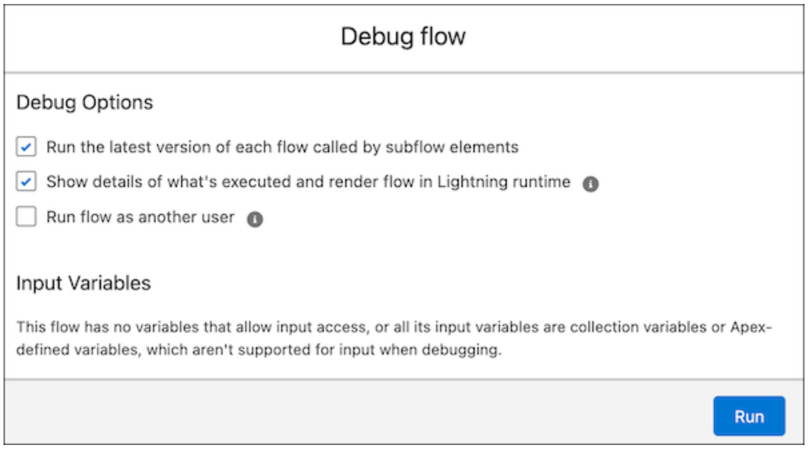
Flow Example: Debug a Screen Flow

Let’s debug a screen flow that creates a contact for each beneficiary on a policy by creating a registration form.

Before activating and distributing your screen flow, you debug the flow to troubleshoot any flow failure.

1. Create or open the Registration forms on page 155 screen flow in Flow Builder.
2. Click **Debug**.
3. Set the debug options.

   If you want to run the flow as another user, ensure that **Let admins debug flows as other users** is enabled in **Process Automation Settings**.



Debug flow

Debug Options

☑ Run the latest version of each flow called by subflow elements

☑ Show details of what's executed and render flow in Lightning runtime

□ Run flow as another user

Input Variables

This flow has no variables that allow input access, or all its input variables are collection variables or Apex-defined variables, which aren't supported for input when debugging.

4. Click **Run**.
   The debug details for the run appear in a panel on the right.
5. Enter the values in the required fields.

EDITIONS

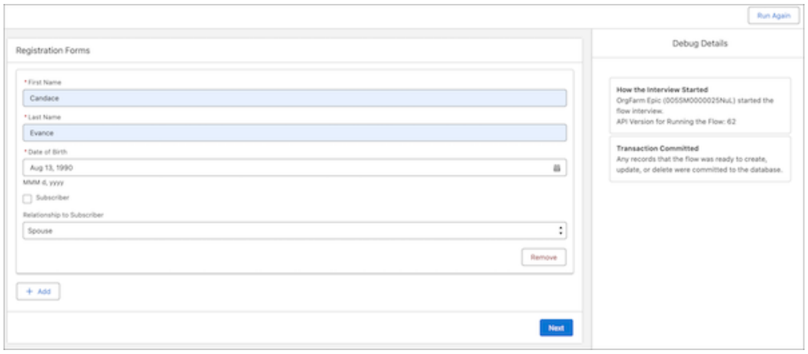
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow



*First Name: Candace

*Last Name: Evance

*Date of Birth: Aug 13, 1990

MMM d, yyyy

□ Subscriber

Relationship to Subscriber: Spouse

How the Interview Started
OrgFarm Epic (005SM000025NuL) started the flow interview.
API Version for Running the Flow: 62

Transaction Committed
Any records that the flow was ready to create, update, or delete were committed to the database.

6. Click **Next**.
7. Review the Debug Details to see the results.
8. (Optional) To restart the flow by using the same or different values for the input variables, click **Change Inputs or Run Again**.

   You can't convert the debug run to a test in the screen flow.

SEE ALSO:

Flow Example: Create a Contact for Each Beneficiary on a Policy

Considerations for Troubleshooting Flows

Trailhead: Flow Troubleshooting

Flow Example: Debug a Template-Triggered Prompt Flow

Let's debug a prompt flow that integrates with a prompt template for generating a list of events.

Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

Before creating your sales email prompt template, you debug the flow to troubleshoot any flow failure.

1. Create or open the Get Marketing Events on page 45 prompt flow in Flow Builder.
2. Click **Debug**.
3. Set the debug options and input variables.

   If you want to run the flow as another user, ensure that **Let admins debug flows as other users** is enabled in **Process Automation Settings**.

   Warning: If you debug a flow without choosing to run the flow in rollback mode, the flow performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.

4. Enable **Run flow in rollback mode** under Debug Options.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on

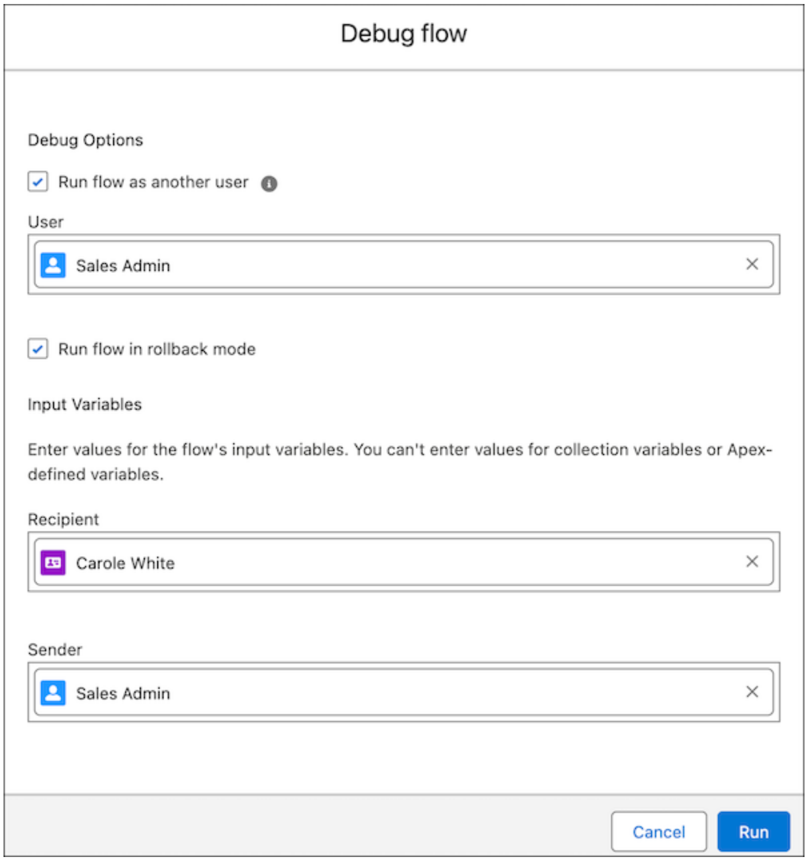
USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and manage prompt templates in Prompt Builder:

- Prompt Template Manager permission set



Debug flow

Debug Options

☑ Run flow as another user

User

Sales Admin

☑ Run flow in rollback mode

Input Variables

Enter values for the flow's input variables. You can't enter values for collection variables or Apex-defined variables.

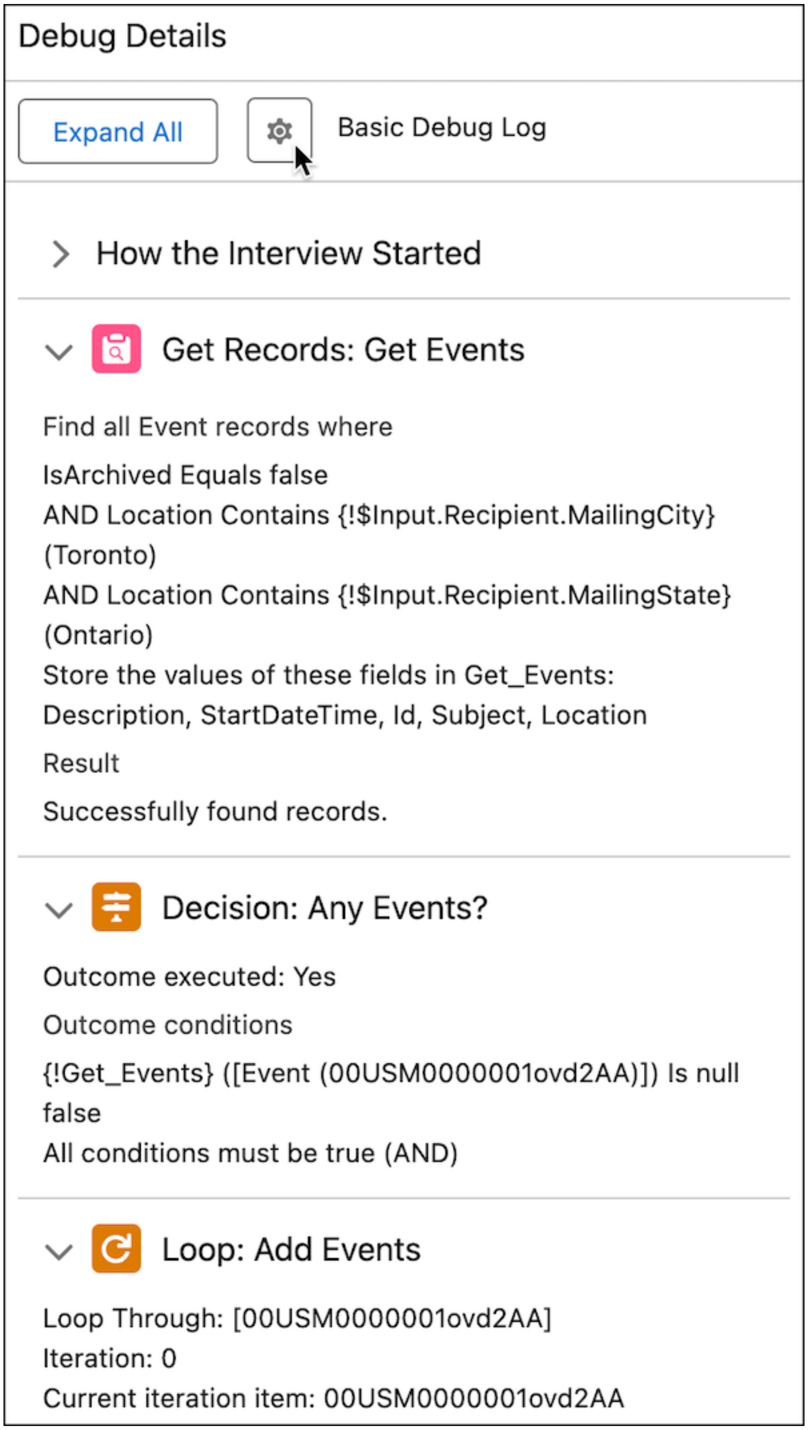
Recipient

Carole White

Sender

Sales Admin

5. Click **Run**.
   The debug details for the run appear in a panel on the right.
6. Review the Debug Details to see the results.



Debug Details

How the Interview Started

Get Records: Get Events

Find all Event records where

IsArchived Equals false
AND Location Contains {!$Input.Recipient.MailingCity}
(Toronto)
AND Location Contains {!$Input.Recipient.MailingState}
(Ontario)
Store the values of these fields in Get_Events:
Description, StartDateTime, Id, Subject, Location

Result

Successfully found records.

Decision: Any Events?

Outcome executed: Yes

Outcome conditions

{!Get_Events} ([Event (00USM0000001ovd2AA)]) Is null
false
All conditions must be true (AND)

Loop: Add Events

Loop Through: [00USM0000001ovd2AA]
Iteration: 0
Current iteration item: 00USM0000001ovd2AA

7. (Optional) To restart the flow by using the same or different values for the input variables, click **Debug Again**.

You can't convert the debug run to a test in the prompt flow.

SEE ALSO:

Example of Sales Email Template-Triggered Prompt Flow

Considerations for Troubleshooting Flows

Trailhead: Flow Troubleshooting

Troubleshooting Flow URLs

If you're distributing a flow and the custom button, custom link, or a direct flow URL isn't working as expected, verify the referenced flow. In addition, verify its variables if you're passing values into a flow from the URL.

To make sure that the URL can find the right flow, verify that:

- The flow that the URL references hasn't been deleted or deactivated.
- The flow name is correctly spelled and capitalized. It must be an exact, case-sensitive match to the flow's API Name.

If your flow URL references a specific flow version, verify that the version hasn't been deleted or deactivated.

If you're using the URL to pass values into the flow and the URL can't access the variable, the parameter that references the variable is ignored.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Make sure that the URL can find the right flow variable and that the value that you're passing is compatible with the variable's data type. Verify that the URL variable:

- Is spelled and capitalized correctly. It must be an exact, case-sensitive match to the flow variable.
- Allows input access.
- Hasn't been renamed in the flow.
- Hasn't been removed from the flow.
- Doesn't have a data type of Record.

SEE ALSO:

Customize a Flow URL to Control Finish Behavior

Customize a Flow URL to Set Variable Values

Troubleshoot Flow Errors

Select Flow and Process Error Email Recipients

When a process or flow interview fails, a detailed email is sent to the admin who last modified the process or flow. However, sometimes the admin isn't the best person to act on the details of what was executed and what went wrong. In that case, you can send error emails to the Apex exception email recipients.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

User Permissions Needed	
To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

Process and flow error emails include the data that's involved in the process or flow, including user-entered data.

1. From Setup, enter `Automation` in the Quick Find box, and then select **Process Automation Settings**.
2. For Send Process or Flow Email to, select who receives the error emails.
   - If you select **User Who Last Modified the Process or Flow**, error emails are sent to the user who last modified the flow that has the error.
   - If you select **Apex Exception Email Recipients**, error emails are sent to the addresses listed on the Apex Exception Email page in Setup.
3. Save your changes.

SEE ALSO:

Troubleshoot Flow Errors

What Happens When an Apex Exception Occurs?

What Happens When a Process Fails?

Customize What Happens When a Flow Fails

Flow Limits and Considerations

When designing, managing, and running flows, consider the permissions, use limits, and data issues.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Usage-Based Entitlements

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce contacts you to discuss additions to your contract. In the meantime, your flow interviews run as usual.

General Flow Limits

When using flows, keep flow limits and Apex governor limits in mind.

Per-Transaction Flow Limits

Salesforce strictly enforces limits to ensure that any runaway flows don't monopolize shared resources in the multitenant environment. Per-transaction limits, which Apex enforces, govern flows. If an element causes the transaction to exceed governor limits, the system rolls back the entire transaction. The transaction rolls back even if the element has a defined fault connector path.

Flow Builder Considerations

When you create a flow in Flow Builder, familiarize yourself with its limitations and behaviors. For example, Flow Builder supports only a few locales. Because of intellectual property protection, you can't open flows installed from managed packages, unless the flows are templates.

Salesforce Feature Considerations for Flows

When designing flows, consider these Salesforce features.

Salesforce Data Considerations for Flows

When designing flows, keep these Salesforce data considerations in mind.

Flow Feature Considerations

When designing flows, keep these flow feature considerations in mind. Also, some resources, elements, and screen components have more considerations that are described in their reference topics.

Flow Data Considerations

When designing flows, keep these data considerations in mind.

Flow Management Considerations

When managing flows, consider the administration and activation limits.

Considerations for Packaging Flows

You can include a flow in a managed or unmanaged package. Before you create, update, or deploy a package that contains a flow, understand the limitations and behaviors of packages.

Change Set Considerations for Flows

Before you use change sets to deploy a flow, understand the limits and behaviors that are related to component dependencies, deployment, and flow triggers.

Considerations for Flows Installed from Packages

Keep these considerations in mind when you distribute, upgrade, or remove a flow that you installed from a package.

Considerations for Troubleshooting Flows

Keep these considerations in mind when reviewing a flow error email or using the debug option in Flow Builder.

Run-Time Changes by Release and API Version

These versioned updates affect only flows that are configured to run on specific API versions. With versioned updates you can test and adopt run-time behavior changes for individual flows at your convenience.

SEE ALSO:

Flow Builder Tour

Flow Usage-Based Entitlements

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce contacts you to discuss additions to your contract. In the meantime, your flow interviews run as usual.

For per-month entitlements, your contract determines the start and end of the month. You can view the start and end dates for your org's usage-based entitlements on the Company Information page in Setup.

Note:

- Flows that are launched by another flow via a Subflow element don't count toward your allocation of flow interviews.
- When a process built in Process Builder launches a flow, both the process and the flow count toward your allocation of flow interviews.
- If you enable recursion for a process built in Process Builder, a separate flow interview starts each time the process evaluates a record. Each flow interview counts toward your allocation of flow interviews.

This table describes the free allocations that are granted based on your org's edition.

Per-Org Usage-Based Entitlement	What's Counted	Essentials and Professional Editions	Performance and Developer Editions	Enterprise and Unlimited Editions
Maximum flow interviews with UI per month	Interviews of flow types that can have screen elements, such as screen flows, user provisioning flows, Field Service flows, contact request flows, and surveys built in Survey Builder	2,000	20,000,000	20,000,000
Maximum flow interviews without UI per month	Interviews of flow types that can't have screen elements, such as autolaunched flows, transaction security flows, and processes built in Process Builder	10,000,000	10,000,000,000	10,000,000,000

This table describes additional allocations that are granted based on purchased user licenses. These allocations apply to the org. It doesn't matter which users run the flows.

Per-Org Usage-Based Entitlement	What's Counted	Enterprise Edition	Unlimited Edition
Additional flow interviews with UI per month	Interviews of flow types that can have screen elements, such as screen flows, user provisioning flows, Field Service flows, contact request flows, and surveys built in Survey Builder	+ 50 for each Service Cloud User license + 50 for each Salesforce CRM Content User license	+ 100 for each Service Cloud User license + 100 for each Salesforce CRM Content User license

If you have questions about increasing your allocation, contact your Salesforce account executive.

SEE ALSO:

- Usage-Based Entitlements
- View Your Salesforce Org’s Usage-Based Entitlements
- Flow Types
- Reevaluate Records in the Process Builder
- How Does Salesforce Process Scheduled Actions?

General Flow Limits

When using flows, keep flow limits and Apex governor limits in mind.

The maximum flow interview size is 1,000,000 B (approximately 1 MB). If the interview is too large, it can't be persisted or paused.

These limits apply to segment-triggered flows, form-triggered flows, and automation-event triggered flows.

Per-Org Limit	**Starter Edition**	**Marketing Cloud Growth Edition**	**Marketing Cloud Advanced Edition**
Active flows per flow type	50	500	750
Total flows per flow type	2,000	50,000	50,000

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

These limits apply to all other flows.

Per-Org Limit	**Essentials or Professional Editions**	**Enterprise, Unlimited, Performance, or Developer Editions**
Versions per flow	50	50
Executed elements at runtime per flow	None	None[1]
Active flows per flow type	5	2,000
Total flows per flow type	5	4,000
Groups of scheduled actions from processes that are executed per hour based on a specific time	1,000	1,000
Combined total of these automations that start or resume based on a record’s field value. • Resume events that are defined in active flows • Groups of scheduled actions that are defined in active processes • Time triggers that are defined in active workflow rules	20,000	20,000

Per-Org Limit	Essentials or Professional Editions	Enterprise, Unlimited, Performance, or Developer Editions
- Inactive flow interviews that are resumed		
Schedule-triggered flow interviews per 24 hours	250,000, or the number of user licenses in your org multiplied by 200, whichever is greater.[2]	

[1] In API version 57.0, the limit of 2000 flow elements was removed. In API version 56.0 and earlier, flows could have a maximum of 2000 flow elements.

[2] The license types that count toward this limit include full Salesforce and Salesforce Platform user licenses, App Subscription user licenses, Chatter Only users, Identity users, and Company Communities users.

SEE ALSO:

Automate Tasks with Flows

Flow Limits and Considerations

Per-Transaction Flow Limits

Salesforce strictly enforces limits to ensure that any runaway flows don't monopolize shared resources in the multitenant environment. Per-transaction limits, which Apex enforces, govern flows. If an element causes the transaction to exceed governor limits, the system rolls back the entire transaction. The transaction rolls back even if the element has a defined fault connector path.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Per-Transaction Limit[1]	Value
Total number of SOQL queries issued (All executions of Get Records elements, and executions of Update Records or Delete Records elements that use filter conditions)	100
Total number of records retrieved by SOQL queries (All executions of Get Records elements, and executions of Update Records or Delete Records elements that use filter conditions)	50,000
Total number of DML statements issued (Create Records, Update Records, and Delete Records executions)	150
Total number of records processed as a result of DML statements	10,000

Per-Transaction Limit[1]	Value
Maximum CPU time on the Salesforce servers	10,000 milliseconds
Total number of duplicate updates allowed in one batch	12

[1] Autolaunched flows are part of the larger transaction that they were launched through and share that transaction's limits. For example, flows launched from Apex or a process are executed with the Apex or process actions as part of the larger transaction. Flows with Screen elements can span multiple transactions. A new transaction begins each time the user clicks **Next** in a screen. Flows with Wait elements span multiple transactions. A transaction ends when a flow interview pauses for an event. When the flow interview resumes, a new transaction begins. Everything after the Wait element is executed as part of a batch transaction that includes other resumed interviews. The batch includes interviews executed by the same user ID, have the same execution time, and have the same flow version ID.

SEE ALSO:

Apex Developer Guide : Execution Governors and Limits

General Flow Limits

Flow Builder Considerations

When you create a flow in Flow Builder, familiarize yourself with its limitations and behaviors. For example, Flow Builder supports only a few locales. Because of intellectual property protection, you can't open flows installed from managed packages, unless the flows are templates.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Access to Salesforce Data

- Flow Builder uses the permissions and locale assigned to the current user.
- Flow Builder has access to information that exists when you open it. If you modify data or metadata in your org and must refer to it in a flow, close and reopen Flow Builder. For example, if you add a custom field or modify an Apex class with Flow Builder open, close and reopen Flow Builder.

Opening Flows That were Saved in Cloud Flow Designer

When you open a flow version created with Cloud Flow Designer in Flow Builder, the Save button is disabled. To edit that version in Flow Builder, save it as a new version in Flow Builder.

Text Formatting

If you open the Display Text screen component, Choice resource labels, help text, Pause confirmation screens, or input validation, Flow Builder converts existing HTML to rich text. Unsupported HTML is removed. The following HTML tags are converted to rich text: <a>, <b>, <br>, <font>, <i>, <li>, <p>, <span>, <u>, and <div>. HTML that is pasted into the rich text editor isn't supported.

Rich Text

- Images uploaded with the rich text editor are stored in the Files tab, and are visible to everyone in your org.
- Images uploaded with the rich text editor aren't visible in Experience Cloud sites.

- Toggle back to plain text when using a text template in a Post to Chatter action, Send Email action, or in a custom action that expects plain text. Click ▼ and select **Plain Text**.
- If you include incomplete HTML symbols such as `<` in rich text, Flow Builder removes the symbols and adjacent strings. For example, Flow Builder renders `2<3` as `2`. Include spaces around HTML symbols to render them. For example, `2 < 3`.

Date/Time Values

At run time, time zones for date/time values can differ from what you see in Flow Builder. During run time, date/time values reflect the time zone settings of the user who's running the flow. In Flow Builder, date/time values reflect the time zone settings of the admin who configures the flow.

Text Values

- Flow Builder doesn't support UTF-8 encoding for text in user input fields.
- Flow Builder contains embedded fonts for all locales it supports. The supported locales are:
  - Chinese (Traditional)
  - Chinese (Simplified)
  - English (US)
  - French (France)
  - German (Germany)
  - Japanese (Japan)
  - Spanish (Spain)

  If you enter unsupported characters for a supported locale, they're displayed using system fonts instead of the embedded fonts.

  In unsupported locales, your system font settings are used to display all characters in Flow Builder.
- Don't enter the string `null` as the value of a text field in Flow Builder.

Output Values

To store the same output value in multiple variables, assign the value to one variable. Then add an Assignment element after the action, and set the other variables to the value of the first variable.

Managed Packages

Flow Builder can't open a flow that is installed from a managed package, unless the flow is a template or overridable.

Step Elements

You can't add or update steps to a flow in Flow Builder. You also can't convert steps into screens. If you added a step in Cloud Flow Designer, the step still appears on the canvas. We recommend that you remove all steps from your flows.

Action Elements

Legacy Apex actions aren't organized by the tag in the plug-in code.

Flows Upgraded from Winter '12 and Earlier

If you open a flow that was last opened in Winter '12 or earlier, each Boolean decision is converted to a multi-outcome Decision element that:

- Uses the same label as the old decision.
- Takes the API name of the old decision, appended with "_switch".
- Has an outcome labeled "True". This outcome's API name matches that of the old decision, and its conditions are migrated from the True outcome of the old decision.
- Has a default outcome labeled "False".

Terminology

The terminology in some warnings, error messages, and debug details isn't updated for Flow Builder or Cloud Flow Designer.

Salesforce Feature Considerations for Flows

When designing flows, consider these Salesforce features.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Security Considerations for Flows
When designing flows, keep these security considerations in mind.

External Object Considerations for Flows
Keep these considerations in mind when building flows that include external objects.

Lightning Component Considerations for Flows
Keep these considerations in mind when building flows that include Lightning components.

Considerations for Reactivity in Screen Flows
Review these considerations before you set up reactivity in your screen flows. Reactivity is supported with API version 57.0 or later.

Security Considerations for Flows

When designing flows, keep these security considerations in mind.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Interviews

When a user session expires, in-progress flow interviews are interrupted and can't be resumed. If the flow executed actions, such as a Create Records or Post to Chatter element, those actions aren't rolled back. But other progress through the interview, such as what the user entered on the screen, is lost.

Tip:

- Set your session timeout settings to log out users after an appropriate period.
- Encourage your users to pay attention during interviews for alerts about their sessions expiring soon.
- Remind users to avoid running flows during release upgrades. A typical upgrade takes about 5 minutes.

Paused or waiting flow interviews aren't affected by expired user sessions.

Shield Platform Encryption

You can't filter or sort records by encrypted fields for these elements and resources.

- Update Records element
- Delete Records element
- Get Records element
- Record Choice Set resource

Screen Flow Inputs

For enhanced security, remove all HTML from publicly accessible input fields in screen flows. For example, an input field on a publicly accessible screen flow is mapped to a rich text field in Salesforce. To prevent a malicious URL from accessing the rich text field, create a separate flow on the object to strip out the HTML. Optimize the new flow for fast field updates, and set it to run whenever the input field isn't blank. Because several sources can write to a publicly accessible input field, check for HTML at the field level and not at the screen level.

You can also use an existing Apex trigger on the object to strip out the HTML.

SEE ALSO:

Modify Session Security Settings

External Object Considerations for Flows

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Keep these considerations in mind when building flows that include external objects.

- When you create or update external object records, don't set values for indirect lookup relationships that map to a different data type on the external system. For example, don't set a value for a Text indirect lookup relationship that maps to a Date value on the external system.
- To find the Salesforce record linked to from an external object by an indirect lookup relationship, match the parent object's `Id` field to the ID in the indirect lookup relationship field. Select the indirect lookup relationship, and add *`.Id`* before the closing curly bracket. For example, an indirect lookup relationship connects Contact (parent standard object) to Social Media post (child external object). In a flow, the record variable `{!socialMediaPost}` contains field values for a social media post. To find the parent contact record, in a Get Records element, filter by:

```
Id Equals {!socialMediaPost.indirectLookupRelationship_c__c.Id}
```

- To find the parent external object record linked to by an external lookup relationship, match the parent record's external ID to the external lookup relationship on the child record. For example, an external lookup relationship connects Product Catalog Item (parent external object) to Case (child standard object). In a flow, the record variable `{!case}` contains field values for a support case. To find the parent product catalog item record, in a Get Records element, filter by:

```
ExternalId Equals {!case.externalLookupRelationship_c__c}
```

- If Salesforce creates, updates, or deletes data in your org and then accesses external data in the same transaction, an error occurs. In your flow, we recommend using a separate transaction to access data in an external system. To do so, end the prior transaction by adding a screen or local action to a screen flow or a Wait element to an autolaunched flow. If you use a Wait element, don't use a record-based resume time.

For example, a screen flow creates a contact and then displays a confirmation screen. Next, the flow updates the contact in the external system. The flow doesn't fail because it uses a separate transaction to access the external data.

- Don't update the External ID and Display URL fields in a process or flow.
- Record-change processes aren't supported.
- A process or flow must save or commit changes to a standard or a custom object before changing an external object within the same transaction. To commit changes to a standard or custom object, you have different options depending on the tool. After an action that changes a standard or custom object record:
  - In Flow Builder, add a screen, local action, or Wait element that pauses until a flow-based time occurs.
  - In Process Builder, add a scheduled action.

Lightning Component Considerations for Flows

Keep these considerations in mind when building flows that include Lightning components.

Note: These topics are designed for developers that build Lightning components.

- Lightning components in flows must comply with Lightning Locker restrictions.
- Flows that include Lightning components are supported only in Lightning runtime.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Which Custom Lightning Component Attribute Types Are Supported in Flows?
Not all Lightning component data types are supported in flows. You can map only these types and their associated collection types between flows and Lightning components.

Design Attribute Considerations for Flow Screen and Action Components
To expose an attribute in Flow Builder, define a corresponding `design:attribute` in the component bundle's design resource. Keep these guidelines in mind when defining design attributes for flows.

Runtime Considerations for Flows That Include Aura Components
Depending on where you run your flow, Aura components can look or behave differently than expected. The flow runtime app that's used for some distribution methods doesn't include all the necessary resources from the Lightning Component framework. When a flow is run from Flow Builder or a direct flow URL (https://yourDomain.my.salesforce.com/flow/MyFlowName), `force` and `lightning` events aren't handled.

Which Custom Lightning Component Attribute Types Are Supported in Flows?

Not all Lightning component data types are supported in flows. You can map only these types and their associated collection types between flows and Lightning components.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Data Type	Lightning Component Attribute Type	Valid Values
Apex	Custom Apex Class	Apex classes that define `@AuraEnabled` fields. Supported data types in an Apex class are Boolean, Integer, Long, Decimal, Double, Date, DateTime, and String. Single values as well as Lists are supported for each data type.
Boolean	Boolean	• True values: `true`, `1`, or equivalent expression

Flow Data Type	Lightning Component Attribute Type	Valid Values
		• False values: `false`, `0`, or equivalent expression
Currency	Number	Numeric value or equivalent expression
Date	Date	`"YYYY-MM-DD"` or equivalent expression
Date/Time (API name is DateTime)	DateTime	`"YYYY-MM-DDThh:mm:ssZ"` or equivalent expression
Number	Number	Numeric value or equivalent expression
Multi-Select Picklist (API name is Multi-Select Picklist.)	String	String value or equivalent expression using this format: `"Blue; Green; Yellow"`
Picklist	String	String value or equivalent expression
Record, with a specified object (API name is SObject.)	The API name of the specified object, such as Account or Case	Map of key-value pairs or equivalent expression. Flow record values map only to attributes whose type is the specific object. For example, an account record variable can be mapped only to an attribute whose type is Account. Flow data types aren’t compatible with attributes whose type is Object.
Text (API name is Text.)	String	String value or equivalent expression

Design Attribute Considerations for Flow Screen and Action Components

To expose an attribute in Flow Builder, define a corresponding `design:attribute` in the component bundle's design resource. Keep these guidelines in mind when defining design attributes for flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Supported Attributes on `design:attribute` Nodes

In a `design:attribute` node, Flow Builder supports only the `name`, `label`, `description`, and `default` attributes. The other attributes, like `min` and `max`, are ignored.

For example, for this design attribute definition, Flow Builder ignores `required` and `placeholder`.

```
<design:attribute name="greeting" label="Greeting" placeholder="Hello" required="true"/>
```

Calculating Minimum and Maximum Values for an Attribute

To validate min and max lengths for a component attribute, use a flow formula or the component's client-side controller.

Modifying or Deleting `design:attribute` Nodes

If a component's attribute is referenced in a flow, you can't change the attribute's type or remove it from the design resource. This limitation applies to all flow versions, not just active ones. Remove references to the attribute in all flow versions, and then edit or delete the attribute in the design resource.

Runtime Considerations for Flows That Include Aura Components

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Depending on where you run your flow, Aura components can look or behave differently than expected. The flow runtime app that's used for some distribution methods doesn't include all the necessary resources from the Lightning Component framework. When a flow is run from Flow Builder or a direct flow URL (https://yourDomain.my.salesforce.com/flow/MyFlowName), `force` and `lightning` events aren't handled.

To verify the behavior of your Aura components, test your flow in a way that handles `force` and `lightning` events, such as `force:showToast`. You can also add the appropriate event handlers directly to your component.

Distribution Method	Handles `force` and `lightning` Events
Direct flow URL	
Run and Debug buttons in Flow Builder	
Run links on flow detail pages and list views	
Web tab	
Custom button or link	
Lightning page	✓
Experience Builder site page	Depends on the site. Aura sites can handle all flows. LWR sites have limitations, including not being able to run flows that use Aura components.
Flow action	✓
Utility bar	✓
`flow:interview` Visualforce component	
`lightning:flow` Aura component	Depends on where you embed the component or whether your component includes the appropriate event handlers

Considerations for Reactivity in Screen Flows

Review these considerations before you set up reactivity in your screen flows. Reactivity is supported with API version 57.0 or later.

For information on how to set up components for reactivity, refer to Make Your Screen Flows Reactive

General Considerations

- Manual outputs of components don't support reactivity. If you manually set a component output, that variable doesn't change on the same screen when referenced in other components.
- Help text and labels don't react to changes in other components.
- Data types must match when you're mapping an output to another component's input to support reactivity.
- If validation rules exist for custom components, reactive changes don't trigger validation.
- The global variable $Flow is reactive. All other global variables such as Custom Labels, Custom Settings, $Organization, $Profile, are not reactive.
- When mapping a `DateTime` field to `Time`, the value is converted to GMT and stays converted when navigating between screens. If mapped to a `DateTime` field, the locale is preserved. For example, if the time value is 8:00 AM in your locale, the converted GMT time could be several hours off your time locale (such as 4:00 PM). Refer to A Note About Date/Time and Time Zones for information about Converting Between Date/Time and Text and Date/Time in time zones: Using Date, Date/Time, and Time Values in Formulas

EDITIONS

Available in: Lightning Experience, Salesforce Classic.

Available in: All versions of the mobile app in Professional, Performance, and Unlimited editions.

Salesforce Data Considerations for Flows

When designing flows, keep these Salesforce data considerations in mind.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Setting the Record Type

For example, use a Get Records element to find the Record Type record whose name is "Reduction Order." Then store that record type's ID in a variable. You can then use the variable to set the `Order Record Type` field on an order record.

To set the record type for a record, use the record type's ID. Look up the record type by its name and then store its ID in the flow.

Working with Person Accounts

If your org uses person accounts, reference `Contact.Salutation` instead of `Account.Salutation`.

Null Values

A flow fails when a filter condition from a Get Records element or an Update Records element references a value that is null. Before you reference a value in a filter condition, add a Decision element to check if the value is null.

Merge Fields

A flow can reference the value of a merge field at run time only if a flow resource stores the value. For example, a flow can't reference a merge field that a messaging template contains.

SEE ALSO:

- Flow Operations and Read-Only Fields
- Considerations for the Apex-Defined Data Type
- External Object Considerations for Flows

Flow Operations and Read-Only Fields

Understand when flows have read-only access to field values. You can control the behavior when a flow tries to update a read-only field and remove read-only field values from flow operations.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Which Fields Are Inaccessible When a Flow Creates or Updates Records?

A flow can perform an operation only if the running user has permission to do so. When a flow tries to create or update records, fields that the running user can't edit are considered *inaccessible*, or read only. A field can be inaccessible because the user hasn't been granted permission to edit the field or because it's a system field that's always read only.

Control What Happens When a Flow Tries to Set Values for Read-Only Fields

When a flow tries to perform an action, such as create or update records, it uses a flow request to set values for specific fields. But what happens if the running user doesn't have edit access to all those fields? When you use a record variable or record collection variable in Create Records and Update Records elements, that's up to you. To control the behavior, select or deselect the `Filter inaccessible fields from flow requests` preference.

Remove Read-Only Fields from a Record Variable

If a flow tries to update fields that the running user can't edit and `Filter inaccessible fields from flow requests` is not enabled for your org, the flow fails. If your record variable includes read-only fields and you can't grant your running users "Edit" permissions for those fields, remove the fields from the record variable. Set the field values individually in a Create Records or Update Records element or copy the writable field values into a new record variable.

Which Fields Are Inaccessible When a Flow Creates or Updates Records?

A flow can perform an operation only if the running user has permission to do so. When a flow tries to create or update records, fields that the running user can't edit are considered *inaccessible*, or read only. A field can be inaccessible because the user hasn't been granted permission to edit the field or because it's a system field that's always read only.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To determine which fields are system fields, see the *Object Reference for Salesforce and Lightning Platform*. To determine which other fields aren't editable, review the running user's permissions.

How Did Read-Only Fields Get in My Record Variable?

If the Variable Is Populated by ...	The Variable Includes ...
A Get Records element, where the field values are stored together	`Id` and any other read-only fields that you choose to include.
An Assignment or Get Records element, where field values are stored in separate variables	Any read-only fields that you choose to include.
A process, workflow rule, or Start element	All the object's system fields and any fields that the running user doesn't have permission to edit. The variable includes every field for the object by default.

What Do I Do When My Record Variable Includes Read-Only Fields?

For each read-only field that's stored in your record variable:

1. Determine whether the flow uses that field anywhere. If it doesn't, update the flow so that it doesn't store a value for that field. This suggestion applies only if an element in the flow, such as Get Records, populates the variable.

   For example, a Get Records element stores `CreatedByDate`, but no other elements reference that field. You update the Get Records element so that it's no longer storing `CreatedByDate`.

2. If the read-only field is referenced in the flow, give the running users the permissions needed for the flow to execute its operations.
3. If you can't give the running users the needed permissions for a field, update the flow so that it doesn't try to update that field.

Here's an example: Using an Update Records element, a flow updates several fields on an account. While your users can edit `Description` and `Account Rating`, they can't edit `Owner ID` or `LastModifiedDate`. To prevent the flow from failing at run time:

- Give your users "Edit" permission for `Owner ID`.
- Copy only the writable field values (`Description`, `Account Rating`, and `Owner ID`) from the original record variable into a new record variable. Reference the new record variable in the Update Records element.

  Copying only the writable field values ensures that the flow doesn't try to set a value for `LastModifiedDate` at run time.

SEE ALSO:

Remove Read-Only Fields from a Record Variable

Control What Happens When a Flow Tries to Set Values for Read-Only Fields

Object Reference for Salesforce and Lightning Platform : System Fields

Control What Happens When a Flow Tries to Set Values for Read-Only Fields

When a flow tries to perform an action, such as create or update records, it uses a flow request to set values for specific fields. But what happens if the running user doesn't have edit access to all those fields? When you use a record variable or record collection variable in Create Records and Update Records elements, that's up to you. To control the behavior, select or deselect the `Filter inaccessible fields from flow requests` preference.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

	When `Filter inaccessible fields from flow requests` **is**	
	Selected	**Not Selected (Recommended)**
Result when the running user doesn't have edit access to all fields	The operation partially succeeds. The flow filters read-only fields out of the operation. The fields that the user can edit are updated. The fields that the user can't edit aren't updated. The flow doesn't execute the fault path.	The operation fails. No fields in the operation are updated. The flow executes the fault path if there's one.
Notification when one or more fields aren't updated	No notification is sent to the user or admin to indicate that some fields weren't updated.	The admin receives a flow error email with full details.
Compared to a Create Records element where you chose to use separate variables, resources, and literal values	Inconsistent	Consistent
Compared to an Update Records element, where you chose to set fields individually	Inconsistent	Consistent

Tip: We recommend disabling this preference so that you always know when a flow doesn't set all expected field values.

1. From Setup, enter *`Automation`* in the `Quick Find` box, then select **Process Automation Settings**.
2. Select or deselect **Filter inaccessible fields from flow requests**.

   If your org was created in Winter '17 or earlier, the preference is enabled by default. Otherwise, the preference is disabled by default.

Example: A flow updates several fields on an opportunity by using a record variable in an Update Records element. At run time, the flow tries to update the Acme account on behalf of your user. The user can edit `Stage` and `Close Date` but not `Amount`. As a result, the flow doesn't have permission to update `Amount`.

- If `Filter inaccessible fields from flow requests` is selected, the flow successfully updates the account, but it only updates `Stage` and `Close Date`. The flow doesn't notify anybody that `Amount` wasn't updated.

- If `Filter inaccessible fields from flow requests` isn't selected, the flow fails to update the account. The admin receives a flow error email. The email includes this error.

  ```
  INVALID_FIELD_FOR_INSERT_UPDATE: Unable to create/update fields: Amount
  ```

  That's API-speak for "The running user doesn't have permission to edit the Amount field."

Warning: If you change your org's selection for this preference, use a sandbox to test how the change impacts your flows. Consider following the same process as you would for a critical update.

SEE ALSO:

Which Fields Are Inaccessible When a Flow Creates or Updates Records?

Remove Read-Only Fields from a Record Variable

If a flow tries to update fields that the running user can't edit and `Filter inaccessible fields from flow requests` is not enabled for your org, the flow fails. If your record variable includes read-only fields and you can't grant your running users "Edit" permissions for those fields, remove the fields from the record variable. Set the field values individually in a Create Records or Update Records element or copy the writable field values into a new record variable.

Note: If the read-only fields in the record variable are populated by a Get Records or Assignment element, consider updating those elements so that they don't populate that field at all.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Copy Field Values from One Record Variable to Another

Record variables and record collection variables can have values set for fields that the running user can't edit. However, you can use the writable values to create or update records with Create Records or Update Records elements. To do so, map the writable values from the original record variable into a new record variable.

Note: With record collection variables, use loops to map the field values to a new collection.

1. Add an Assignment element to your flow. Make sure that the flow executes this element after the original record variable has been populated but before the Create or Update element.
2. For each writable field in the original record variable, add a row.

   Variable—Select {!*recordVar2.field*}, where *recordVar2* is the name of the new variable and *field* is the field on that variable.

   Operator—Select **equals**.

   Value—Select {!*recordVar1.field*}, where *recordVar1* is the name of the original variable and *field* is the field on that variable.

   Note: If you plan to reference the variable in an Update Records element, include the record's ID in the new record variable. Although `Id` is read only, the flow uses the value to determine which records to update.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

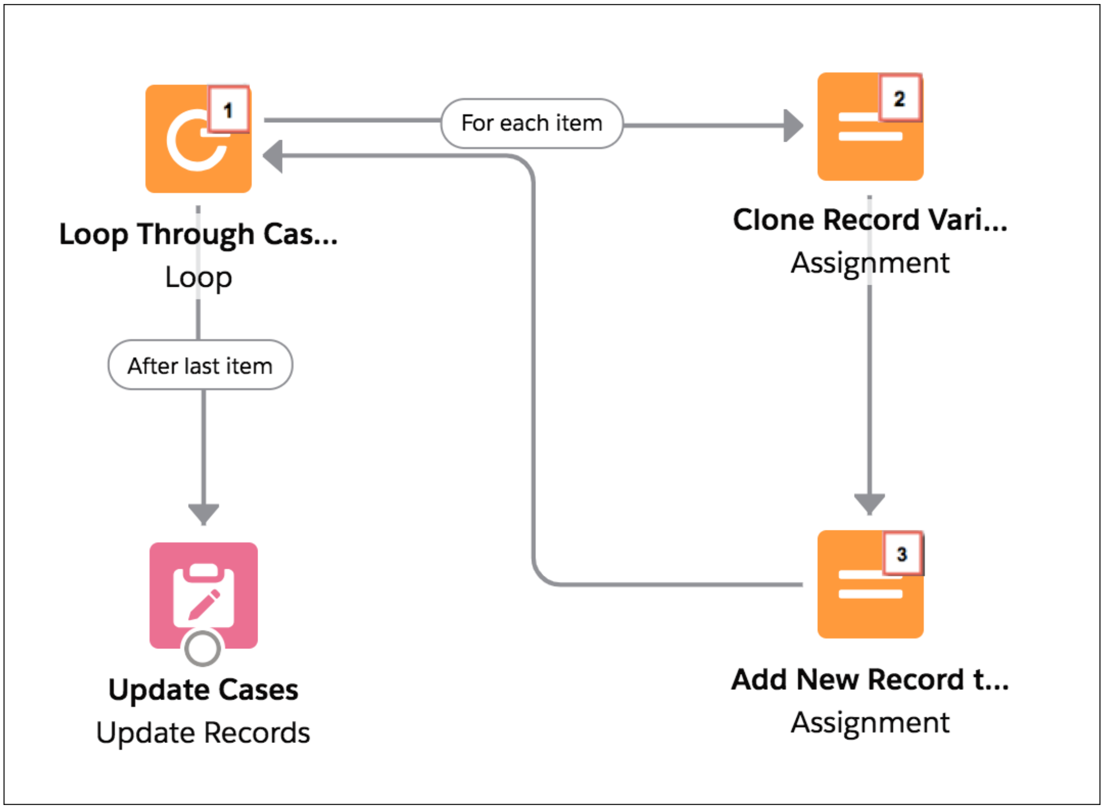
To open, edit, or create a flow in Flow Builder:

- Manage Flow

Example: You have a case record variable called `{!myCaseVar_all}`. It stores values for some read-only fields, so you can't use it in an Update Records element. Copy the fields that you want to update to a new record variable: `IsEscalated` and `Status`. Also, copy `Id` because it's required for an update operation. Here's what those assignment rules look like.

Variable	Operator	Value
{!myCaseVar_final.Id}	equals	{!myCaseVar_original.Id}
{!myCaseVar_final.IsEscalated}	equals	{!myCaseVar_original.IsEscalated}
{!myCaseVar_final.Status}	equals	{!myCaseVar_original.Status}

The same example works for a record collection variable. However, because you can't directly change the values of a collection variable, you use a loop. After the flow has iterated over every item in the original collection, it exits the loop.



- Using a Loop element, the flow passes each item's values into a loop variable (`{!myCaseLoopVar_original}`).
- For each iteration, an Assignment element copies the `Id`, `IsEscalated`, and `Status` fields from the loop variable to another record variable (`{!myCaseLoopVar_final}`).
- The flow then adds the `{!myCaseLoopVar_final}` variable's values to a new collection. The second Assignment element includes this assignment rule.

Variable	Operator	Value
{!myCaseColl_updated}	add	{!myCaseLoopVar_final}

SEE ALSO:

Flow Element: Create Records

Flow Element: Update Records

Considerations for the Apex-Defined Data Type

Understand these considerations when you're building flows that include an Apex-defined data type.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Builder

- Cloud Flow Designer isn't supported.
- A custom component that displays a value, like the Display Text screen component, can display all fields from an Apex-defined variable. For example, the {!Car} variable stores all field values that are defined in the Car Apex class. If a Display Text screen component has the {!Car} Apex-defined variable as the input attribute, the screen displays all the fields from the Car Apex class. If the Apex class is from a managed package, only the Apex class ID is displayed.
- The first time you open an element or resource window in an org with over 200 Apex classes that have the `@AuraEnabled` annotation, the window can take longer to load.
- Deprecated Apex classes in a managed package appear in Flow Builder.
- If a flow invokes Apex, the running user must have the corresponding Apex class assignment in their profile or permission set.
- A flow doesn't support a list of lists data type if it's a field on a flow variable that's an Apex-defined data type.

Apex

- Supported data types in an Apex class are Boolean, Integer, Long, Decimal, Double, Date, DateTime, and String. Single values and lists are supported for each data type. Multiple Apex classes can be combined to represent complex web objects.
- The @AuraEnabled annotation for each field is required.
- A constructor with no arguments is required.
- Class methods aren't supported.
- Getter methods for fields aren't supported.
- Inner classes aren't supported.
- An outer class that has the same name as an inner class isn't supported.
- Referential integrity isn't supported for Apex class fields. For example, a flow has an Apex-defined variable that represents the model field in the Car Apex class. If the model field is modified or deleted in the class, the flow fails.

Input and Output Values

- An Apex-defined variable value can't be set or stored outside the flow. The value can't be passed to a Subflow element.

Local Actions

- An Aura component that's used as a local action can't set an Apex-defined attribute.

Flow Feature Considerations

When designing flows, keep these flow feature considerations in mind. Also, some resources, elements, and screen components have more considerations that are described in their reference topics.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Conditional Visibility Considerations

Before you set visibility for a screen component, understand the behavior of conditional visibility in flows.

Considerations for Flow Choice Components with Default Values

Understand how to set a default value using any flow resource for a screen flow's choice component, such as Radio Buttons or a Multi-Select Picklist component.

Flow Variable Considerations

Before you create a variable resource, understand the behavior of variables in flows.

Multi-Select Resource and Screen Field Considerations for Flows

Checkbox Group, Multi-Select Picklist, and Choice Lookup screen components let flow users select multiple choices. Before you start using these screen components, understand how they work in flows—both when you design the flow and when your users run it.

Paused Flow Interview Considerations

Before you design flows that contain one or more Wait elements, understand the behavior and guidelines.

Flow Stage Considerations

Before you add stages to your flow, understand how stage references and default active stages work, as well as considerations for troubleshooting stages.

Two-Column Flow Considerations

If your org has Lightning runtime enabled, you can control whether a flow displays in one column or two columns. Before you use this feature, understand how the flow layout currently behaves.

Schedule-Triggered Flow Considerations

A schedule-triggered flow starts at the specified time and frequency for a batch of records. Understand the considerations and special behaviors of schedule-triggered flows, also known as scheduled flows.

Record-Triggered Flow Considerations

A record-triggered autolaunched flow makes additional updates to the triggering record before or after it's saved to the database. Understand the considerations and special behaviors of flows that make before- and after-save updates.

SEE ALSO:

Flow Resources

Flow Elements

Standard Flow Screen Components

Flow Conditional Visibility Considerations

Before you set visibility for a screen component, understand the behavior of conditional visibility in flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Null and Empty Strings

A `null` value is evaluated the same way as a `{!$GlobalConstant.EmptyString}`.

Unsupported Data Types and Operators

- These operators aren't supported in conditional visibility.
  - Was Visited
  - Was Set
- These data types aren't supported in conditional visibility, but you can reference attributes and fields.
  - Apex-defined data types
  - Record variables
- You can't reference these results in conditional visibility.
  - Results of Apex-defined invocable actions
  - Results of a flow referenced by the current flow with the Subflow element
- Any screen input component with **Manually assign variables (advanced)** selected isn't available as a resource for conditional visibility on the same flow screen.
- Text that has merge fields included isn't supported in values. Merge fields on their own are supported.

Text Templates and Formulas

- Initial values are evaluated in text templates and formulas.
- Changes based on user input aren't evaluated.

Hidden Screen Input Components

- Screen input components hidden by conditional visibility aren't required when a user runs the flow, even if `Required` is set to *`{!$GlobalConstant.True}`*. When the component appears to the user, it's treated as required.
- If a screen input component is hidden because it doesn't meet conditional visibility requirements, its value is set to null. But hidden picklists in a Dependent Picklists component aren't set to null unless the entire Dependent Picklists component is hidden.
- In an Update Records element, if you update a field by using the value of a hidden screen component, the field value is set to blank. Instead, update the field by using a Formula resource that checks if the field is blank before setting the field value. For example, use this formula.

```
IF( ISBLANK( {!myTextField} ), {!myOriginalFieldValue}, {!myTextField})
```

- When you define a condition to set component visibility, you can specify a variable as the resource value. The variable can traverse up to three object fields, for example, *`Contact.Account.Owner`*.

Sections and Conditional Field Visibility

If a section's visibility conditions reference a component contained within the section, the entire section is hidden. If a section's visibility conditions reference a component visibility condition and the visibility conditions evaluate as true, the section is visible.

Focus

When screen components or their parts are rendered after the screen is initially displayed, they're never focusable. For example, if a component asynchronously fetches a list of tasks to display, the focus can't be set to any of the tasks. If a screen component uses conditional visibility and appears only after user input, the focus can't be set to any part of the screen component.

Circular Logic

Avoid circular logic in your conditions, which can result in poor performance, strange behavior, or an error when your flow is run.

Related Record Fields

Related record fields in your field visibility conditions work only for Lookup fields that have a value set when entering the screen.

Commas in Condition Values

If your condition value contains a comma, add a quotation mark at the beginning and the end of the value. For example, *`"Email, Phone, and Social Media"`*.

Performance

For the best performance, minimize the number and complexity of conditional visibility conditions on your screen components and record fields. There are other ways to control what screen components and record fields your screen flow users have access to view or update.

- Use a Section component. If you need multiple screen components or record fields to be visible using the same logic, put them in a Section component. Then, set the conditional visibility on the Section component, instead of on each component or field.
- Use the component's Disabled attribute instead of conditional visibility. If you don't want your screen flow users to fill out a screen component's input fields, but it's OK for them to see the fields, you can disable the component input fields. Set the screen component's Disabled attribute by selecting a resource with a Boolean value. For example, set the Disabled attribute to a Formula resource that evaluates to `true` if another screen component is empty.
- Use reactive formulas or Screen Actions. If you want to conditionally hide and set the default value of a screen component dynamically, depending on the value of another screen component, you don't need to add a screen component for each value. Instead, use a single conditionally visible screen component and use a reactive formula or screen action to populate the default value. See Reactive Screen Flow Formula Operators.

SEE ALSO:

Make Flow Screens Dynamic with Conditional Visibility

Considerations for Flow Choice Components with Default Values

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Understand how to set a default value using any flow resource for a screen flow's choice component, such as Radio Buttons or a Multi-Select Picklist component.

The Default Value field appears below the choice options list when you add at least one choice. For the Default Value field, specify a picklist value or another flow resource (a variable, a field on a record variable, a manually entered value, and so on). You can choose any compatible type reference for the flow. You can use a value from a record as the default value, which applies to picklist choices or record choices.

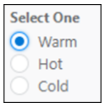
When you save and run a flow, the default value determines which options are preselected. None of the choice options are duplicated, and the order of the choices is retained.

Flow run time behavior for a default choice option

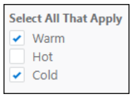
At run time, Salesforce preselects a choice if its value matches the component's default value. When the default value references a flow resource, Salesforce resolves that reference before matching. When you save and run the flow, the default value is preselected in the resulting list of choices.

If your default value is...	Then at run time Salesforce preselects choice options based on...
A choice resource that's included in the list of choice options for the component	A match for the selected choice's API name.
A choice resource that's *not* included in the list of choice options for the component	The resolved value of the choice resource. If a given choice's value matches the resolved default value, then that choice is selected.
Another resource in the flow such as a reference to a record variable from a Get Records element	The resolved value of the flow resource. If a given choice's value matches the resolved default value, then that choice is selected.
A manually entered value	The manually entered value. If a given choice's value matches the manually entered value, then that choice is selected.

For choice components that let the user select a single option, such as Picklists and Radio Buttons, Salesforce preselects the first choice that matches:



For choice components that let the user select multiple options, like Multi-Select Picklists and Checkbox Groups, Salesforce preselects every choice that matches:



Multiple default values for a choice component

To specify multiple default values for choice components that let the user select multiple options, separate the values with semicolons. If the resolved default value includes semicolons, like "Red;Blue", Salesforce treats each value as a separate default. For example, to set the default value to both "Red" and "Blue", enter `Red;Blue`. At run time, Salesforce preselects every choice option whose value is Red or Blue. Using values that contain semicolons can cause issues with multi-select value matching behavior. If a choice's value is an exact match, like Red;Blue, then Salesforce doesn't select it.

If you configure a choice component with multiple records, use a variable whose value resolves to the record ID for each record, and separate the IDs with a semicolon.

Default values from collection choice sets

If you reference a value from a collection choice set as the default value of a choice component, the default value is null when the screen loads. To trigger the screen to reload and display the value at runtime, reference the value on the same screen or wrap the value in a formula.

Flow Variable Considerations

Before you create a variable resource, understand the behavior of variables in flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Referring to Blank Fields or Resources

- If you leave a field or resource value blank, the value is `null` at run time. To treat a text value as an empty string instead of `null`, set it to `{!$GlobalConstant.EmptyString}`.

Boolean Variables

- Boolean Types Treat `null` Differently than `false`
- A flow treats `null` as a different value than `false`. For example, if you try to find a record whose checkbox field is set to `null`, no records are returned. Instead, look for records where the checkbox field is set to `false`. If you're using a variable (such as `myCheckbox = {!varBoolean}`), make sure that the variable isn't set to `null` before you reference it in your record filter or condition.

Percentage Variables

- If a flow uses record variables to manipulate percentage values, test the flow carefully. When you insert a value into a record variable's percentage field and then reference that field in a formula, the value is divided by 100.

  For example, an opportunity's Probability field is set to 100. If you assign that value to the `{!Opportunity.Probability}` record variable, the value is still 100. But if you create a formula whose expression is `{!Opportunity.Probability}`, the value is 1.

Available for Input/Output

- Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, subflows, and processes.

Setting Input Variables

- Process Builder: When a process or flow launches another flow, that flow's input variables can be assigned values during the launch. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of `null` is converted to an empty string.
- Actions: Flow actions let you pass the value of the record's ID field into the flow, but that's it. If your flow has a Text input variable called recordId, the action passes the record's ID into that variable at runtime. If not, it doesn't and the flow tries to run anyway.
- Lightning App Builder: Collection variables, record variables, and record collection variables aren't supported. The Flow component supports only manually entered values for input variables. Text input variables accept a maximum length of 4,000 characters.

Distributing Flows

- When you distribute a flow, don't pass a currency field value from a Salesforce record into a flow currency variable with a URL parameter. When a currency field is referenced through a merge field (such as `{!Account.AnnualRevenue}`), the value includes the unit of currency's symbol (for example, $). Flow currency variables can accept only numeric values, so the flow fails at run time. Instead, pass the record's ID to a text variable with a URL parameter. Then in the flow, use the ID to look up that record's value for the currency field.

Number Variables

- Number variables are treated as integers by default.

Multi-Select Resource and Screen Field Considerations for Flows

Checkbox Group, Multi-Select Picklist, and Choice Lookup screen components let flow users select multiple choices. Before you start using these screen components, understand how they work in flows—both when you design the flow and when your users run it.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configuring a Checkbox Group, Multi-Select Picklist, or Choice Lookup Screen Component

- These screen components support only one default value. You can't individually select multiple default values. However, you can manually add a value in the default value field and separate each value with a semicolon.
- You can configure a record choice set resource to assign field values from a user-selected record to variables in the flow. When a Checkbox Group, Multi-Select Picklist, or Choice Lookup screen component uses a record choice set, only values from the last record that the user selects are stored in the flow variables. If multiple Checkbox Group, Multi-Select Picklist, or Choice Lookup components on one screen use the same record choice set, the variable assignments come from the last record selected from all of those components.

Using Values from a Checkbox Group, Multi-Select Picklist, or Choice Lookup Screen Component

- At run time, the value of a Checkbox Group, Multi-Select Picklist, or Choice Lookup screen component is a concatenation of the user-selected choice values, separated by semicolons. If a selected choice's value includes semicolons, the semicolons are removed.
- If you reference a Checkbox Group, Multi-Select Picklist, or Choice Lookup screen component in a flow condition:
  - Make sure that each choice in the screen component has a choice value configured.
  - Don't use the same choice in multiple Checkbox Group, Multi-Select Picklist, or Choice Lookup screen components on the same screen.
- If a Checkbox Group, Multi-Select Picklist, or Choice Lookup has at least one default value, at run time the choices are preselected if the choice's value matches the default value.

SEE ALSO:

Using Choice Resources with Flow Screen Components

Paused Flow Interview Considerations

Before you design flows that contain one or more Wait elements, understand the behavior and guidelines.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Time-based resume events are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Platform event-based resume events are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

General Considerations

- After you deactivate a flow version, its paused interviews continue to wait for the configured resume events. If a flow version has paused interviews, you can't delete it.
- An interview can execute only one connector per Wait element. After one of its resume events is processed, the remaining resume events are removed from the queue.
- If the user who started the interview is deactivated when Salesforce tries to execute a wait connector, the interview fails to resume.
- If a flow is paused and the flow interview exceeds 1 MB, the interview fails to save and can't be resumed.
- You can't call flows that contain wait elements as subflows.

Transactions and Paused Interviews

A transaction ends as soon as a flow interview pauses for one or more resume events. When the flow interview resumes, a new transaction begins. Everything after the Wait element is executed as part of a batch transaction that includes other resumed interviews.

Interviews aren't resumed independently. They're grouped into a single batch that starts resuming within one hour after the first interview enters the batch. Actions that execute as a result of the grouped interviews are also executed in that transaction. The batch can have other interviews that resume at the same time, have the same flow version ID, and are executed by the same user ID. This behavior can cause you to exceed your Apex governor limits if the resumed interview executes DML operations or SOQL queries through. For details, see Per-Transaction Flow Limits on page 246.

- Flow elements, such as Create Records or Apex Action (Legacy)
- Apex triggers
- Immediate workflow actions

If a Wait element precedes a flow element that executes DML operations or SOQL queries:

- Ensure that your flows don't let a single user execute DML operations or SOQL queries that can exceed limits between Wait elements.
- Consider using multiple Wait elements so that the DML operations and SOQL queries are performed in multiple transactions.
- Add fault paths for those elements so that the flow returns to the Wait element if the fault message contains: `Too many SOQL queries` or `Too many DML operations`.

If an interview fails after it's resumed:

- Prior interviews in that batch's transaction are successful.
- Operations that the interview executed before it paused are successful.
- If a fault path handles the failure, operations that the interview executed between when it resumed and when it failed are successful. The operation that caused the interview to fail isn't successful.
- If a fault path doesn't handle the failure, operations that the interview executed between when it resumed and when it failed are rolled back. The operation that caused the interview to fail isn't successful.
- The remaining interviews in that batch are tried.

Platform Events

Tip: Make sure to also review the considerations and allocations for platform events.

- Supported Platform Events

  Flows can subscribe to custom platform events and these standard platform events.

  - AIPredictionEvent
  - BatchApexErrorEvent
  - FlowExecutionErrorEvent
  - FOStatusChangedEvent
  - OrderSummaryCreatedEvent
  - OrderSumStatusChangedEvent
  - PlatformStatusAlertEvent
- Formulas—To reference a platform event in a formula, pass the event data into a record variable in the Wait element. Then reference the appropriate field in that record variable.
- Value Truncation—When you filter platform event messages, values for conditions can't be more than 765 characters.
- Subscriptions Related List—On the platform event's detail page, the Subscriptions related list shows which entities are waiting to receive that platform event's messages. The related list includes a link to each subscribed process. If flow interviews are waiting for that platform event's messages, one "Process" subscriber appears in the Subscriptions related list.
- Uninstalling Events—Before you uninstall a package that includes a platform event, delete the interviews that are waiting for that platform event's messages.
- Einstein Predictions—A prediction event is sent for each Einstein prediction result, so use event condition filters if you want your flow to be triggered only by predictions on a specific object. For example, if your flow uses a Wait element that acts only on predictions written to Lead records, add a resume event to check that the AIPredictionEvent.TargetId field equals the current record.

  If your flow updates a field that is used by an Einstein prediction, Einstein runs the prediction again and writes back the new results. The new results generate a new prediction event that could trigger your flow again, resulting in a loop. To avoid creating a loop, only update fields that aren't used in Einstein predictions.

Platform Cache

When a flow contains a Wait element, make sure that later elements in the flow don't invoke Apex code that stores or retrieves values from the session cache. The session-cache restriction applies to Apex actions and to changes that the flow makes to the database that cause Apex triggers to fire.

Time-Based Resume Events

- Time-based resume events don't support minutes or seconds.
- If an interview is waiting for a time in the past, Salesforce resumes the interview as soon as possible. Depending on how many actions Salesforce is processing at the time, actions are executed within one hour.

  For example, a flow is configured to email an opportunity owner seven days before the close date. An interview starts for an opportunity with the close date set to today. Salesforce resumes the interview within an hour.
- An org can process up to 1,000 time-based resume events per hour. When a resume event is processed, its associated interview resumes and any other resume events for that interview are removed from the queue. If an org exceeds this limit, Salesforce defers the remaining resume events to be processed in the next hour.

For example, an org has 1,200 resume events scheduled to be processed between 4:00 PM and 5:00 PM. Salesforce processes 1,000 resume events between 4:00 PM and 5:00 PM and the additional 200 resume events between 5:00 PM and 6:00 PM.

- You can't archive a product or price book that's referenced in a time-based resume event in a paused interview.

Flow-Based Time

For resume events based on a specific time, the resume time is evaluated using the time zone of the user who created the flow.

Record-Based Time

- For resume events based on a record field value, the resume time is evaluated using the org's time zone.
- Resume events can't reference:
  - `DATE` or `DATETIME` fields that contain automatically derived functions, such as *`TODAY`* or *`NOW`*.
  - Formula fields that include related-object merge fields.
- If you change a date field that's referenced by an unexecuted resume event in a paused interview, Salesforce recalculates the resume events associated with the interview.

  For example, a flow is configured to email an opportunity owner seven days before the opportunity close date, and the close date is 2/20/2014. The following things could happen.
  - The close date isn't updated before the interview resumes. Result: Salesforce resumes the interview on 2/13/2014 and sends the email.
  - The close date is updated to 2/10/2014 before the interview resumes. Result: Salesforce reschedules the resume event and the interview resumes on 2/3/2014.
  - The close date is updated to a date in the past. Result: Salesforce recalculates the resume event and resumes the interview shortly after you save the record.
- If a resume event references a null date field when the interview executes the Wait element, Salesforce resumes the interview as soon as possible. Depending on how many actions Salesforce is processing at the time, actions are executed within one hour.
- If a resume event references a date field that has a non-null value when the flow interview executes the Wait element and it's updated to `null` before the resume event is processed, Salesforce resumes the interview within an hour after the date field is updated.
- If a record or object that's referenced by a resume event is deleted, the resume event is removed from the queue. If the interview has no other resume events to wait for, the interview is deleted.
- Lead Convert Limitations
  - You can't convert a lead that's referenced in a paused interview's resume event.
  - If Validation and Triggers from Lead Convert is enabled, existing operations on leads after a Wait element aren't executed during lead conversion.
  - If a campaign member based on a lead is converted before a paused interview that's associated with that record finishes, Salesforce still executes the interview.

SEE ALSO:

Platform Events Developer Guide : Considerations for Defining and Publishing Platform Events

Flow Limits and Considerations

Flow Operators in Decision, Wait, and Collection Filter Elements

Flow Elements: Wait

Platform Events Developer Guide : Subscribe to Platform Even Messages with Flows

Flow Stage Considerations

Before you add stages to your flow, understand how stage references and default active stages work, as well as considerations for troubleshooting stages.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Stage References

When you reference a stage merge field in a display text field or other label, it resolves to the stage's label. Everywhere else, a stage merge field resolves to the stage's fully qualified name: `namespace.flowName:stageName` or `flowName:stageName`.

Whenever possible, use the stage merge field to refer to stages, such as {!myStage}. When you reference a stage in a subflow, use the fully qualified name.

Default Active Stages

When you mark a stage resource **Active by Default**, the flow automatically sets values for the global variables. Use this setting when a stage applies to every branch of the flow.

At run time, the default active stages are sorted in ascending order. How the flow uses the default active stages to update `$Flow.ActiveStages` and `$Flow.CurrentStage` depends on whether the flow is a parent flow or a referenced flow.

Parent Flows

The default active stages are added to `$Flow.ActiveStages` in ascending order. `$Flow.CurrentStage` is set to the default active stage with the lowest order.

When a flow references two flows, one with stages and one without, configure the flow with stages so that it sets the value for `$Flow.ActiveStage` to null at the end of the flow. Then set the value for `$Flow.CurrentStage` to stage 1 at the start of the flow.

Referenced Flows

The default active stages are inserted in `$Flow.ActiveStages` in ascending order. `$Flow.CurrentStage` isn't automatically updated.

- When `$Flow.CurrentStage` is included in `$Flow.ActiveStages`, the default active stages are inserted in `$Flow.ActiveStages` after `$Flow.CurrentStage`.

  For example, Flow1 sets `$Flow.ActiveStages` to "1, 2, 3, 4" and `$Flow.CurrentStage` to "3." It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, `$Flow.ActiveStages` becomes "1, 2, 3, A, B, C, 4." `$Flow.CurrentStage` is still "3."

- When `$Flow.CurrentStage` isn't included in `$Flow.ActiveStages`, the default active stages are added to the end of `$Flow.ActiveStages`.

  For example, Flow1 sets `$Flow.ActiveStages` to "1, 2, 3, 4" and doesn't set `$Flow.CurrentStage`. It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, `$Flow.ActiveStages` becomes "1, 2, 3, 4, A, B, C." `$Flow.CurrentStage` remains unset.

- When `$Flow.CurrentStage` is duplicated in `$Flow.ActiveStages`, the default active stages are appended after the first occurrence.

  For example, Flow1 sets `$Flow.ActiveStages` to "1, 2, 2, 3, 4" and `$Flow.CurrentStage` to "2." It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, `$Flow.ActiveStages` becomes "1, 2, A, B, C, 2, 3, 4." `$Flow.CurrentStage` remains "2."

Troubleshooting Stages

The flow error email doesn't specify the values of `$Flow.ActiveStages` and `$Flow.CurrentStage` at the start of an interview. To confirm what the initial values are, add temporary elements to display the initial values, such as in a screen display text field.

SEE ALSO:

Show Users Progress Through a Flow with Stages

Flow Resource: Stage

Two-Column Flow Considerations

If your org has Lightning runtime enabled, you can control whether a flow displays in one column or two columns. Before you use this feature, understand how the flow layout currently behaves.

Note: Starting in Winter '23, two-column flow layouts are ignored. For a better layout option, add Section components to your flow screens. Each Section component lets you organize record fields and screen components in up to four adjustable-width columns.

These considerations don't apply to the Section component in flow screens.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Granularity

The layout setting is applied at the flow level. So you can't control the layout at the screen or field level. If you set a flow to use two columns, every screen in that flow displays in two columns.

Order of Fields

You can't manually control which fields go in which columns. If the flow is set to display two columns, the fields alternate in each column. The odd fields (first, third, fifth, and so on) are placed in the left column. The even fields (second, fourth, sixth, and so on) are placed in the right column.

If your users navigate screens with the Tab key, they tab through all the fields in the left column and then all the fields in the right column. You can't configure the fields to tab left-to-right.

Responsiveness

The flow layout isn't responsive to the user's screen dimensions. It uses the same layout whether the user's screen is 1 inch wide or 20 inches wide.

Tip: If users run a flow from a phone or small tablet, don't apply a two-column layout to the flow.

Compatibility with Section component

For flows that are distributed via Experience Builder, the Lightning App Builder, or the utility bar, each flow screen that contains a Section component ignores the Layout property.

For flows that are distributed via URL, each flow screen that contains a Section component ignores the `flowLayout` URL parameter.

SEE ALSO:

Flow Limits and Considerations

Flow Screen Output Component: Section

Customize a Flow URL to Render Two-Column Screens

Schedule-Triggered Flow Considerations

A schedule-triggered flow starts at the specified time and frequency for a batch of records. Understand the considerations and special behaviors of schedule-triggered flows, also known as scheduled flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

These considerations apply to schedule-triggered flows.

- A schedule-triggered flow starts at the specified time and frequency. You can't launch a schedule-triggered flow by any other means.
- The Start Time field value is based on the Salesforce org's default time zone.
- The View All Data permission is required to activate an autolaunched flow that has a trigger.
- The maximum number of schedule-triggered flow interviews per 24 hours is 250,000, or the number of user licenses in your org multiplied by 200, whichever is greater. One interview is created for each record retrieved by the schedule-triggered flow's query.

  If you specify an object so that the flow runs for a batch of records, then set the time, frequency, and record conditions to avoid reaching this limit. The maximum limit of records per batch is 200. You can use debug logs to check how many records a schedule-triggered flow runs on. Track the number of records with the SCHEDULED_FLOW_DETAIL event. If your org reaches the limit, Salesforce sends a flow error email.
- If you delete a schedule-triggered flow from the Scheduled Jobs page in Setup, all future recurrences of that flow are canceled. To enable future runs, deactivate and reactivate the flow.
- If a flow is scheduled to run one time with a date and time that already passed, the flow doesn't run.
- The Default Workflow User runs schedule-triggered flows.
- If you need a schedule-triggered flow to invoke Apex code, don't enable the Require User Access to Apex Classes Invoked by Flow update. When that release update is activated, schedule-triggered flows fail when they invoke Apex.
- A schedule-triggered flow can make callouts only after executing a Wait element. For example, without a Wait element, the flow can't access external objects, execute Apex actions that make callouts, or execute actions that are generated from External Services registrations.

  Tip: You can insert a Wait element that pauses the flow for only a moment. Configure the resume event to pause until a specified time, with a specific time as the time source. For the base time, specify the `$Flow.CurrentDateTime` global variable. Then set the offset to 0 hours. At run time, a Wait element that's set up this way typically pauses the flow for less than a minute.
- If you configure an Update Records element to use the ID and all field values from the $Record global variable, enable `Filter inaccessible fields from flow requests` in your org's process automation settings. Otherwise, the flow fails because the Update Records element tries to set the values for system fields and other read-only fields.
- Synchronous Apex transactions invoked by an asynchronous flow contribute to synchronous per-transaction Apex limits. Asynchronous flows include scheduled flows and flows with scheduled or asynchronous paths.

- At run time, if a schedule-triggered flow has a Create Records, Delete Records, Get Records, or Update Records element that processes multiple records and some records fail, all records are rolled back. The successful records are retried. If the flow has another Create Records, Delete Records, Get Records, or Update Records element later in the flow that processes the same failed records, all changes are rolled back and the flow transaction fails.
- The order of your filter conditions doesn't matter. The SFDC Optimizer evaluates all filters to optimize performance.

SEE ALSO:

Schedule Triggers for Flows That Run for Batches of Records

Troubleshoot Flow Errors

Flow Operations and Read-Only Fields

Considerations for Troubleshooting Flows

Record-Triggered Flow Considerations

A record-triggered autolaunched flow makes additional updates to the triggering record before or after it's saved to the database. Understand the considerations and special behaviors of flows that make before- and after-save updates.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

General Considerations

These considerations apply to any record-triggered flows.

- Record-triggered flows run custom validation rules.
- You can't reference a screen flow from an autolaunched flow.
- The `isChanged` operator isn't supported on asynchronous paths.
- Due to their position in the order of execution, record-triggered flows can behave differently from similar workflow rules.
- Flows that run only when a record is updated to meet the condition requirements are triggered only if all the condition requirements change from `false` to `true`. If all the condition requirements already evaluate to `true` and still evaluate to `true` after the record is updated, the flow doesn't run. Scheduled paths are scheduled only if the previous version of the record didn't meet the requirements, and the updated record does meet the requirements.

  For example, a record-triggered flow that is set to trigger when a flow is created or updated has the condition Industry equals Agriculture. The flow is set to run only when a record is updated to meet the condition requirements.

Scenario	Result
A new account where Industry = Agriculture	The flow is triggered. Scheduled paths are scheduled.
A new account where Industry = Finance	The flow isn't triggered. Scheduled paths aren't scheduled.
An existing account where Industry = Agriculture Is updated to: – Industry = Agriculture – Billing State = CA	The flow isn't triggered. The condition was already met before the record was updated. Scheduled paths aren't scheduled, but already-scheduled paths stay scheduled.
An existing account where Industry = Finance is updated to Industry = Agriculture	The flow is triggered. The condition wasn't met before, but is satisfied after the record update. Scheduled paths are scheduled.

Scenario	Result
An existing account where Industry = Agriculture is updated to Industry = Finance	The flow isn't triggered. The condition wasn't met after the record update. Because the record doesn't meet the condition, any scheduled paths are also canceled.

In this example, a record-triggered flow that is set to trigger when a flow is created or updated has the conditions Industry equals Agriculture OR Billing State equals CA. The flow is set to run only when a record is updated to meet the condition requirements.

Scenario	Result
An existing account where: - Industry = Agriculture - Billing State = NJ Is updated to: - Industry = Agriculture - Billing State = CA	The flow isn't triggered. The conditions were already met before the record was updated. Scheduled paths aren't scheduled, but already-scheduled paths stay scheduled.
An existing account where: - Industry = Finance - Billing State = NJ Is updated to: - Industry = Agriculture - Billing State = NJ	The flow is triggered. The conditions weren't met before, but are satisfied after the record update. Scheduled paths are scheduled.
An existing account where: - Industry = Finance - Billing State = CA Is updated to: - Industry = Finance - Billing State = NJ	The flow isn't triggered. The conditions weren't met after the record update. Because the record doesn't meet the conditions, any scheduled paths are also canceled.

Considerations for Fast Field Updates

These considerations apply to record-triggered flows that are optimized for fast field updates (before-save).

- The flow can't perform actions other than updating the triggering record's field values.
- The flow can't update values in records that are related to the triggering record.
- Only these elements are supported: Assignment, Decision, Get Records, and Loop.
- The View All Data permission is required to activate an autolaunched flow that has a trigger.

Considerations for Debug Mode

- The ISCLONE() formula function always evaluates to FALSE when you're in debug mode. For example, if a record-triggered flow contains an ISCLONE() formula function in the entry criteria or in a Decision element, ISCLONE() evaluates to FALSE even when you're debugging with a cloned record.

SEE ALSO:

Record Triggers for Flows That Make Before-Save Updates

Apex Developer Guide : Triggers and Order of Execution

Flow Data Considerations

When designing flows, keep these data considerations in mind.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Limits

- Each flow interview that executes the flow element Get Records or Update Records enforces the SOQL query limit for the maximum number of characters because the element uses a SOQL query. For each element per flow interview, the SOQL query limit is 100,000 characters. For example, a flow interview executes the Get Records element that uses the In operator on a collection of account IDs. If the element contains a collection of account IDs that exceeds 4,700 IDs and specifies other criteria to exceed the 100,000 character limit, the flow interview can fail.

Permissions

- For flows that interact with the Salesforce database, make sure that your users have permission to create, read, edit, and delete the relevant records and fields. Otherwise, users receive an insufficient privileges error when they try to launch a flow. For example, a flow looks up and updates a case record's status. The flow users must have Read and Edit permissions on the Status field of the Case object.

Variables

- If you delete a record variable or record collection variable, variable assignments that use the deleted variable are set to null.
- When a process or flow launches another flow, that flow's input variables can be assigned values during the launch. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of null is converted to an empty string.
- Storing field values automatically in the Get Record element is available only for screen flows and autolaunched flows.

Date and Date/Time

- At run time, time zones for date/time values can differ from what you see in Flow Builder. During run time, date/time values reflect the time zone settings of the user who's running the flow.

Flow Lightning Runtime Considerations

When running flows, keep these considerations in mind.

Note: In Lightning runtime, flow users always run the active flow version. Flow admins with the Manage Flow permission run the latest flow, so they can test the latest flow version without activating it for flow users. A flow admin also runs the latest flow version that is referenced via a Subflow element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Interviews

A *flow interview* is an instance of a flow, much like a record is an instance of an object. The flow interview can do many things, including look up and manipulate Salesforce data. In an interview, you can pass data into variables and other resources. The data can come from a variety of sources, such as Salesforce records that the flow queries, information that a user enters in a screen input field, or something that you manually enter.

Interviews don't perform actions—such as sending emails or creating, editing, or deleting records—until the associated transaction is complete. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element. In addition to data elements, the Post to Chatter, Submit for Approval, and Quick Actions core actions also create and update records.

When an interview is in flight, the data in the interview isn't saved to the Salesforce database. If the flow executes an element that creates or updates records, such as Update Records or Post to Chatter, only the information configured in that element is saved to the Salesforce database.

When an interview executes a Wait element or a user pauses it, all the interview data is serialized and saved to the database as a Paused Flow Interview record. When the interview resumes, the Paused Flow Interview record is deleted.

Limitations of Lightning Runtime for Flows

When Lightning runtime is enabled for your Salesforce org, flows in Lightning Experience don't load in:

- Web tabs
- List buttons that are set to display an existing window with or without a sidebar

When Lightning runtime is enabled for your org, flows in Salesforce Classic don't load in custom buttons or links that are set to display in an existing window with or without a sidebar.

In number input fields, users can enter up to 17 digits, including digits before and after a decimal point.

At runtime, validation error messages persist on screen flow components even if a user corrects the errors. The user can complete the flow interview despite the messages.

SEE ALSO:

Lightning Runtime vs. Classic Runtime for Flows

Flow Element: Subflow

Flow Runtime Accessibility Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

We strive to make the run-time experience of screen flows follow the best practices in Section 508 of the Rehabilitation Act and the Web Content Accessibility Guidelines (WCAG) 2.0 Level AA. But we have some exceptions.

If you use screen readers or other assistive technology while running screen flows, consider these known issues.

- The title of the screen doesn't change when you click Next or Previous, so it's not always obvious that you've switched to a different page.
- Flow screen components that don't have defined labels can't be read properly by assistive technology.
- Unless you use the ARIA alert role or another method of identifying errors for assistive technology, these types of custom error messages can't be detected by assistive technology.
  - Error messages that are text components with conditional visibility
  - Error messages that are displayed for components when the associated Validate Input formula expression evaluates to false
- Screen readers base pronunciation on their language setting. When that language setting differs from the flow's language, screen readers can't correctly read the flow screens. This limitation affects flows run from:
  - Run and Debug buttons in Flow Builder
  - URLs
  - Custom buttons and links
  - Web tabs
- Error messages for some Salesforce-provided components aren't associated with their corresponding input fields. This limitation means that screen readers can't read error messages associated with them. Affected Salesforce-provided components:
  - Dependent Picklists
  - Email
  - Lookup
  - Phone
  - Toggle
  - URL
- When a user doesn't complete a required field in a Dependent Picklists component, the resulting error messages can't be read by screen readers or other assistive technology. Sometimes, one of the error messages is announced one time, but later attempts to focus on the field don't cause the error message to be announced again.
- When a user clicks Finish in the Resume window from the Paused Flow Interviews Lightning component on a desktop (LEX), focus isn't set to the Refresh icon button.
- When a flow screen is initially displayed, the focus is set to the first visible screen field. Exceptions:
  - If the flow screen contains an error, the focus is set to the first field with an error.
  - If the flow screen contains only Display Text components, the focus is set to the body of the flow.

- When screen components or their parts are rendered after the screen is initially displayed, they're never focusable. For example, if a component asynchronously fetches a list of tasks to display, the focus can't be set to any of the tasks. If a screen component uses conditional visibility and appears only after user input, the focus can't be set to any part of the screen component.

SEE ALSO:

Flow Limits and Considerations

Flow Screen Input Component: Display Image

Flow Management Considerations

When managing flows, consider the administration and activation limits.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Viewing Flows

In Lightning Experience, the Flows page in Setup doesn't display any flows if a user sets the Sharing Settings of the All Flows list view to Only I can see this list view.

Activating Flows

When you activate a new version of a flow, the previously activated version (if one exists) is automatically deactivated. Any running flow interview continues to run using the version with which it was initiated.

Deleting Flows

To delete an active flow version, first deactivate it. If a flow has paused interviews, it can't be deleted until those interviews are finished or deleted. You can delete flows that have never been activated at any time.

Flow Type

If a flow has versions with different types, the active (or latest) version determines the flow type.

Deploying Flows

In production orgs, you can enable the setting to deploy a new active version of a process or flow via change sets or Metadata API. The setting doesn't appear in non-production orgs (such as scratch, sandbox, and developer orgs), because you can always deploy a new active version.

SEE ALSO:

Deploy Processes and Flows as Active

Flow Limits and Considerations

Considerations for Packaging Flows

You can include a flow in a managed or unmanaged package. Before you create, update, or deploy a package that contains a flow, understand the limitations and behaviors of packages.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Creating Packages

- If you want to deploy a flow with a change set, the change set must include all components that the flow references.
- When you package a flow, all components and fields that the flow references must be available in the same package or a dependent package.
- If these elements are used in a flow, the packageable components that they reference aren't included in the package automatically. To deploy the package successfully, manually add the referenced components to the package.
  - Post to Chatter
  - Send Email
  - Submit for Approval

  For example, if you deploy a flow that posts to a particular Chatter group, manually add the referenced Chatter group to the package.
- If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package automatically.
- When you upload a package or package version, the active flow version is included. If the flow doesn't have an active version, the latest version is packaged.

Updating Packages

- To update a managed package with a different flow version, activate that version and upload the package. Or, deactivate all versions of the flow, make sure that the latest flow version is the one to distribute, and then upload the package. If you activate a flow version by mistake and upload the package, that flow version is distributed to everyone.
- If you install a flow from an unlocked package with the same API name, the new flow overrides the existing flow in the target org.
- You can't include flows in package patches.

Other Considerations

- Flow Builder displays Apex actions from managed packages only if the associated method is marked global.
- Flow Builder displays email alerts from managed packages only if the email alert isn't protected.
- If you register your namespace after you reference a flow in a Visualforce page or Apex code, add the namespace to the flow name before you install the package.
- If a flow is installed from a managed package, error emails for that flow's interviews don't include details about the individual flow elements. The email is sent to either the user who installed the flow or the Apex exception email recipients.
- You can't package flow triggers.
- In a packaging org, you can't delete a flow after you upload it to a released or beta first generation managed package. You can delete a flow version from a packaging org after you upload it to a released or beta first-generation managed package, if all these criteria are met:
  - Salesforce Customer Support activated the Managed Component Deletion permission.
  - The flow version isn't the most recently packaged version of the flow.

  - The flow version isn't active.
  - The flow version is not the only version.

- Images in rich text for screens aren't supported in packages.

Change Set Considerations for Flows

Before you use change sets to deploy a flow, understand the limits and behaviors that are related to component dependencies, deployment, and flow triggers.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Creating Change Sets

- If you want to deploy a flow with a change set, the change set must include any component the flow references.
- When you view the dependent components for the change set, the Component Dependencies page lists the dependencies for *all* versions of the flow. Add all interdependent components for the relevant flow version to the outbound change set.
- If a flow element references these components, the Component Dependencies page doesn't display that component. To deploy the flow successfully, manually add those referenced components to the change set.
  - Post to Chatter
  - Send Email
  - Submit for Approval

  For example, if you deploy a flow that includes a Submit for Approval element, manually add the referenced approval process.
- If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package or change set.

Deploying Change Sets

- You can include only one version of a flow in a change set.
- An active flow in a change set is deployed to its destination as inactive. Manually activate the flow after deployment.
- If the flow has no active version when you upload the outbound change set, the latest inactive version is used.
- Deploying or redeploying a flow with a change set creates a version of the flow in the destination Salesforce org.
- In production orgs, you can enable the setting to deploy a new active version of a process or flow using change sets or Metadata API. The setting doesn't appear in non-production orgs (such as scratch, sandbox, and developer orgs), because you can always deploy a new active version.

Flow Triggers

- Flow triggers aren't available in change sets.

SEE ALSO:

Deploy Processes and Flows as Active

Considerations for Flows Installed from Packages

Keep these considerations in mind when you distribute, upgrade, or remove a flow that you installed from a package.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Flow Builder can't open a flow that is installed from a managed package, unless the flow is a template or overridable.
- If you install a managed package that contains multiple flow versions in a fresh destination org, only the latest flow version is deployed.
- If you install a non-template flow from a managed package, error emails for that flow's interviews don't include any details about the individual flow elements. The email is sent to either the user who installed the flow or the Apex exception email recipients.
- If you install a flow from an unmanaged package that has the same name but a different version number as a flow in your org, the newly installed flow becomes the latest version of the existing flow. However, if the packaged flow has the same name and version number as a flow already in your org, the package install fails. You can't overwrite a flow.
- If you install a flow from an unlocked package that has the same name as a flow in your org, the newly installed flow overrides the existing flow.

Status

An active flow in a package is active after it's installed. The previous active version of the flow in the destination org is deactivated in favor of the newly installed version. Any in-progress flows based on the now-deactivated version continue to run without interruption but reflect the previous version of the flow.

Distributing Installed Flows

- When you create a custom button, link, or web tab for a flow that's installed from a managed package, include the namespace in the URL. The URL format is `/flow/namespace/flowuniquename`.
- When you embed a flow that's installed from a managed package in a Visualforce page, set the name attribute to this format: `namespace.flowuniquename`.

Upgrading Installed Flows

Upgrading a managed package in your org installs a new flow version only if there's a newer flow version from the developer. After several upgrades, you can end up with multiple flow versions.

Removing Installed Flows

- You can't delete a flow from an installed package. To remove a packaged flow from your org, deactivate it and then uninstall the package.
- In a packaging org, you can't delete a flow after you upload it to a released or beta first generation managed package. You can delete a flow version from a packaging org after you upload it to a released or beta first-generation managed package, if all these criteria are met:
  - Salesforce Customer Support activated the Managed Component Deletion permission.
  - The flow version isn't the most recently packaged version of the flow.
  - The flow version isn't active.
  - The flow version is not the only version.

- If you have multiple versions of a flow installed from multiple unmanaged packages, you can't remove only one version by uninstalling its package. Uninstalling a package—managed or unmanaged—that contains a single version of the flow removes the entire flow, including all versions.
- Delete flows from an unlocked package manually—you can't delete them by removing them from the unlocked package.

Translating Installed Flows

You can translate flow definition names only on the Translate page.

SEE ALSO:

Use Managed Packages to Develop Your AppExchange Solution

Select Flow and Process Error Email Recipients

Considerations for Packaging Flows

Salesforce DX Developer Guide Components Available in Managed Packages

First-Generation Managed Packaging Developer Guide: Install a Managed Package

Select Flow and Process Error Email Recipients

Considerations for Troubleshooting Flows

Keep these considerations in mind when reviewing a flow error email or using the debug option in Flow Builder.

Be careful when debugging flows that contain delete elements. Even if the flow is inactive, it triggers the delete operation.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Debugging a Flow

- If you debug a flow without choosing to run the flow in rollback mode, the flow performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.
- You can't pass values into input variables of type collection, record, and record collection.
- Clicking **Pause** or executing a Wait element closes the flow and ends debugging.
- When you debug a flow as another user, the flow's record changes and actions are performed as that user. Also, the user's profile and permission sets determine the object permissions and field-level access of the flow. However, flows that always run in system context ignore the user's object permissions and field-level access.
- When you click **Finish** in a flow, the debug details incorrectly state "Selected Navigation Button: NEXT."
- When you debug a schedule-triggered flow, the flow starts only for one record.
- When you debug a record-triggered flow, only what's within the flow is tested. This smaller scope can lead to scenarios where the flow executes as intended while debugging, but not at run time. This behavior difference can be due to other triggered flows and processes. To see how a record-triggered flow behaves in a real-world scenario, make sure to test it in a sandbox org.

Tracking More Information About a Flow Interview

- To store more information about an interview when it's saved as a Salesforce record, build a custom object that references the interview's GUID. An interview is assigned an 18-character Salesforce ID only when it's paused and saved as a Salesforce record. Each interview, whether in-flight or paused, has a GUID.

Troubleshooting Stages

The flow error email doesn't specify the values of `$Flow.ActiveStages` and `$Flow.CurrentStage` at the start of an interview. To confirm what the initial values are, add temporary elements to display the initial values, such as with a text field.

Considerations for Flow Error Emails

Review these considerations for the email sent to the admin or Apex exception email recipients regarding flow interview errors and Screen and Subflow elements.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

General

- If the user who started the flow doesn't have a first name, `null` replaces the user's first name in the How the Interview Started section.
- Variable assignments display in this pattern: `{!variable} (prior value) = field/variable (new value)`. If the variable had no prior value, the parentheses display as empty. For example: `{!varStatus} () = Status (Delivered)`
- If you install a non-template flow from a managed package, error emails for that flow's interviews don't include any details about the individual flow elements. The email is sent to either the user who installed the flow or the Apex exception email recipients.
- Failed flow interviews for these flow types that are built with the free-form layout in Flow Builder are saved and available to open in Flow Builder.
  - Screen flows
  - Record-triggered flows
  - Schedule-triggered flows
  - Autolaunched flows that aren't triggered
- Failed flow interviews aren't saved if:
  - The flow is installed as part of a managed package and isn't a template.
  - The failure occurs after the flow interview is paused and then resumed at least one time.
  - The error is handled because the element that encounters the error is connected to a fault connector.
  - The failure occurs during an Apex test method.
  - The flow is a standard flow.
  - The value of the flow's metadata field `status` is `Draft` or `InvalidDraft`.
  - The failed flow interview exceeds 1 MB.
  - The failed flow interviews already saved in the database exceeds 1 GB.
- Failed flow interviews don't count toward data, file, or paused flow interview storage limits. When failed flow interviews are saved, they're available for up to 14 days and then automatically deleted from the database.
- These limits apply when failed flow interviews are saved.
  - For any specific flow, no more than 100 failed flow interviews are saved in a 24-hour period.
  - For a batch of up to 200 failed flow interviews in the same transaction, one interview is saved.
  - Across all the flows in an organization, no more than 3,000 failed flow interviews are saved in a 24-hour period.
  - Failed flow interviews exceeding 1 MB aren't saved.
  - Failed flow interviews aren't saved if more than 1 GB of failed flow interviews are already saved in the database.

Screen elements

Password fields display in plain text

Subflow elements

- The merge field annotation (`{!variable}` as opposed to just `variable`) is missing for variables in a referenced flow. For example, when an interview enters a subflow and gives details about the inputs, the subflow's variable is `subVariable` instead of `{!subVariable}`.
- If the error occurs in a referenced flow, the email is sent to the author of the parent flow, but the subject references the name of the referenced flow.
- If you see multiple Entered flow *`ReferencedFlowName`* version *`ReferencedFlowVersion`* messages with no Exited *`ReferencedFlowName`* version *`ReferencedFlowVersion`* messages in between them, the flow user navigated backwards. To prevent this scenario, adjust the navigation options in the first screen of the referenced flow so that the user can't click **Previous**.

SEE ALSO:

Troubleshoot Flow Errors

Select Flow and Process Error Email Recipients

Run-Time Changes by Release and API Version

These versioned updates affect only flows that are configured to run on specific API versions. With versioned updates you can test and adopt run-time behavior changes for individual flows at your convenience.

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To change the run-time API version of a flow, open it in Flow Builder and edit the flow version properties.

Winter '25 (API Version 62.0)

These updates affect only flows that are configured to run on API version 62.0 and later.

Summer '24 (API Version 61.0)

These updates affect only flows that are configured to run on API version 61.0 and later.

Winter '25 (API Version 62.0)

These updates affect only flows that are configured to run on API version 62.0 and later.

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Enforce Sharing Rules when Apex Launches a Flow

This versioned update enforces sharing rules when an Apex class that's declared using the with sharing keyword launches an autolaunched flow that runs in the default context. To enforce sharing, the Apex class must be declared using the with sharing keyword.

Previously, the flow ran in system context without sharing even when an Apex class was declared using the with sharing keyword launched the flow.

With this versioned update, the flow runs more securely in the default context when an Apex class that's declared using the with sharing keyword launches an autolaunched flow. The flow enforces the sharing rules of the user that executes the Apex class. Previously, when sharing rules weren't enforced, the flow was able to access all data.

This versioned update restricts data access for autolaunched flows that are run in the default context and launched by an Apex class. The Apex class must be declared using the with sharing keyword. Data access is restricted to the sharing rules of the user that executed the Apex class.

For example, a query can return fewer rows than it did in system context without sharing. An operation can fail because the user doesn't have the correct permissions.

Set Screen Action Outputs to Null Correctly

In API version 62.0 and later, this versioned update makes sure that if a flow run by a screen action has an output that isn't set by using an Assignment element, its outputs are set to null, as expected. Screen components using that output are now updated automatically.

Set Conditionally Hidden Screen Component Outputs to Null Correctly

In API version 62.0 and later, this versioned update makes sure that if a conditionally hidden screen component has a collection as an output, its outputs are set to null, as expected.

Summer '24 (API Version 61.0)

These updates affect only flows that are configured to run on API version 61.0 and later.

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Evaluate Null Text Values

With this versioned update, a null text value evaluates to null in a flow. Previously, a null text value evaluated to an empty string value. For example, an empty picklist value evaluates to a null text value when the flow runs on API version 61.0 and later.

Flow Reference

Bookmark this page for quick access to information about flow elements, resources, events, and more.

Flow Resources

Each *resource* represents a value that you can reference throughout the flow.

Flow Elements

An element represents an action that the flow can execute. Examples include reading or writing Salesforce data, displaying information and collecting data from flow users, executing business logic, or manipulating data.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Provided Flow Core Actions

Perform an action outside of the flow. Choose from Salesforce-provided actions, like Submit for Approval or Send Email, or from your org's quick actions and local actions. To add one of these actions to your flow, add an Action element. Then, in the Action field, search for the appropriate action.

Standard Flow Screen Components

Salesforce provides several standard screen components that extend the types of input fields available in screens.

Flow Connectors

A connector determines the path that a flow takes at run time.

Flow Operators

Operators behave differently, depending on what you're configuring. In Assignment elements, operators let you change resource values. In conditions and filters, operators let you evaluate information and narrow the scope of a flow operation.

Flow Version Properties

A flow version's properties consist of its label, description, interview label, and type. These properties drive the field values that appear on the flow's detail page.

Flow Resources

Each *resource* represents a value that you can reference throughout the flow.

In Flow Builder, the Manager panel displays the resources that are available in the flow.

You can create some resources by clicking **New Resource**. Some resources, such as global constants and global variables, are provided by the system. Other resources are automatically created when you add an element to a flow. For example, when you add a Decision element, a resource for each decision outcome is automatically created.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Resource	Description	Creatable from the Resources Tab
Actions	Output values that are stored automatically from Action elements.	
Choice	Create a choice option to use in a screen component, such as a Radio Buttons or Multi-Select Picklist component.	✓
Collection Choice Set	Generate a set of choices by using an existing collection of records.	✓
Constant	Store a fixed value that you can use throughout a flow.	✓
Decision Outcome	When you add a Decision element to a flow, its outcomes are available as Boolean resources. If an outcome path has already been executed in the flow interview, the resource's value is `True`.	
`Element`	Any element that you add to a flow is available as a resource with the `was visited` operator in decision outcome criteria. An element is considered visited when it's executed in the flow interview.	

Flow Resource	Description	Creatable from the Resources Tab
	Any element that you add to a flow that supports a fault connector is available as a Boolean resource. If the element is already successfully executed in the flow interview, the resource's value is `True`. If the element wasn't executed or was executed and resulted in an error, the resource's value is `False`.	
Formula	Calculate a value when the formula is used in the flow.	✓
Global Constant	Fixed, system-provided values, such as `EmptyString`, `True`, and `False`.	
Global Variable	System-provided variables that reference information about the Salesforce org or running user, such as the user's ID or the API session ID.	
Wait Configuration	When you add a Wait element to a flow, its configurations are available as Boolean resources. If a configuration's wait conditions are met, the resource's value is `True`. If the configuration has no wait conditions set, the resource's value is always `True`.	
Picklist Choice Set	Generate a set of choices by using the values of a picklist or multi-select picklist field.	✓
Picklist Values	System-provided values for picklist fields in record variables and record collection variables. Available only for Assignment elements and conditions.	
Record Choice Set	Generate a set of choices by using a filtered list of records.	✓
Screen Component	Any screen component that you add to a flow is available as a resource. The resource value depends on the type of screen component. The value for a Text component is what the user enters. The value for a Picklist component is the stored value of the choice that the user selects. The value for a Display Text component is the text that's displayed to the user.	
Stage	Represent the user's progress throughout the flow. To identify which stages are relevant to the user throughout the flow, assign the stages to the stage system variables. You can reference stages in flow logic or in the UI, such as with a progress indicator. For example, in a payment flow, the stages are payment details, shipping details, billing details, and order confirmation.	✓
Text Template	Store text that can be changed and used throughout the flow. To format the text, use HTML tags.	✓
Variable	Store a value that can be changed throughout the flow.	✓

SEE ALSO:

Flow Builder Tour

Flow Resource: Choice

Create a choice option to use in a screen component, such as a Radio Buttons or Multi-Select Picklist component.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this choice from other resources.
`Choice Label`	A user-friendly label for the choice.
`Data Type`	Controls which screen components this choice can be used in. For example, you can't use a Text choice in a Currency radio button. You can't change the data type of a previously saved choice.
`Choice Value`	If the user selects this choice, the screen component is set to this value. Exceptions: • If no choice value is configured, the screen component is set to the choice label. • If the choice value references a formula resource, the screen component is set to the choice label.
`Display text input`	Displays a text input component below the choice. This option isn't available if the choice's data type is `Boolean`.

Configure Text Input

These fields appear when you select `Display text input`.

Field	Description
`Input Label`	A user-friendly label for the text input component.
`Require`	Requires the user to enter a value in the text input component before progressing or finishing the flow.
`Validate`	Evaluates whether the user entered an acceptable value.
`Error Message`	If the user didn't enter an acceptable value, this message displays under the text input component. Available only when `Validate` is selected.
`Formula`	Boolean formula expression that evaluates whether the user entered an acceptable value. Available only when `Validate` is selected.

Example: To let users choose a particular service level, create choices for Gold, Silver, and Bronze. In a screen, display the choices with a description of the features included. Then, in the same screen, let the user choose from a Radio Buttons screen component.

Formatting Choices

- Add rich text formatting using the toolbar.
- If you open the Display Text screen component, Choice resource labels, help text, Pause confirmation screens, or input validation, Flow Builder converts existing HTML to rich text. Unsupported HTML is removed. The following HTML tags are converted to rich text: <a>, <b>, <br>, <font>, <i>, <li>, <p>, <span>, <u>, and <div>. HTML that is pasted into the rich text editor isn't supported.

SEE ALSO:

Flow Resources

Standard Flow Screen Components

Using Choice Resources with Flow Screen Components

Flow Resource: Collection Choice Set

Use an existing collection of records or external data to generate a set of choices.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can’t end with an underscore. It also can’t have two consecutive underscores.
`Description`	Helps you differentiate this resource from other resources.
`Record Collection`	The collection you want to use to generate the choices. You can reference an Apex-defined collection from an external service, Apex action, or another screen component.

Configure Each Choice

For each record that meets the filter conditions, the flow creates a choice using values from the record. Identify which fields to use for each choice’s label and value.

Field	Description
`Choice Label`	Determines which field to use as the label for each generated choice. Select a field that enables users to differentiate between the generated choices. Make sure to choose a field that contains data. If the selected field has no value for a given record, the corresponding choice’s label is blank at run time.
`Data Type`	Data type of the choice’s value. You can’t change the data type of a previously saved collection choice set.
`Choice Value`	Determines which field’s value to store when the user selects this choice at run time. The value is determined by the most recent user selection of a choice within the generated set. `Data Type` determines the available options. If you don’t select a field as the choice value, the choice label is used instead.

Tip: In most cases, set the choice label to *Name* and the choice value to *ID*.

Example: Collection choice sets are useful when a flow reuses the same dataset over multiple screens. For example, you're designing a support flow for a company's IT department that handles support requests related to employee hardware. The flow references the same employee hardware data over several screens. To get the employee hardware information, use a Get Records action, which populates a record collection. To define the conditions relevant to the support request, use a collection filter on the record collection. Next, to display the user choices, add a collection choice set that uses the filtered collection. Create a relevant collection filter and collection choice set for each branch of the support flow.

With collection choice sets, the server is queried only when the Get Records element is first executed. In comparison, record choice sets require a server query with each use.

SEE ALSO:

Standard Flow Screen Components

Using Choice Resources with Flow Screen Components

Flow Resources

Flow Resource: Constant

Store a fixed value that can be used but not changed throughout a flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate the constant from other resources.
`Data Type`	Determines the type of value that the constant can store. You can't change the data type of a previously saved constant.
`Value`	The constant's value. This value doesn't change throughout the flow.

SEE ALSO:

Flow Resources

Flow Resource: Formula

Calculate a value when the formula is used in the flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this formula from other resources.
`Data Type`	The data type for the value returned by the formula. You can't change the data type of a previously saved variable.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs. Available only when the data type is Number or Currency.
`Formula`	The formula expression that the flow evaluates at run time. The returned value must be compatible with `Data Type`. Some formula functions aren't supported in Flow Builder.

SEE ALSO:

- Formula Operators and Functions by Context
- Which Functions Aren't Supported in Flow Formulas?
- Flow Resources
- Creating Flow Formulas with Flow Formula Builder

Flow Resource: Global Variables

A system-provided variable holds information that can be referenced throughout the flow. For example, it can contain information about the Salesforce org, flow, running user, or triggering record.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: Use `{!$User.Id}` to access the ID of the user who's running the flow interview.

Global Variable API Name	Label	Description
`$Api`	API	The Session ID's SOAP API endpoints. These merge fields are available. • `Enterprise_Server_URL_`***`xxx`***—The Enterprise WSDL SOAP endpoint, where *`xxx`* represents the API version. • `Partner_Server_URL_`***`xxx`***—The Partner WSDL SOAP endpoint, where *`xxx`* represents the API version. • `Session_ID`
`$Client`	Running User's Client	The form factor of the running user's device. Available only in Lightning Scheduler flows and supported only in Decision elements. `$Client.FormFactor` is automatically set to `Large` (computer), `Medium` (tablet), or `Small` (phone), depending on the device that's running the flow. As you build a flow, use a Decision element to create separate paths for large, medium, and small devices. In each path, use screens that are optimized for that path's device form factor.
`$Event`	$Event	The event that triggers an automation event-triggered flow to run, for example, a form submission. With this variable you can access information about the event such as the event name.
`$Flow`	Running Flow Interview	The flow instance while it's running. For details, see Flow Resource: $Flow Global Variables.
`$Input`	$Input	Input data that's provided from outside of a flow. The flow can reference inputs to perform logic and actions. For example, a prompt template in Prompt Builder sets the Recipient input to Mary Smith. The prompt template provides the Recipient input to the flow. The Recipient input is set to the Contact data type in the flow. To retrieve Mary Smith's contact record by first name, the Get Records element filters for contacts with the First Name field equal to the value for `$Input > Recipient > First Name`, which is Mary. Available only in the Prompt Flow process type.
`$Label`	Custom Label	A label in a custom label file in your Salesforce org. This global variable appears only if your org has custom labels. The returned value depends on the language setting of the contextual user. The value returned is one of the following, in order of precedence: • The local translation's text • The packaged translation's text • The primary label's text
`$Organization`	Running Org	The Salesforce org where the flow is running. With this variable, you can access information like the organization's name or address.
`$Permission`	Running User's Permission	The custom permission access of the user running the flow.
`$Output`	$Output	Output data from a flow that's available inside or outside the flow. The flow can reference outputs to perform logic and actions. For example, a flow that's associated with a prompt template in Prompt Builder can reference `$Output > Prompt`

Global Variable API Name	Label	Description
		to assign text to a field on a record. Outside the flow, an integrated app can merge the value that's referenced by `$Output`. For example, a prompt template merges the value of `$Output` into the prompt template's resolution. Available only in the Prompt Flow process type.
`$Profile`	Running User Profile	The profile of the user running the flow. With this variable, you can access information like the license type or name. • Use profile names to reference standard profiles in `$Profile` merge fields. • Users don't need access to their profile information to run a flow that references these merge fields.
`$Record`	Triggering *`Object`*, for example, Triggering Account	The record that triggered the flow. Available only in autolaunched flows with triggers. In a record-triggered flow, the `$Record` global variable contains the triggering record's values. You can reference and change `$Record` values throughout the flow. • If the flow runs before the record is saved to the database, Salesforce automatically applies any changed `$Record` values to the record in the database. • If the flow runs after the record is saved to the database, use an Update Records element to apply changed `$Record` values to the record in the database. A schedule-triggered flow starts at the specified time and frequency for a batch of records. A flow interview runs for each record in the batch and stores all of the record's field values in the `$Record` global variable. You can reference and change these values throughout the flow. If you configure an Update Records element to use the ID and all field values from the $Record global variable, enable `Filter inaccessible fields from flow requests` in your org's process automation settings. Otherwise, the flow fails because the Update Records element tries to set the values for system fields and other read-only fields.
`$Record__Prior`	Prior Values of Triggering *`Object`*, for example, Prior Values of Triggering Account	The triggering record before a change triggered the flow. Available only in record-triggered flows that are configured to run when a record is updated or when a record is created or updated. The `$Record__Prior` global variable contains the values that the triggering record had immediately before the flow started. You can't change these values in the flow. When the flow is triggered by a newly created record, all `$Record__Prior` values are null.

Global Variable API Name	Label	Description
`$Setup`	Custom Hierarchy Settings	Your custom settings of type `hierarchy`. This global variable appears only if your org has custom hierarchy settings. You can access custom settings of type `list` only in Apex. Hierarchical custom settings allow values at these levels. Salesforce determines the correct value for this custom setting field based on the running user's current context. • Organization—The default value for everyone • Profile—Overrides the Organization value • User—Overrides the Organization and Profile values
`$System`	System	The system's OriginDateTime (`$System.OriginDateTime`), which represents the literal value of `1900-01-01 00:00:00`. Use this merge field to perform date/time offset calculations.
`$User`	Running User	The user who's running the flow interview. For example, reference the user's ID or title. • The running user is the person who caused the flow to start. • When a flow is started because a Web-to-Case or Web-to-Lead process changed a record, the running user is the `Default Lead Owner` or `Default Case Owner`. `$User.UITheme` and `$User.UIThemeDisplayed` identify the look and feel that the running user sees on a Salesforce page. The difference between the two variables is that `$User.UITheme` returns the look and feel the user is supposed to see, while `$User.UIThemeDisplayed` returns the look and feel the user actually sees. For example, a user can have the preference and permissions to see the Lightning Experience look and feel, but if they're using a browser that doesn't support that look and feel, for example, older versions of Internet Explorer, `$User.UIThemeDisplayed` returns a different value. These merge fields return one of these values. • `Theme1`—Obsolete Salesforce theme • `Theme2`—Salesforce Classic 2005 user interface theme • `Theme3`—Salesforce Classic 2010 user interface theme • `Theme4d`—Modern "Lightning Experience" Salesforce theme • `Theme4t`—Salesforce mobile app theme • `Theme4u`—Lightning Console theme • `PortalDefault`—Salesforce Customer Portal theme • `Webstore`—AppExchange theme
`$UserRole`	Running User Role	The role of the user running the flow. For example, reference the role name or ID. • The running user is the person who caused the flow to start.

Global Variable API Name	Label	Description
		- When a flow is started because a Web-to-Case or Web-to-Lead process changed a record, the running user is the `Default Lead Owner` or `Default Case Owner`.

Global Variable Considerations for Flows

- `$Flow` is the only global variable available in screen component visibility conditions.
- In a record-triggered flow, the `$Record` global variable doesn't contain the triggering record's values for fields whose values are derived from other records. Examples of derived fields include `Contact.Name` and `User.MediumPhotoUrl`.
- Multi-select picklist, time, and location global variables are available only in formulas.
- If a field in the database has no value, the corresponding merge field returns a blank value. For example, if no value is set for your org's Country field, `{!$Organization.Country}` returns no value.
- `$Label` global variables take longer to load in the flow resource selection list. When selecting a `$Label` global variable, if the `$Label` option isn't visible in the flow resource selection list, close the window and try again in a few minutes.

SEE ALSO:

Flow Operations and Read-Only Fields

Salesforce Data Considerations for Flows

Flow Resource: `$Flow` Global Variables

A `$Flow` global variable provides information about the running interview. Some variables contain system-provided values. You can update the other variables throughout the flow by using Assignments or by storing output values in the variables.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Global Variable	Supported Resource Types	Description	Value Set By
`$Flow.ActiveStages`	Stage	A collection of stages that are relevant to the current path of the flow. For example, each item in a progress indicator corresponds to a stage in `$Flow.ActiveStages`.	Assignment
`$Flow.CurrentDate`	Text, Date, and Date/Time	Date when the flow interview executes the element that references the global variable.	System
`$Flow.CurrentRecord`	Text	ID of a related record. The value must be a single ID for a valid	Assignment

Global Variable	Supported Resource Types	Description	Value Set By
		object. All custom objects and most standard objects are valid. When a user pauses the flow interview or the interview executes a Wait element, the interview is associated with this record by creating a FlowRecordRelation record. If the ID isn't valid, the interview fails to pause.	
`$Flow.CurrentStage`	Stage	The currently selected stage. For example, the selected item in a progress indicator corresponds to `$Flow.CurrentStage`.	Assignment
`$Flow.CurrentDateTime`	Text, Date, and Date/Time	Date and time when the flow interview executes the element that references the global variable.	System
`$Flow.FaultMessage`	Text	System fault message that can help flow administrators troubleshoot runtime issues.	System
`$Flow.InterviewGuid`	Text	Unique identifier for the interview.	System
`$Flow.InterviewStartTime`	Text, Date, and Date/Time	Date and time when the flow interview started. For a flow launched by a Subflow element, `$Flow.InterviewStartTime` indicates when the initial master flow started.	System

Example: A flow is used internally by call center personnel. For each flow element that interacts with the Salesforce database, a fault connector leads to a screen. A Display Text screen component displays the system fault message and instructs the flow user to provide that message to the IT department.

```
Sorry, but you can't
        read or update records at this time.
Please open a case with IT, and include the following error message:
{!$Flow.FaultMessage}
```

Example: If a customer asks to be forgotten, make sure to delete all references to information that could personally identify the customer, including data in paused flow interviews. When an interview executes a Wait element or is paused by a user, all the interview data is serialized and saved to the database as a Paused Flow Interview record. When the interview is resumed, the Paused Flow Interview record is deleted.

To identify which paused interviews include personal data for a contact, lead, or user, build a custom object to track the interview's GUID and the affected contact, lead, or user. When an interview references personal data, such as a lead's email or credit card number, create a record of the custom object using the lead's ID and `{!$Flow.InterviewGuid}`. Before the final screen, delete all records of the custom object referencing the interview's GUID. That way, the custom object tracks only interviews that are saved to the database.

When a customer asks to be forgotten, create a report that lists all the custom object records where LeadId matches the customer's record. Then for each custom object record, delete the flow interview that corresponds to the provided GUID.

SEE ALSO:

Customize What Happens When a Flow Fails

Flow Resources

Flow Resource: Global Constant

Fixed, system-provided values.

Global Constant	Supported Data Type
`{!$GlobalConstant.True}`	`Boolean`
`{!$GlobalConstant.False}`	`Boolean`
`{!$GlobalConstant.EmptyString}`	`Text`

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: When you create a Boolean variable, the supported values are `$GlobalConstant.True` and `$GlobalConstant.False`.

Null Versus Empty String

At run time, `{!$GlobalConstant.EmptyString}` and `null` are treated as separate, distinct values.

- `{!$GlobalConstant.EmptyString}` indicates a text value with zero characters. It's used to determine whether a field or variable is blank.
- `null` indicates that a value doesn't exist. It's used to determine whether a field or variable value is available.
- To check if a field or variable has been populated with data in a condition, use **Equals** for the operator, and **{!$GlobalConstant.EmptyString}** for the value.
- To check if a field or variable value isn't available, in a condition, use **Is Null** for the operator and **{!$GlobalConstant.True}** for the value.

Example: To check if a Get Records element found records, in a Decision element outcome condition, use the Get Records record collection for the resource, **Is Null** for the operator, and **{!$GlobalConstant.False}** for the value.

Considerations

- If you don't give a text field or variable a starting value, the value is `null` at run time. If you want the value to be treated as an empty string, set it to `{!$GlobalConstant.EmptyString}`.
- For a text field or component placed on a Screen element, the Is Null operator always evaluates to false. To determine if the field or component has no value, use **Equals** for the operator and **{!$GlobalConstant.EmptyString}** for the value.
- If a condition compares two text variables, make sure that their default values are either set to `{!$GlobalConstant.EmptyString}` or left empty (`null`).

- To check for both a `null` or `{!$GlobalConstant.EmptyString}` value at the same time, use the ISBLANK formula function.

SEE ALSO:

Flow Resources

Flow Resource: Picklist Choice Set

Generate a set of choices by using the values of a picklist or multi-select picklist field.

EDITIONS

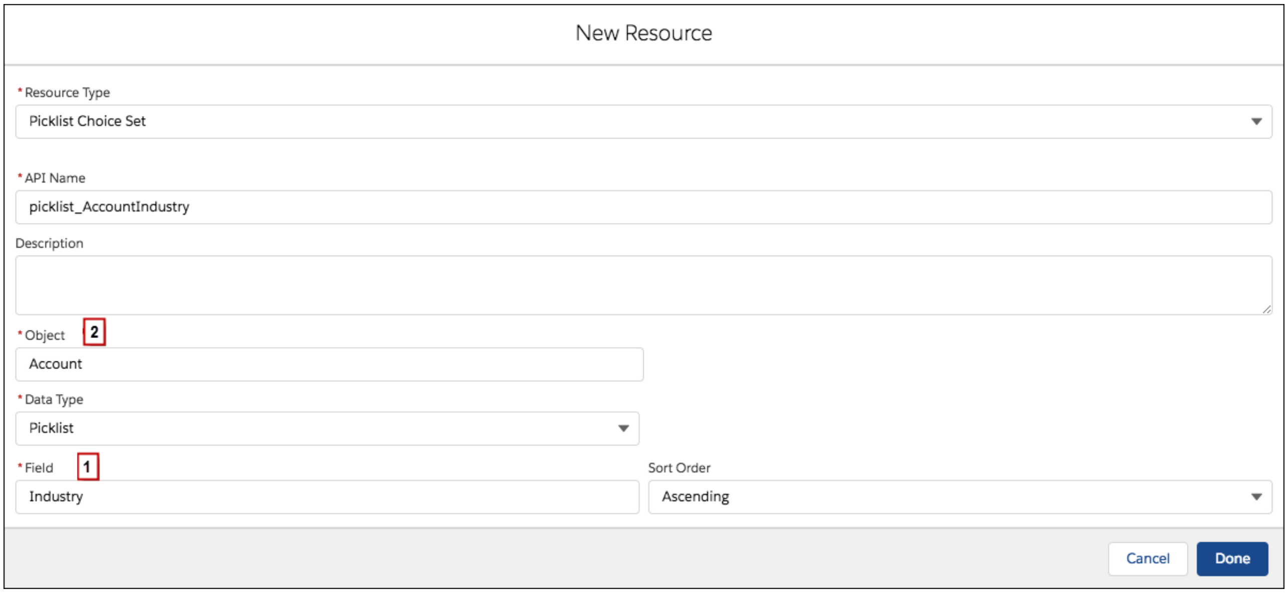
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this resource from other resources.
`Object`	The object whose fields you want to select from. You can't change the object for a previously saved picklist choice set.
`Data Type`	Determines whether you can choose from picklist fields or multi-select picklist fields. You can't change the data type of a previously saved picklist choice set.
`Field`	The picklist or multi-select picklist field to use to generate the list of choices.
`Sort Order`	Controls the order that the choices appear in. The choices sort based on the translated picklist value for the running user's language.

Example: In a flow that simplifies the process of creating an account, users identify the company's industry.

Rather than creating one choice for each industry, you add a picklist choice set to the flow and populate a Picklist screen component with it. When a user runs the flow, the picklist choice set finds all the values in the database for the Industry field (1) on the Account object (2).



New Resource

* Resource Type: Picklist Choice Set

* API Name: picklist_AccountIndustry

Description:

* Object: Account

* Data Type: Picklist

* Field: Industry

Sort Order: Ascending

In addition to being easier than the standalone choice resource to configure, a picklist choice set reduces maintenance. When someone adds options to the Account Industry field, the flow automatically reflects the changes. You don't have to manually update the flow.

Considerations

You can't do the following when using a picklist choice set.

- Filter out values that come back from the database. The flow always displays every picklist value for the field, even if you're using record types to narrow down the picklist choices in page layouts.
- Customize the label for each option. The flow always displays the label for each picklist value.
- Customize the stored value for each option. The flow always stores the API value for each picklist value.

Picklists for Knowledge Articles aren't supported.

Labels and Values for Translated Fields

When a picklist field has been translated:

- Each choice's label uses the version of the picklist value in the running user's language.
- Each choice's stored value uses the version of the picklist value in the org's default language.

SEE ALSO:

Standard Flow Screen Components

Using Choice Resources with Flow Screen Components

Flow Resources

Place Record Fields Directly on Flow Screens

Flow Resource: Record Choice Set

Generate a set of choices by using a filtered list of records.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this resource from other resources.
`Object`	The object whose records you want to use to generate the choices. You can't change the object for a previously saved record choice set.

Filter *`Object`* Records

Determines which records are included in the choice set. For example, to generate a list of all accounts in San Francisco, use filters to include only accounts whose Billing City is San Francisco.

Tip: Without filter conditions, a choice is generated for every record of the selected object. If you don't apply filter conditions, make sure to sort the records in ascending or descending order.

Sort *`Object`* Records

Determines how to sort the filtered list of records and how many records to include in the choice set.

Field	Description
`Sort Order`	Controls the order that the choices appear in.
`Sort By`	When the sort order is ascending or descending, select the field to order the choices by.
`Maximum Number of Choices`	The maximum number of choices to display for the screen component that uses this record choice set. By default, the maximum is 200.

Configure Each Choice

For each record that meets the filter conditions, the flow creates a choice using values from the record. Identify which fields to use for each choice's label and value.

Field	Description
`Choice Label`	Determines which field to use as the label for each generated choice. Select a field that enables users to differentiate between the generated choices. Make sure to choose a field that contains data. If the selected field has no value for a given record, the corresponding choice's label is blank at run time.

Field	Description
`Data Type`	Data type of the choice's value. You can't change the data type of a previously saved record choice set.
`Choice Value`	Determines which field's value to store when the user selects this choice at run time. The value is determined by the most recent user selection of a choice within the generated set. `Data Type` determines the available options. If you don't select a field as the choice value, the choice label is used instead.

Store More *`Object`* Field Values

When a choice is selected, store field values from the associated record in flow variables that you can reference later.

Note: When a Checkbox Group, Multi-Select Picklist, or Choice Lookup screen component uses a record choice set, only values from the last record that the user selects are stored in the flow variables. If multiple Checkbox Group or Multi-Select Picklist components on one screen use the same record choice set, the variable assignments come from the last record selected from all of those components.

Example: In a support flow for a computer hardware manufacturer, users identify a product to find its latest updates. You create a record choice set that displays all products whose product ID starts with a specific string of characters. However, the flow users are more likely to know the product's name than its ID. So for `Choice Label`, select the field that contains the product name, and for `Choice Value`, select the ID field. Elsewhere in the flow, you want to display the associated description. To do so, you store the Description field value from the user-selected record in a variable.

SEE ALSO:

- Flow Operators in Data Elements and Record Choice Sets
- Standard Flow Screen Components
- Using Choice Resources with Flow Screen Components
- Flow Resources

Flow Resource: Stage

Represent the user's progress throughout the flow. To identify which stages are relevant to the user throughout the flow, assign the stages to the stage system variables. You can reference stages in flow logic or in the UI, such as with a progress indicator. For example, in a payment flow, the stages are payment details, shipping details, billing details, and order confirmation.

Field	Description
`Label`	A user-friendly label for the stage. Merge fields aren't supported.
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Description`	Helps you differentiate this stage from other resources.
`Order`	Required. Determines how to sort this stage among the other stages in the flow. The order must be unique among all other stages in the flow.
`Active by default`	Adds this stage to `{!$Flow.ActiveStages}` when an interview starts.

Usage

When ordering your stages, leave gaps between the numbers in case you later want to add a stage between two other stages. For example, if you use 10, 20, and 30 as the order, you can insert a stage at order 15 without updating the original three stages.

Most of the time, stages resolve to the fully qualified name: `namespace.flowName:stageName` or `flowName:stageName`. Stages resolve to the label in:

- Display contexts, such as choice labels and Display Text screen components
- Attributes in screen components that require Lightning runtime

SEE ALSO:

Plan the Stages in Your Flow

Identify the Relevant Stages in Your Flow

Flow Stage Considerations

Flow Resources

Sample Flows That Display Stages

These Online Purchase flows display stages as sections on a progress indicator. Each sample flow displays stages differently based on how the flow is configured.

Sample Flow That Displays Stages as Breadcrumbs

This Online Purchase flow shows visitors what parts of the flow they've completed by displaying all stages up to the current stage. This flow displays only the stages that the user has visited.

Sample Flow That Displays All the Active Stages

This Online Purchase flow shows visitors all active stages and the current stage so that they know what to expect throughout this flow.

Sample Flow That Displays Stages as Breadcrumbs

This Online Purchase flow shows visitors what parts of the flow they've completed by displaying all stages up to the current stage. This flow displays only the stages that the user has visited.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

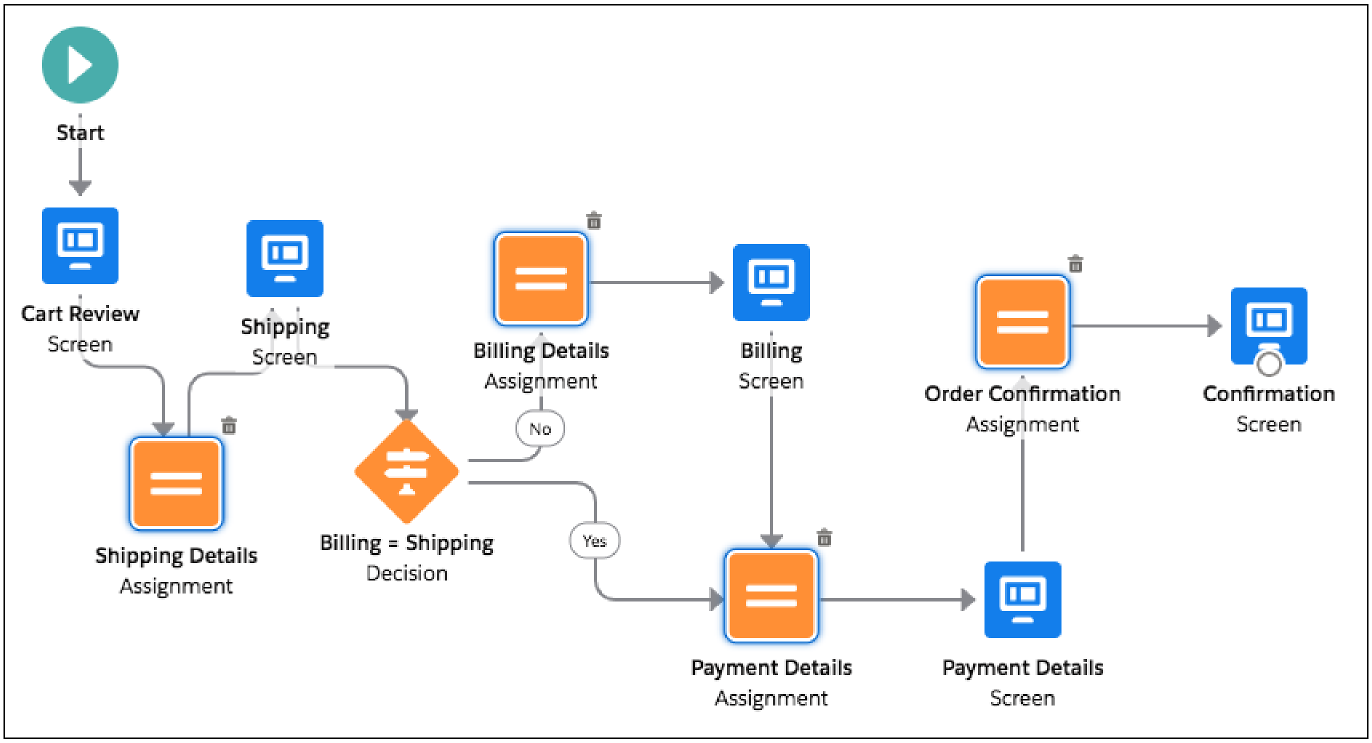
Example

This flow includes stages for users to review their cart, enter shipping details, enter billing details, enter payment details, and confirm their order. Since we're displaying the stages as breadcrumbs, only the first stage is active by default.

Stage Label	Unique Name	Order	Active by Default
Review Cart	Review_Cart	0	Yes
Shipping Details	Shipping_Details	1	No
Billing Details	Billing_Details	2	No
Payment Details	Payment_Details	3	No
Order Confirmation	Order_Confirmation	4	No

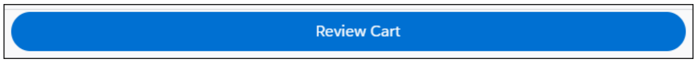
When the flow starts, Review Cart is automatically set to `$Flow.CurrentStage` and is the only stage in `$Flow.ActiveStages`.

Each time the flow moves to a different stage, an Assignment element resets the current stage and adds the new stage to the active stages.

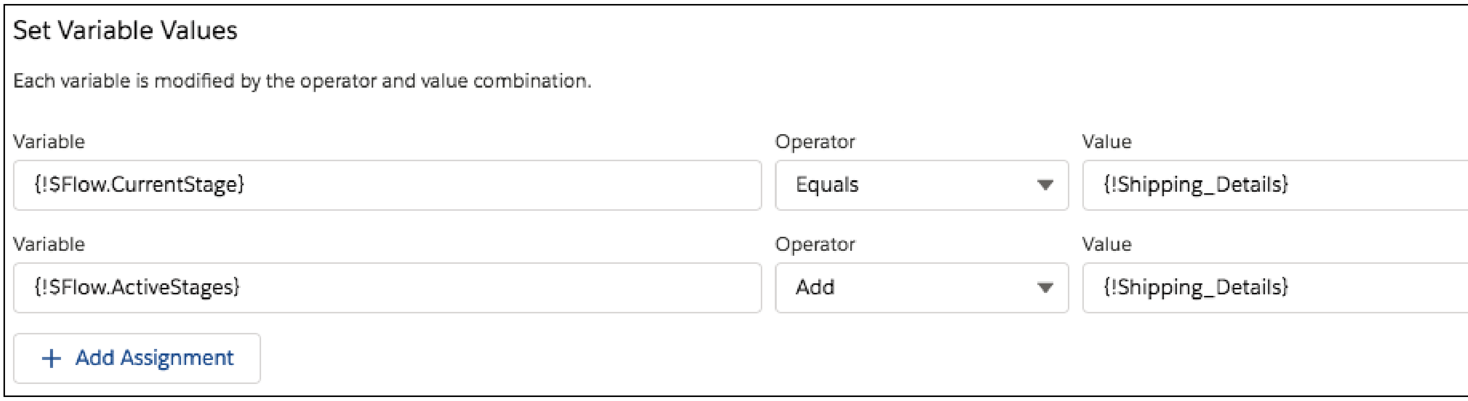

Start
Cart Review Screen
Shipping Screen
Shipping Details Assignment
Billing = Shipping Decision
No
Yes
Billing Details Assignment
Billing Screen
Payment Details Assignment
Payment Details Screen
Order Confirmation Assignment
Confirmation Screen

Note: This sample uses an Aura component to display the flow's stages. For details, see Represent Your Flow's Stages Visually.

The first screen displays only one active stage, which is also the user's current stage: Review Cart.



Next, the flow moves to a new stage: Shipping Details. To make sure that the active stages and current stage respect the change, the flow updates the global variables with an assignment.



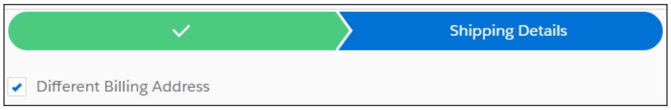
Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Shipping_Details}
{!$Flow.ActiveStages}	Add	{!Shipping_Details}

`$Flow.ActiveStages` now contains the Review Cart and Shipping Details stages, and `$Flow.CurrentStage` is set to the Shipping Details stage.

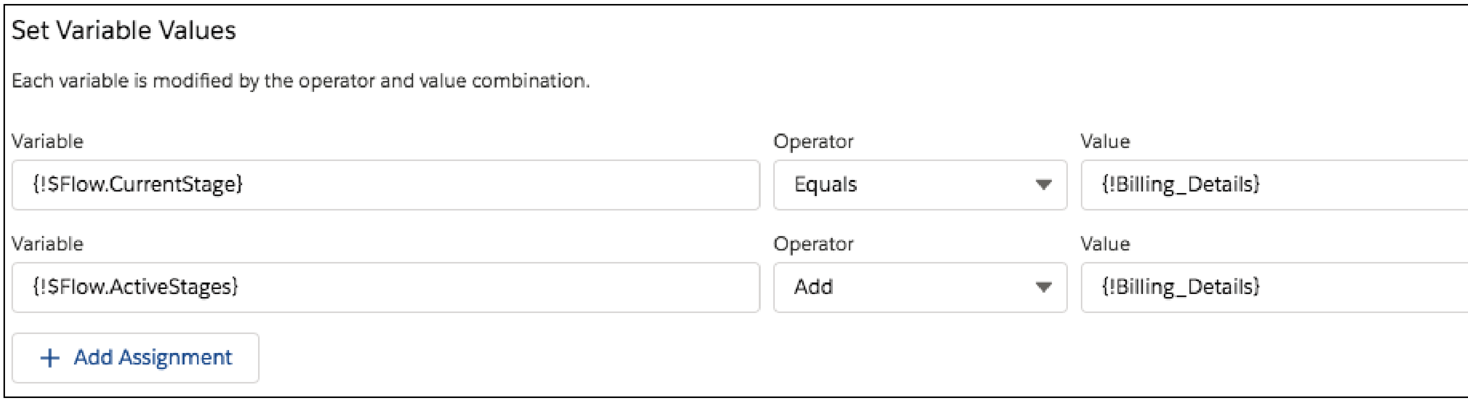
Often, a user's shipping details and billing details are the same. On the Shipping Details screen, the user can indicate that the billing address is different.



☑ Different Billing Address

The flow uses the value of the Different Billing Address checkbox to determine where to go next. If the shipping and billing details are the same, the flow continues to the Payment Details assignment. If the billing and shipping details are different, the flow moves to the Billing Details assignment.

To make sure that the active stages and current stage respect the change, the flow updates the global variables with an assignment.

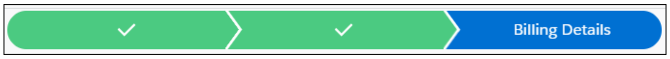


Set Variable Values

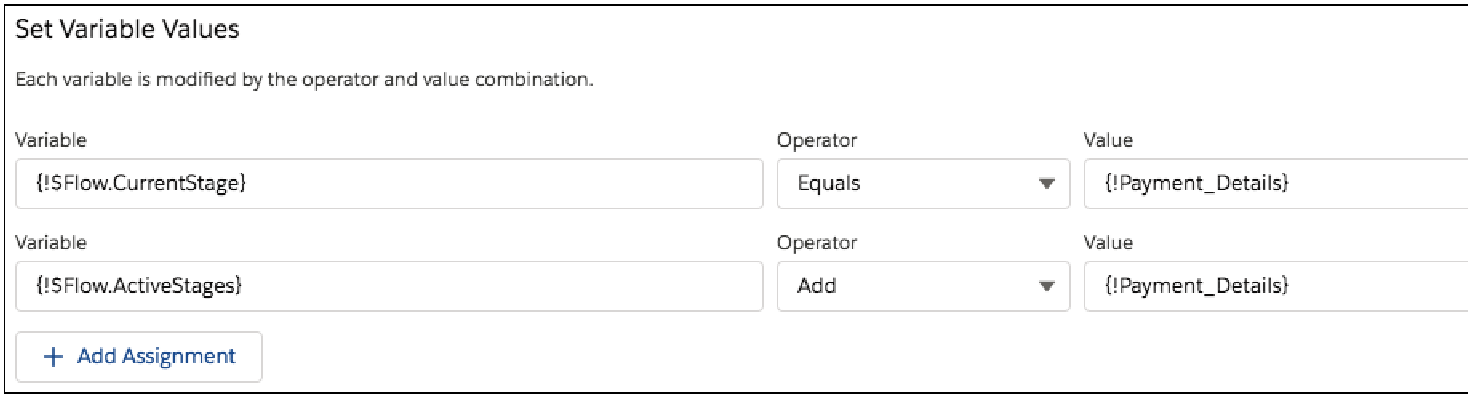
Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Billing_Details}
{!$Flow.ActiveStages}	Add	{!Billing_Details}

Now `$Flow.ActiveStages` contains the Review Cart, Shipping Details, and Billing Details stages, and `$Flow.CurrentStage` is set to the Billing Details stage.



After the shipping and billing details are complete, the flow moves to the Payment Details stage. To make sure that the active stages and current stage respect that change, the flow updates the global variables with an assignment.

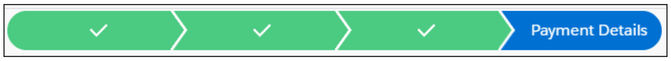


Set Variable Values

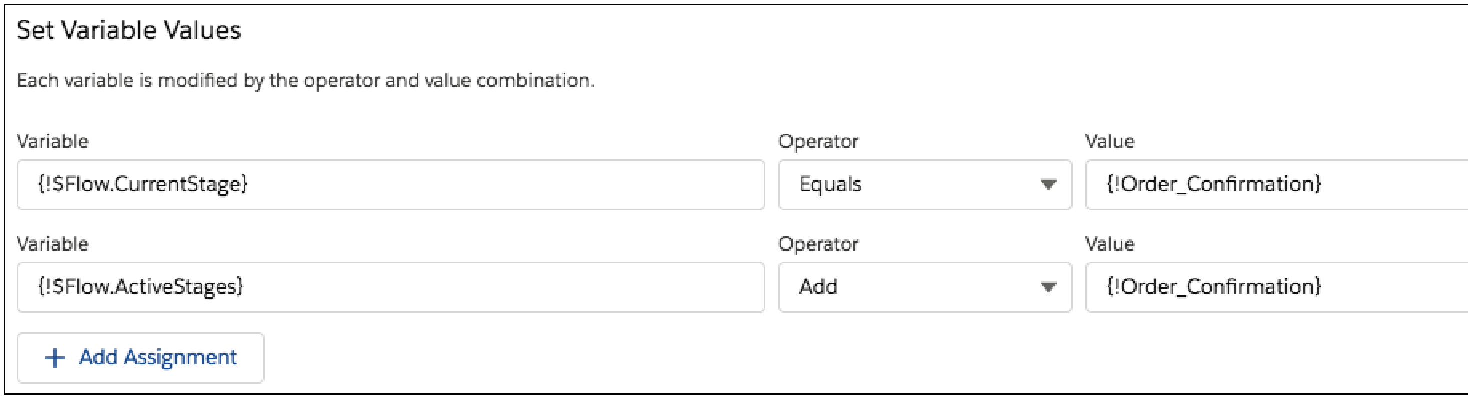
Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Payment_Details}
{!$Flow.ActiveStages}	Add	{!Payment_Details}

`$Flow.ActiveStages` contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different), and Payment Details stages. The `$Flow.CurrentStage` global variable is set to the Payment Details stage.



Finally, the flow moves to the last stage: Order Confirmation. To make sure that the active stages and current stage respect the change, the flow updates the global variables with an assignment.

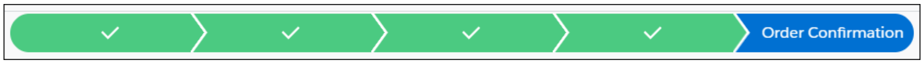


Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Order_Confirmation}
{!$Flow.ActiveStages}	Add	{!Order_Confirmation}

`$Flow.ActiveStages` now contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different), Payment Details, and Order Confirmation stages. The `$Flow.CurrentStage` global variable is set to the Order Confirmation stage.



Sample Flow That Displays All the Active Stages

This Online Purchase flow shows visitors all active stages and the current stage so that they know what to expect throughout this flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example

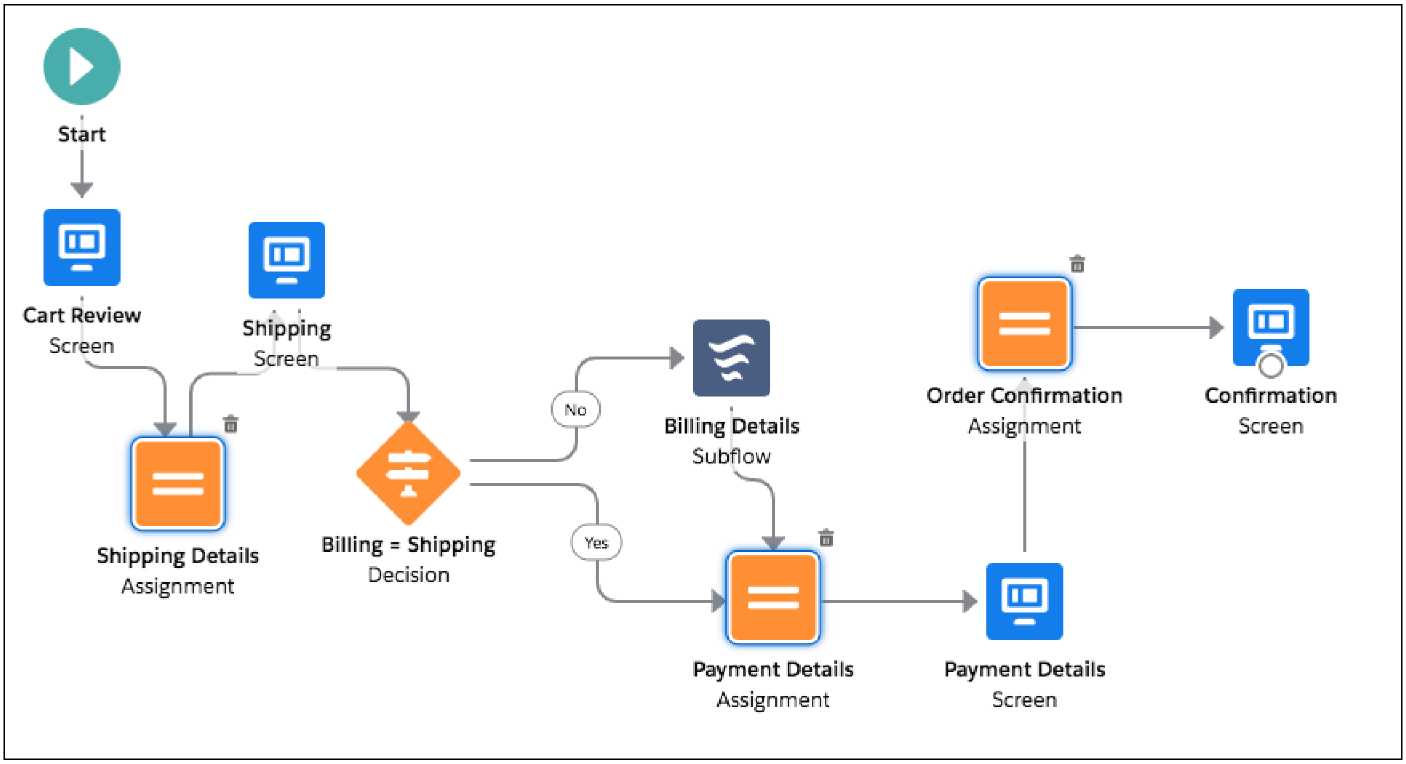
This flow includes stages for users to review their cart, enter shipping details, enter billing details, enter payment details, and confirm their order. To give users an idea of the steps they go through in the flow, we're displaying all the applicable stages when the flow starts. Every user goes through the Review Cart, Shipping Details, Payment Details, and Order Confirmation stages, so those stages are all active by default.

Not all users enter billing details, because a user's shipping and billing details can be the same. To insert an optional stage in the flow's active stages, create another flow and reference it by using a Subflow element

Stage Label	Unique Name	Order	Active by Default
Review Cart	Review_Cart	0	Yes
Shipping Details	Shipping_Details	1	Yes
Payment Details	Payment_Details	2	Yes
Order Confirmation	Order_Confirmation	3	Yes

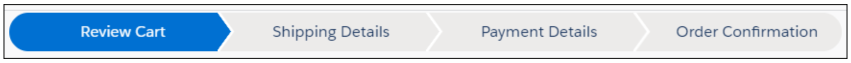
When the flow starts, Review Cart is automatically set to `$Flow.CurrentStage`, and `$Flow.ActiveStages` contains Review Cart, Shipping Details, Payment Details, and Order Confirmation.

Each time the flow moves to a different stage, an Assignment element resets the current stage.

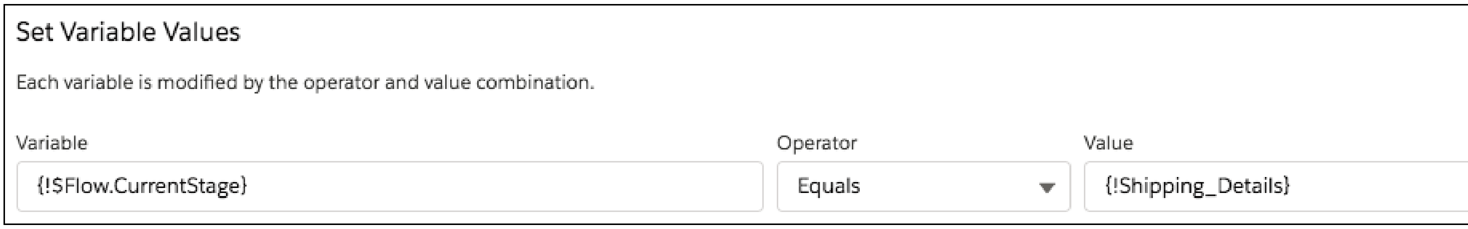


Note: This sample uses an Aura component to display the flow's stages. For details, see Represent Your Flow's Stages Visually.

The first screen displays all active stages and the user's current stage: Review Cart.



Next, the flow moves to a new stage: Shipping Details. To make sure that the current stage respects the change, the flow updates the global variable with an assignment. `$Flow.CurrentStage` is set to the Shipping Details stage.

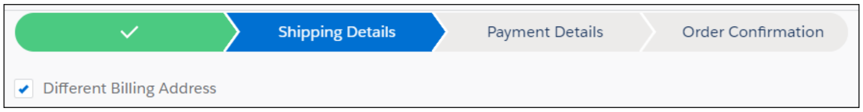


Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Shipping_Details}

Often, a user's shipping details and billing details are the same. On the Shipping Details screen, the user can indicate that the billing address is different.



☑ Different Billing Address

The flow uses the value of the Different Billing Address checkbox to determine where to go next. If the shipping and billing details are the same, the flow continues to the Payment Details assignment. If the billing and shipping details are different, the flow uses a Subflow element to reference the Billing Details flow.

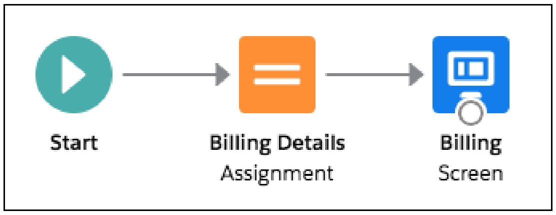
The Billing Details flow includes an optional stage for users to enter billing details between shipping and payment details.

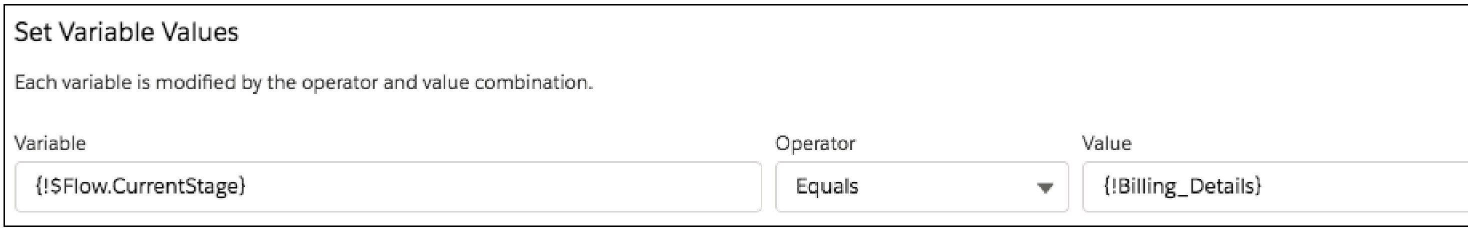
Stage Label	Unique Name	Order	Active by Default
Billing Details	Billing_Details	1	Yes

When a referenced flow starts, its default active stages are automatically inserted in `$Flow.ActiveStages` after the current stage.

When the Billing Details flow starts, `$Flow.CurrentStage` is Shipping Details. The Billing Details stage is inserted into `$Flow.ActiveStages` immediately after the current stage. Now `$Flow.ActiveStages` contains the Review Cart, Shipping Details, Billing Details, Payment Details, and Order Confirmation stages.

The flow uses an assignment to set the current stage to Billing Details.



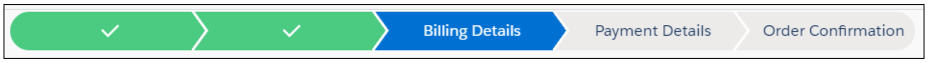


Set Variable Values

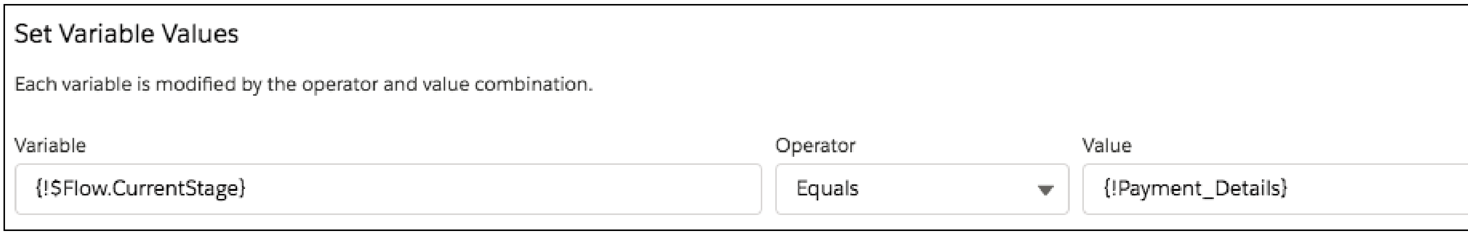
Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Billing_Details}

The `$Flow.CurrentStage` global variable is set to the Billing Details stage.



After the shipping and billing details are complete, the flow moves to the Payment Details stage. To make sure that the current stage respects that change, the flow updates the system variable with an assignment.

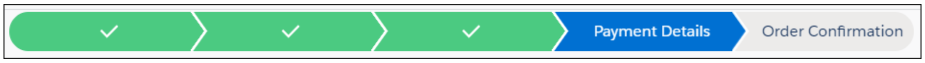


Set Variable Values

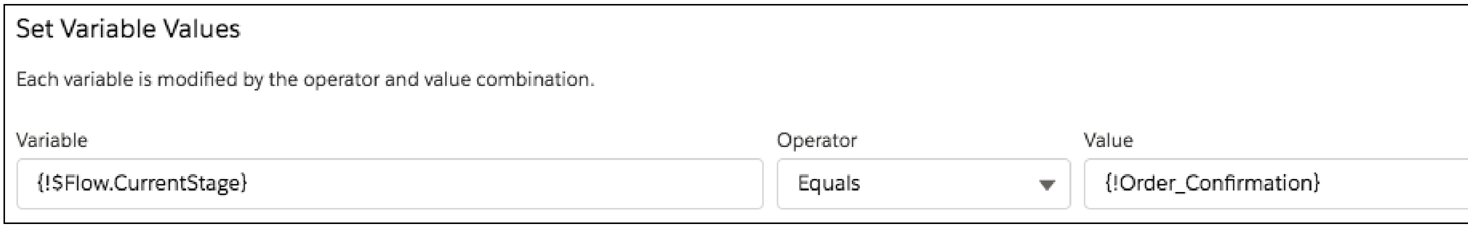
Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals	{!Payment_Details}

The `$Flow.CurrentStage` global variable is set to the Payment Details stage.



Finally, the flow moves to the last stage: Order Confirmation. To make sure that the current stage respects the change, the flow updates the global variable with an assignment.

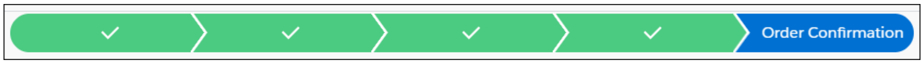


Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
{!$Flow.CurrentStage}	Equals ▼	{!Order_Confirmation}

`$Flow.ActiveStages` now contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different), Payment Details, and Order Confirmation stages. The `$Flow.CurrentStage` global variable is set to the Order Confirmation stage.



Flow Resource: Text Template

Store text that can be changed and used throughout the flow.

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this text template from other resources.
Text Template	The text for the template. To reference information from other resources, use merge fields.
Rich Text	Control the font, size, color, and alignment of text. Add merge fields, HTML links, bullet points, or numbered lists. Rich text is on by default. Click ▼ to change to Rich Text.
Plain Text	Send email core actions use plain text. Some custom actions from AppExchange or built by Salesforce developers also expect plain text. Click ▼ to change to Plain Text.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: You're designing a flow that registers people for an event. You create a text template that includes a registrant's name, address, and other information. Then you use the template in an email confirmation that the flow sends when it finishes.

SEE ALSO:

Flow Resources

Flow Resource: Variable

Store a value that can be used or changed throughout the flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Apex Class`	The Apex class that defines fields for the Apex-defined data type. Only fields with the @AuraEnabled annotation are available in a flow.
`API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this variable from other resources.
`Data Type`	Determines the types of values that the variable can store. You can't change the data type of a previously saved variable. The Record data type can store multiple field values for one record. The Apex-defined data type can store multiple field values for one Apex class. Looking for sObject? In Flow Builder, that data type changed to Record.
`Allow multiple values (collection)`	When selected, the resource is a collection variable. You can store a list of values in collection variables. Collection variables can store only values that are compatible with its data type. When the data type is Record, the collection variable can only store values for the associated object's records. For example, store multiple email addresses in a collection variable, and reference the collection variable to send an email.
`Object`	The object whose field values you can store in the variable. You can't change the object of a previously saved variable. Available only when the data type is Record.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs. Available only when the data type is Number or Currency.
`Availability Outside the Flow`	When a variable is available for input, it can be set at the start of the flow, such as when a flow is started from a Lightning page, a process, or another

Field	Description
	flow. When a variable is available for output, it can be accessed throughout the flow, such as by a Lightning web component or another flow. The default value of the field depends on the release or API version in which the variable is created. • If the variable was created in Summer '12 or later, or in API version 25.0 or later, by default the variable isn't available for input or output. • If the variable was created in Spring '12 or earlier, or in API version 24.0 or earlier, by default the variable is available for both input and output. Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, processes, and other flows. This field doesn't affect how variables are assigned or used within the same flow, for example, through these types of elements: Assignment, Create Records, Get Records, and Apex Action.
`Default Value`	Determines the variable value when the flow starts. If you leave this field blank, the value is `null`. Not available for Picklist and Multi-Select Picklist variables.

SEE ALSO:

Sample Flow That Loops Through a Collection

Flow Element: Loop

Flow Operators in Assignment Elements

Flow Resources

Flow Variable Considerations

Flow Element: Transform

Add Values to a Collection Variable

After you create a collection variable, populate it with values to reference throughout your flow. You can't use a Get Records element to populate a collection variable, but there are some workarounds.

To use values from outside the flow, make sure that the collection variable is available for input. When the values come from outside the flow, the values can be set only when the flow interview starts.

To add values that are stored in a...	Do this action...	For more information
Screen component	Add the field's entered or stored value to a collection variable by using an Assignment element.	• Screen components • Assignments
Variable	Add the variable's stored value to a collection variable by using an Assignment element.	• Variables • Assignments

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To add values that are stored in a...	Do this action...	For more information
Record variable	Add one of the record variable's stored field values to a collection variable by using an Assignment element.	• Variables • Assignments
Record collection variable	Loop through the record collection variable. Within the loop, add one of the loop variable's stored field values to a collection variable by using an Assignment element.	• Variables • Loops • Assignments

Sample Flow That Populates a Collection Variable

Populate a collection variable by populating a record collection variable. Then individually assign the record collection variable's values to the collection variable.

In this scenario, you're designing a flow to send an email to every employee who lives in San Francisco.

The Send Email core action lets you easily send emails from a flow. However, the Recipients parameter only accepts text variables and text collection variables. Since multiple users live in San Francisco, use a collection variable (rather than entering the email address for each individual user).

You can't use a Get Records element to populate collection variables. First populate a User-based record collection variable with field values, including `Email`, from the employees who live in San Francisco. Then add those emails to the collection variable.

After the collection variable is populated, use the collection variable as the value for the Send Email element's `Email Addresses (collection)` parameter.

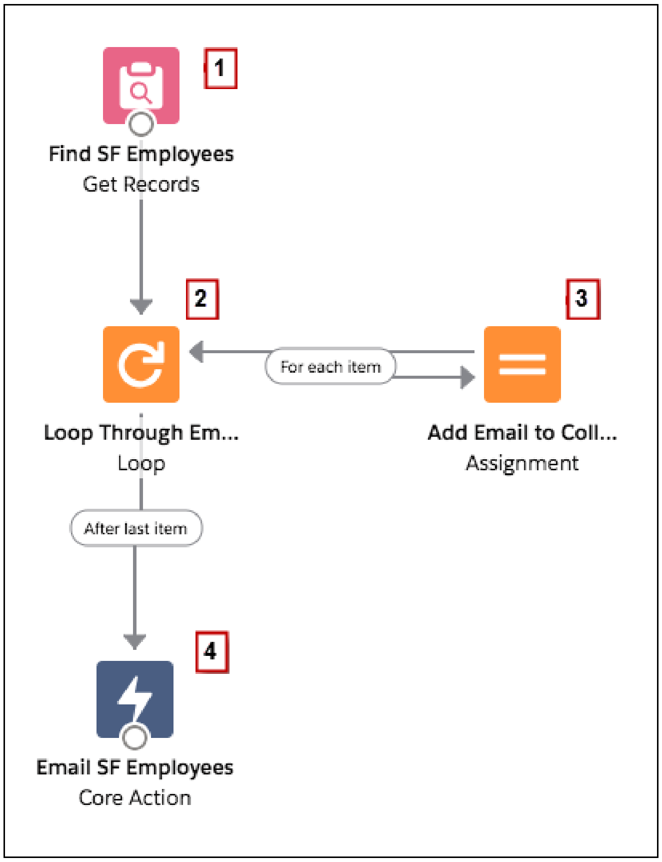
This flow already contains these resources.

- A User-based record collection variable called `employeesInSF`
- A User-based record variable called `loopVariable`
- A Text-based collection variable called `emails_employeesInSF`

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



The example flow:

1. Finds all user records whose `City` is "San Francisco" and populates `employeesInSF` with those records' `Email`.
2. Loops through the record collection variable so that it can look at each individual user record. The loop copies the values of each item in `employeesInSF` to `loopVariable`.
3. For each iteration, assigns the user's `Email` to a collection variable that has a Data Type of Text.
4. When the loop ends, the flow sends an email to the users whose emails are now stored in `emails_employeesInSF`.

SEE ALSO:

Add Values to a Collection Variable

Flow Elements

An element represents an action that the flow can execute. Examples include reading or writing Salesforce data, displaying information and collecting data from flow users, executing business logic, or manipulating data.

Flow Builder gives you the option of building flows in free-form or in auto-layout. In free-form, the Elements tab shows the types of elements that you can add to the flow by dragging them onto the canvas. In auto-layout, click ⊕ to display the types of elements that you can add. For a list of all elements already added to the flow, see the Elements section of the Manager tab.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Elements: Action
Launch an action that's available in Salesforce by adding an Action element to your flow.

Flow Element: Add Prompt Instructions
Provide data in the form of prompt instructions that are merged into a prompt template in Prompt Builder. This element is available only in template-triggered prompt flows.

Flow Element: Apex Action
Call an Apex class. Apex classes are available as Apex actions only if one of the methods is annotated with `@InvocableMethod`.

Flow Element: Apex Action (Legacy)
Call an Apex class that uses a legacy Apex interface. Apex classes are available as legacy Apex actions only if the `Process.Plugin` interface has been implemented.

Flow Element: Assignment
Set values in variables, including collection variables, record variables, record collection variables, and global variables.

Flow Element: Collection Filter
Apply criteria to a collection, and then output a new collection that contains only the items that meet the criteria.

Flow Element: Collection Sort
Reorder the items within a collection and optionally limit the number of items that remain in the collection after the sort.

Flow Element: Create Records
Create or update multiple Salesforce records by using a record collection variable. Create or update exactly one Salesforce record by using a record variable or other values from the flow.

Flow Element: Custom Error
Create targeted error messages in record-triggered flows to display in a window on the overall record page or as an inline error on a specific field for your users. The change that triggered the flow is rolled back until the error is fixed.

Flow Element: Get Records
Find Salesforce records that meet filter conditions, and store values from the records in variables.

Flow Element: Decision
Evaluate a set of conditions, and route users through the flow based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

Flow Element: Delete Records
Identify Salesforce records to delete by using the IDs stored in a record variable or record collection variable, or by specifying conditions.

Flow Element: Email Alert
Send an email using an Email Alert action where you specify an email template and a static list of recipients. You add an Action element to your flow and search for the name of your already configured Email Alert action.

Flow Element: Loop

Start a loop path for iterating over items in a collection variable. For each iteration, the flow temporarily stores the item in the loop variable. To execute actions on each item's field values, use other elements in the loop.

Flow Elements: Wait

Resume a flow interview after specific conditions are met, a specified amount of time passes, or until a specific date.

Flow Element: Recommendation Assignment

Generate Einstein Next Best Action recommendations by combining data from records in the recommendation object, records in other objects, values in collections, and manually entered values.

Flow Element: Screen

Collect information from or display information to a user who runs the flow.

Flow Element: Start

Connect the Start element to the flow element that you want to execute first at run time. In an autolaunched flow, you can open the Start element to add a trigger that launches the flow. Without a trigger, you must set up other things to invoke the autolaunched flow, such as custom buttons, processes, Apex classes, or Einstein Bots.

Flow Element: Subflow

Launch another active flow that's available in your org. A flow launched by another flow is called the *referenced flow*.

Flow Element: Transform

Select the flow resources for mapping and transforming source data to target data. You can use the Transform element in screen flows, autolaunched flows with no triggers, and record-triggered flows.

Flow Element: Update Records

Identify Salesforce records to update, and set the values to change in those records. To do so, use the IDs and field values stored in a record variable or record collection variable, or use specify conditions to identify the records and set the field values individually.

Flow Builder Elements for Marketing Cloud

A Flow Builder element represents an action that a flow can execute. Examples include decisions based on criteria and creating and deleting Salesforce data. Some Flow Builder elements are available only in Marketing Cloud, such as Send Email Message and Send SMS Message.

SEE ALSO:

Flow Resources

Flow Builder Tour

Add and Edit Elements

Flow Elements: Action

Launch an action that's available in Salesforce by adding an Action element to your flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Usage

Add an Action element on page 74 to your flow. Then, in the Action field, search for and select the action you want to perform or filter by category or type of action. Flow Builder displays descriptions for action inputs and outputs. See Provided Flow Core Actions on page 382. If something goes wrong, go back to select a different action.

Set Input Values for the Selected Action

To set the input values for the referenced action, use values from earlier in the flow. Assign values for all required inputs. To assign values to optional inputs, select **Include** or **Include with Specified Value** for the toggle associated with the input.

Store Output Values

To use the referenced action's outputs, either use automatic output or assign manual variables. You can't store output values using both automatic output and manually assigned variables.

To use automatic output to reference the action's outputs later in the flow, select the desired output from the resource: Outputs from *ElementName*.

To manually assign the referenced action's outputs to variables, expand **Advanced**, and assign variables to the needed outputs. To return the referenced action's outputs from a flow, manually assign variables defined with **Available for output** selected.

SEE ALSO:

- Flow Elements
- Add and Edit Elements
- Provided Flow Core Actions
- *Trailhead*: Data and Actions in Flows

Flow Element: Add Prompt Instructions

Provide data in the form of prompt instructions that are merged into a prompt template in Prompt Builder. This element is available only in template-triggered prompt flows.

Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

Field	Description
Prompt Instructions	The text and flow resources that the flow evaluates at run time. The text and resource values are stored in the $Output global variable. In the flow, the first Add Prompt Instructions element adds text and flow resources to the $Output global variable. Each subsequent Add Prompt Instructions element appends text and flow resources to the $Output global variable.

EDITIONS

Available in: Lightning Experience

Available in: **Unlimited+** Edition

Available for an additional cost in: **Enterprise** and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on.

Example: The first Add Prompt Instructions element adds the text `Hello!`. The next Add Prompt instructions element appends the {!contact} record variable resource. When the flow finishes, the $Output global variable contains the text, `Hello! Mary`, where `Mary` is the value that {!contact} references. The same text is merged into the associated prompt template in Prompt Builder.

Handle Missing Data in Prompt Instructions

Consider logic or actions that return no data in prompt instructions. Consider these options:

- Incorporate an alternative or default value.

- Add instructions that don't use the data.
- Clear the prompt instructions, so nothing is passed outside the flow.
- Include prompt instructions that address the missing data.

For example, your prompt instructions depend on accounts that meet filter criteria, but the flow finds no accounts. We recommend using logic to handle the missing data. The Decision element can check for data, and the Add Prompt Instructions element can send contextual instructions that no data is available.

Get Record IDs

When you insert a resource that references the Get Records element in the Add Prompt Instructions element, the $Output global variable stores only the record IDs. For example, the Get Records element retrieves a collection of accounts. Its flow resource {!Get_Accounts} is inserted into the Add Prompt Instructions. The $Output global variable stores only the account IDs.

SEE ALSO:

Template-Triggered Prompt Flows

Flow Element: Apex Action

Call an Apex class. Apex classes are available as Apex actions only if one of the methods is annotated with `@InvocableMethod`.

Add an Action element to the flow. Filter the list of actions by type rather than category. If your canvas is in free-form layout, select **Apex**. If your canvas is in auto-layout, select **Apex Action**. Select the action that you want to configure. For details about creating Apex actions, see "`InvocableMethod` Annotation" in the Apex Developer Guide.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

When you set the inputs for the Apex action, use values from earlier in the flow.

Apex actions don't support lookup fields in record variables as input values.

Store Output Values

To reference output values that are stored automatically, specify the API name of the Action element. To store the action's output values manually, select **Manually assign variables (advanced)**. The values are assigned when the code is executed.

Usage

If a flow invokes Apex, the running user must have the corresponding Apex class assignment in their profile or permission set.

If the invoked method creates, updates, or deletes a record, that action isn't performed until the interview's transaction completes. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

Flow Builder doesn't display descriptions for input and output values. For details about each parameter, ask the Apex developer for more information.

Flow Builder displays Apex actions from managed packages only if the associated method is marked global.

SEE ALSO:

- Add and Edit Elements
- Let Flows Execute Apex Actions
- Customize What Happens When a Flow Fails
- Move and Connect Elements to Change a Flow Route

Flow Element: Apex Action (Legacy)

Call an Apex class that uses a legacy Apex interface. Apex classes are available as legacy Apex actions only if the `Process.Plugin` interface has been implemented.

Add an Action element to your flow. Filter the list of actions by type rather than by category. Select Apex Action (Legacy), and then select the action that you want to configure.

Tip: If your developer hasn't already implemented the `Process.Plugin` interface on the desired class, we recommend using the `@InvocableMethod` annotation instead. Unlike the `Process.Plugin` interface, the `@InvocableMethod` annotation supports sObject, Collection, Blob, and Time data types and bulkification. It's also easier to implement. To compare the interface and the annotation, see Let Flows Execute Apex Actions on page 171.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

When you set the inputs for the Apex action, use values from earlier in the flow.

Store Output Values

To use the legacy Apex action's outputs later in the flow, store them in variables. The values are assigned when the code is executed.

Usage & Limitations

If the Apex class creates, updates, or deletes a record, the action isn't performed until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

Flow Builder doesn't display descriptions for input and output values. For details about each parameter, ask the Apex developer for more information.

Legacy Apex actions aren't organized by the tag in the plug-in code.

Cloud Flow Designer lets you save an Apex Plug-in element without setting values for its required input parameters. If you open the corresponding legacy Apex action in Flow Builder, you can't save changes to the element unless you set values for the required input parameters.

Important: Legacy Apex actions aren't supported in auto-layout in Flow Builder. Legacy Apex actions are only available to be added in free-form in Flow Builder. Existing actions can be edited in both auto-layout and free-form mode.

SEE ALSO:

- Add and Edit Elements
- Let Flows Execute Apex Actions
- Customize What Happens When a Flow Fails
- Move and Connect Elements to Change a Flow Route
- *Apex Reference Guide* : Process Namespace

Flow Element: Assignment

Set values in variables, including collection variables, record variables, record collection variables, and global variables.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Usage

To update the value of a variable, add an Assignment element to your flow. Specify the API name of a variable, an operator, and the value to use.

To update the value of more than one variable in an Assignment element, click Add Assignment. For each row, specify the variable, the operator, and the value to assign. At run time, variable assignments are made consecutively in the order they're listed in the element.

Field	Description
`Variable`	The API name of the variable you want to assign a value to. Select an existing variable, or create one.
`Operator`	The operation to perform for the assignment. The available operators depend on the data type of the specified `Variable`. See Flow Operators in Assignment Elements
`Value`	The value to use in the assignment or the API name of a resource that contains the value to use in the assignment. `Variable` and `Value` in the same row must have compatible data types.

Example: **Update a Record Variable and Add It to a Record Collection**

To set the field values for a record variable and then add the record variable to a record collection, use an Assignment element on a Loop element's For Each path. To create all the records at the same time, use the record collection variable with a Create Records element outside of the loop.

This example updates the Account ID, Amount, Description, and Stage fields of the NewOpportunity record variable and then adds the record variable to the NewOpportunities record collection variable.

This example contains five variables:

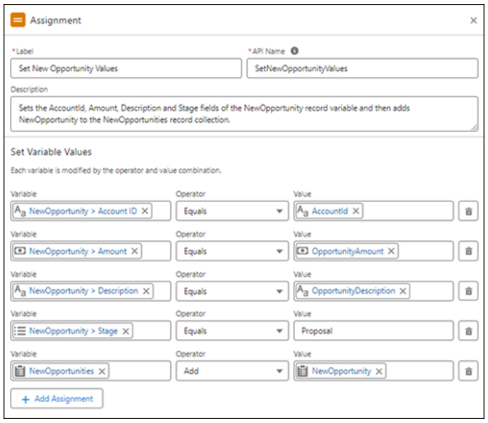
- NewOpportunity: **Data Type** is *`Record`* and **Object** is *`Opportunity`*.
- AccountId: **Data Type** is *`Text`*.
- OpportunityAmount: **Data Type** is *`Currency`*.

- OpportunityDescription: **Data Type** is *`Text`*.
- NewOpportunites: **Data Type** is *`Record`* and **Object** is *`Opportunity`*. **Allows multiple values**is selected.

The Assignment element in this example has five rows:

- The first row sets the record variable's account ID to a variable value:
  - **Variable** is *`NewOpportunity > Account ID`*
  - **Operator** is *`Equals`*
  - **Value** is *`AccountId`*
- The second row sets the record variable's amount to a variable value:
  - **Variable** is *`NewOpportunity > Amount`*
  - **Operator** is *`Equals`*
  - **Value** is *`OpportunityAmount`*
- The third row sets the record variable's description to a variable value:
  - **Variable** is *`NewOpportunity > Description`*
  - **Operator** is *`Equals`*
  - **Value** is *`OpportunityDescription`*
- The fourth row sets the record variable's stage to a literal value from the picklist associated with the Stage field:
  - **Variable** is *`NewOpportunity > Stage`*
  - **Operator** is *`Equals`*
  - **Value** is *`Proposal`*
- The fifth row adds the record variable to the record variable collection:
  - **Variable** is *`NewOpportunities`*
  - **Operator** is *`Add`*
  - **Value** is *`NewOpportunity`*

Here's an Assignment element that sets properties of an opportunity record variable and then adds the variable to an opportunities record collection



Assignment

*Label: Set New Opportunity Values

*API Name: SetNewOpportunityValues

Description: Sets the AccountId, Amount, Description and Stage fields of the NewOpportunity record variable and then adds NewOpportunity to the NewOpportunities record collection.

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
NewOpportunity > Account ID	Equals	AccountId
NewOpportunity > Amount	Equals	OpportunityAmount
NewOpportunity > Description	Equals	OpportunityDescription
NewOpportunity > Stage	Equals	Proposal
NewOpportunities	Add	NewOpportunity

Example: Set the Value of an Error Message in a Fault Path

To set a variable equal to a relevant error message, use an Assignment element on a Fault path. Next, use a Go To connector to go to a Screen element that displays errors to the flow user. The Screen element uses the variable from the Assignment element in a Display Text component.

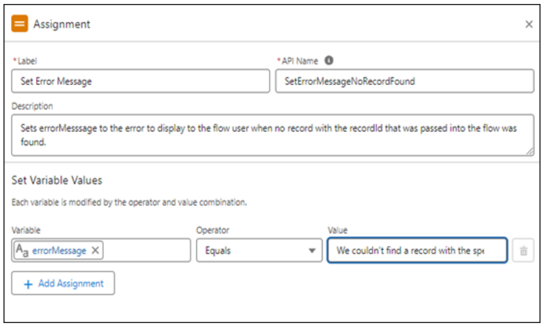
This example sets the value of the ErrorMessage variable to a string that identifies the problem and what to do about it.

This example contains one variable: ErrorMessage where **Data Type** is `Text`.

This example contains one row:

- **Variable** is `ErrorMessage`
- **Operator** is `Equals`
- **Value** is `We couldn't find a record with the specified record ID. Check the record ID and try again.`

Here's an Assignment element that sets the value of a variable to a literal string



Assignment

*Label: Set Error Message

*API Name: SetErrorMessageNoRecordFound

Description: Sets errorMesssage to the error to display to the flow user when no record with the recordId that was passed into the flow was found.

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
errorMessage	Equals	We couldn't find a record with the spe

SEE ALSO:

- Flow Elements
- Flow Operators in Assignment Elements
- Flow Element: Create Records
- Flow Element: Loop
- Move and Connect Elements to Change a Flow Route
- Flow Resources

Flow Element: Collection Filter

Apply criteria to a collection, and then output a new collection that contains only the items that meet the criteria.

Field	Description
`Collection`	The collection variable that is filtered. This field accepts any collection variable within the same flow.
`Condition Requirements`	Determines the logic that evaluates conditions. • Choose `All Conditions Are Met` to include values that meet all the specified criteria.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
	- Choose `Any Condition Is Met` to include values that meet any of the specified criteria. - Choose `Custom Condition Logic Is Met` to include values that meet the logic entered in `Condition Logic`. - Choose `Formula Evaluates to True` to include values where `Formula` is true for that value.
`Condition Logic`	Only appears if `Condition Requirements` is set to `Custom Condition Logic Is Met`. Enter logic such as *`1 AND (2 OR 3)`*.
`Formula`	Only appears if `Condition Requirements` is set to `Formula Evaluates to True`. Enter a formula that can evaluate to TRUE or FALSE, such as *`{!currentItemFromSourceCollection.Id} == {!varContactID}`*.
`Field`	The field evaluated by this condition. Doesn't appear if `Condition Requirements` is set to `Formula Evaluates to True`.
`Operator`	The available operators depend on the data type of the selected `Field`.
`Value`	`Field` and `Value` in the same row must have compatible data types. Options:

Usage

You can filter any collection found in Flow Builder, including collection variables that contain single values, collection variables that contain records, and Apex-defined collection variables.

Collection Filter outputs a collection with the filtered results and doesn't change the contents of the source collection. The output collection is null until its corresponding Collection Filter runs.

Note: The output collection variable is named after the Collection Filter element's API name.

Example:

- For example, if a Collection Filter element is named *`FilterLeads`*, its output collection is called *`Leads from FilterLeads`*.
- The Collection Filter element also creates a single variable called *`CurrentItem_FilterLeads`*. It acts as a loop variable for the Collection Filter element's input collection.
- You can reference the single variable in a formula resource. For example, you can create a Filter element formula condition where you set condition requirements to *`Formula Evaluate to True`*: AND ({currentItem_FilterLeads.LastViewedDate} < {!$Flow.CurrentDateTime,{!currentItem_FilterLeads.IsConverted})

If you filter your collection with a formula, the formula must evaluate to a boolean (true or false) value. For more formula considerations, see Flow Formula Considerations in Salesforce Help.

If you delete a Collection Filter element, the *`CurrentItem_FilterLeads`* variable remains in the flow. You can safely delete this single record variable after you remove the collection filter element.

Considerations for Defining Filter Criteria

- When you define multiple filter criteria, the filter logic usually defaults to AND. But if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

For example, your filters check whether a case Type equals Problem, Type equals Feature Request, and Escalated equals true. At run time, the filters are combined to be `Type = (Problem OR Feature Request) AND Escalated = true`.

- The available filter operators depend on the data type of the selected fields. For details, see Flow Operators in Data Elements and Record Choice Sets.

SEE ALSO:

Flow Formula Considerations

Flow Operators in Data Elements and Record Choice Sets

Creating Flow Formulas with Flow Formula Builder

Flow Element: Collection Sort

Reorder the items within a collection and optionally limit the number of items that remain in the collection after the sort.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Collection Variable`	The collection variable that is sorted. This field accepts any collection variable within the same flow.
`Sort By`	The field that the collection is sorted by. This field is only shown if the collection variable contains more than one field.
`Sort Order`	Sort the collection in ascending or descending order.
`Put empty string and null values first`	When selected, this element sorts records with an empty or null value in the Sort By field at the start of the collection. Otherwise, they're placed at the end.
`How Many Items to Keep After Sorting`	Select `Set the maximum number of items` to determine the number of items that remain in the collection after the sort.

Usage

When the Collection Sort element removes values or changes their order, it makes those changes directly in the selected collection variable.

If the collection variable contains more than one field, click `Add Sort Option` to sort by additional fields in order of greater to lesser priority. You can sort by up to 3 fields at a time.

Flow Element: Create Records

Create or update multiple Salesforce records by using a record collection variable. Create or update exactly one Salesforce record by using a record variable or other values from the flow.

Note: Looking for the Fast Create and Record Create elements from Cloud Flow Designer? The Create Records element combines the functionality of both elements. For the equivalent of the Record Create element, create one record and set the record fields using separate variables, resources, and literal values. Choosing the other options is the equivalent of the Fast Create element.

How many records you choose to create or update and how to set the field values determine what to enter in the rest of the Create Records element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

To create a collection of records

To create multiple records, you must use the values from a record collection variable. Earlier in the flow, populate the record collection variable with the new records' field values. Make sure the ID field is blank.

To dynamically create some records and update others in a collection, enable **Update Existing Records**. Choose a field on the records in the record collection variable that uniquely identifies each record. The flow uses this field to check whether the records exist. Choose how to process the remaining records if the flow fails to create or update a record.

When you use a record collection variable to create or update multiple records at once, you reduce the number of DML requests in your flow. That means you're more likely to stay within your org's limits. For more information, see Flow Bulkification in Transactions.

To create a single record

If you've populated a record variable with the field values for the new record, choose to set the record fields by using all the values from a record variable. Then select the record variable to use. Make sure the ID field is blank.

To dynamically create or update a record, enable **Update Existing Records**. Choose a field on the record in the record variable that uniquely identifies the record. The flow uses this field to check whether the record exists.

To manually map values from various resources in the flow, choose to set the record fields by using separate variables, resources, and literal values. Choose the object that you want to create a record for, and set the field values for the record. Optionally, store the ID of the created record in a Text variable if you manually created the variable. For example, when you need the record's ID to create child records or to provide a link to the created record.

To dynamically check for a duplicate record to update or create, enable **Check for Matching Records**.

Example: A user enters a name and address into the flow. Verify that a matching user exists by using the Get Records element. If a matching contact doesn't exist, create a record for that user by using the Create Records element.

Usage

To prevent the flow from failing, make sure that:

- All required fields are populated with values. The Create Records element populates potentially required fields for you. The fields shown are required in the master record type. For custom objects, confirm the required fields in the object definition.
- For record variables the ID field values are blank. The flow populates the ID fields after the record is created.

Note: The record isn't created until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

Considerations

- If a Create Records element uses a record collection, doesn't have a fault path, and the flow fails, no records are created, and the flow stops, generating an error.
- If a Create Records element uses a record collection and has a fault path, only the successful records are created. The IDs of the successful records aren't populated for the records in the record collection in the flow. The IDs are populated on the records in the org.
- The Create Records element can't update matching records that are locked for editing. If Flow Builder finds a matching record and attempts to update a field in the record, a warning appears.
- The Create Records element can't update read-only fields in matching records. If Flow Builder finds a matching record and attempts to update a field that's always read only or that you don't have permission to update, a warning appears.
- The Create Records element can't update fields in matching records for objects that don't support the update function. If Flow Builder finds a matching record and attempts to update a field in an object that doesn't support the update function, an error appears, and you can't activate the flow. To determine whether an object can be updated, see *Object Reference for the Salesforce Platform*.

SEE ALSO:

Flow Operators in Data Elements and Record Choice Sets

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Elements

Check for Duplicate Records

To prevent duplicate records, check for records that match a set of criteria and specify what happens if the flow finds matching records. Some field-level configurations and validations in your org override the settings in the Create Records element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

1. Enable **Check for Matching Records**.
2. In Condition Requirements, select an option.
3. Configure the first condition.
4. Add more conditions as needed.
5. In the If a single matching record exists area, select an option.

Option	Description
Update the matching record	Update the matching record with the values that you specified in the Create Records element.
Skip the matching record	Don't create or update any records.

6. In the If multiple matching record exists area, select an option.

Option	Description
Update the most recently modified matching record	Update the most recently modified matching record with the values that you specified in the Create Records element.
Skip all matching records	Don't create or update any records.

7. Save your work.

Flow Element: Custom Error

Create targeted error messages in record-triggered flows to display in a window on the overall record page or as an inline error on a specific field for your users. The change that triggered the flow is rolled back until the error is fixed.

You can create a custom error message only in the before-save or after-save path of a record-triggered flow. You can't run an error message asynchronously, and the error message can't be called from another flow type.

EDITIONS

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

FIELD	DESCRIPTION
`Label`	Identifies the error message on the canvas.
`API Name`	The API name must be unique within the current flow. Two elements can have the same API name if they're used in different flows. The name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Describes the error message.
`Where to Show the Error Message`	Select **In a window on a record page** to display the error message as an overall message. Select **As an inline error on a field** to display the error message on the field that is causing the error.
`Error Message`	Enter text or select a resource to create an error message to display to the user when there's an error on a record change. The error message can have up to 255 characters. You can use Translation Workbench to translate your error messages.

Usage

Use the Custom Error element to roll back a change that triggered a flow and inform the user exactly what caused the error. The user can fix the problem and try again. For example, when a user deletes a record that triggers a flow, the flow can return an error message that tells the user why the deletion wasn't allowed.

Considerations

- A Custom Error element can contain only one record page error message. To create another record page error message in the same flow, use another Custom error element.
- A field can have only one error message, but each field can have an error message.
- Compound fields aren't supported.
- If an executed fault path has a Custom Error element, the change that triggered the flow is rolled back.
- Custom error messages use the same functionality as the addError() Id method in Apex.

SEE ALSO:

Salesforce Developers: Apex Reference Guide

Flow Element: Get Records

Find Salesforce records that meet filter conditions, and store values from the records in variables.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: Looking for the Fast Lookup and Record Lookup elements from Cloud Flow Designer? The Get Records element combines the functionality of both elements. For the equivalent of the Record Lookup element, store only the first record in separate variables. Choosing the other options is the equivalent of the Fast Lookup element.

Identify the object whose records you want to find, and specify conditions to narrow down the list of returned records. How many records you choose to store and where to store the field values determines what to enter in the rest of the Get Records element. When you add a Get Records element to a screen flow or an autolaunched flow, we automatically store all the record values in a flow variable. When the flow moves to the next element, the values are assigned to the variable.

To store record values manually in a screen flow or autolaunched flow, select **Choose fields and assign variables (advanced)**.

To store field values manually for only the first record

Tip: If you choose to store values from only the first record, filter by a unique field, such as ID. Otherwise, you can't guarantee which record's field values are stored.

You can store field values from the first record together in a record variable or in separate variables.

To store the values together, select the record variable, and identify the fields whose values you want to store.

To store the values in separate variables, select each field you want to store, and select the variable to store each field value in.

To store field values manually for more than one record

To store field values for multiple records, you must store the values in a record collection variable.

When you use a record collection variable to create, update, or delete multiple records at once, you reduce the number of DML requests in your flow. That means you're more likely to stay within your org's limits. For more information, see Flow Bulkification in Transactions.

Example:

- Find the product name or description for a product with a specific bar code.
- Confirm stock availability for a particular item.
- Verify a caller's identity.

Considerations for Defining Filter Criteria

- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

  For example, your filters check whether a case's Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At run time, the filters are combined to `(1 OR 2) AND 3`.

- The available filter operators depend on the data type of the selected fields. For details, see Flow Operators in Data Elements and Record Choice Sets.

- To use a Data Cloud object as the data source, your org must include a data mapping for the object. Data mapping relates Data Lake Object (DLO) fields to Data Model Object (DMO) fields. If your org doesn't include a mapping, you can't select a Data Cloud data space or object in the Get Records element.

SEE ALSO:

Flow Operators in Data Elements and Record Choice Sets

Flow Elements

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Element: Decision

Evaluate a set of conditions, and route users through the flow based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Outcomes

For each path that the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path. To relabel the path that the flow takes if no outcome's conditions are met, click **Default Outcome**.

Field	Description
`Label`	Identifies the connector for this outcome on the canvas.
`Outcome API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Condition Requirements to Execute Outcome`	Determines whether the flow takes this outcome's path. Sets logic and conditions for each outcome that determine if the flow follows its path.
`When to Execute Outcome`	Available on record-triggered flows. Determines whether this outcome's path is taken, based on whether the triggering record is updated to meet the condition requirements. For example, the opportunity update that triggered the flow to run changed its stage to Closed Won from any value that isn't Closed Won. This option checks if the triggering record didn't previously meet the condition requirements and if the $Record variable now meets the condition requirements. If your flow changes any of the $Record variable's fields before it runs the configured Decision element, the Decision checks if the $Record's new field values now meet the condition requirements.

Example: Using a Decision element, determine whether to:

- Give customers a return shipping address or instructions on how to resolve the problem when an item is determined to be faulty.
- Offer a customer a loan based on the results of a credit scoring formula.
- Inform sales leaders when an opportunity's stage is changed to Closed Won.

Tip: Configure your flow so that it does different things based on what a user selects for a Picklist screen component. To do so, add a decision after the screen to create the branches of the flow based on the choices available in the picklist. Then you can represent each choice in your decision and connect it to a branch of your flow.

Usage

When a flow executes a Decision element, it evaluates each decision outcome in order. For the first outcome whose conditions are met, the flow takes the associated path. If no outcome's conditions are met, the flow takes the path for the default outcome.

SEE ALSO:

Flow Elements

Define Conditions in a Flow

Flow Operators in Decision, Wait, and Collection Filter Elements

Move and Connect Elements to Change a Flow Route

Flow Element: Delete Records

Identify Salesforce records to delete by using the IDs stored in a record variable or record collection variable, or by specifying conditions.

Note: Looking for the Fast Delete and Record Delete elements from Cloud Flow Designer? The Delete Records element combines the functionality of both elements.

- For the equivalent of the Fast Delete element, use the IDs from a record variable or record collection variable.
- For the equivalent of the Record Delete element, specify the conditions to identify the records to delete.

How you choose to identify the records to delete determines what to enter in the rest of the Delete Records element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Use a record variable or record collection variable.

  If you store the IDs of the records to delete in a record variable or record collection variable, choose to use the IDs from a record variable or record collection variable. Then select the variable to use.

  Important: For the variable that you select, make sure that each record's ID value is set. The flow identifies which records to delete based on the ID value.

  When you use a record collection variable to delete multiple records at once, you reduce the number of DML requests in your flow. That means you're more likely to stay within your org's limits. For more information, see Flow Bulkification in Transactions.

- Specify conditions.

  To use conditions to identify the records to delete, choose the object, and add at least one condition to filter down the list of records.

Example: When a customer accepts a quote, delete the remaining quotes from the opportunity.

Considerations for Defining Filter Criteria

- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

  For example, your filters check whether a case's Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At run time, the filters are combined to `(1 OR 2) AND 3`.
- The available filter operators depend on the data type of the selected fields. For details, see Flow Operators in Data Elements and Record Choice Sets.

Usage

Warning:

- Be careful when testing flows that contain delete elements. Even if the flow is inactive, it triggers the delete operation.
- To prevent deleting records by mistake, be as specific in your filter criteria as possible.
- Records are deleted from your org the moment the flow executes the delete element.
- Deleted records are sent to the Recycle Bin and remain there for 15 days before they're permanently deleted.
- Flows can delete records that are pending approval.

Note: At run time, the record isn't deleted until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

SEE ALSO:

Flow Operators in Data Elements and Record Choice Sets

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Elements

Flow Element: Email Alert

Send an email using an Email Alert action where you specify an email template and a static list of recipients. You add an Action element to your flow and search for the name of your already configured Email Alert action.

Note: If you're using Marketing Cloud Growth, use the Send Email Message action instead of the Email Alert action. Email Alert actions don't work with segments.

Before you begin:

- Create a Lightning email template or Classic email template. Specify the recipient record in any merge fields you use. For example, you could use the contact record or the lead record. When you build the email Alert, you match the object to the referenced record in the flow. For example, if the email alert references the Lead object, use a lead record ID when configuring the element in the flow.
- In Setup, search for Email Alerts and configure your email alert on page 813. See Email Alert Actions in the Actions Developer Guide.
- Understand the daily limits for emails sent from email alerts.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Determine which record you want to reference in the email and use that record's ID in the element. For example, if you're sending the email to the contact that triggered the flow, use the ID of the triggering contact. Or, if you're using a Get Records element to get the record that receives the email, use the ID of the record found in the Get Records element. If the email alert has merge fields, the referenced record is the starting point for those fields.

Add an Action element to your flow. Search for your email alert using the name of the email alert and select the email alert to configure. To filter the list of email alerts by a specific object, enter the API name of the object. The unique name for each email alert is prefixed with its object. For example, email alert `emailAlert-Account.Owner_Changed` is associated with the Account object.

Set Input Values

Field	Description
`Record ID`	Select a resource that contains the ID for the record that you want the email to reference. If the email alert uses merge fields, this record is the starting point for those merge fields. This field accepts single-value resources of any type. The value is treated as text. The object of the referenced record must match the object of the email alert.

Usage

At run time, the email isn't sent until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

Flow Builder displays email alerts from managed packages only if the email alert isn't protected.

SEE ALSO:

Options for Sending Emails from Flows

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Daily Allocations for Email Alerts

Flow Element: Loop

Start a loop path for iterating over items in a collection variable. For each iteration, the flow temporarily stores the item in the loop variable. To execute actions on each item's field values, use other elements in the loop.

A *collection* is a list of items, such as field values or email addresses. A loop uses a *loop variable* to store the values for the current item in the collection. When the loop finishes examining an item, it copies the field values for the next item into the loop variable. To reference each collection item in elements along the loop path, use the loop variable. To keep changes made along the loop path, add the loop variable as an item in a new collection variable.

Field	Description
`Collection Variable`	The collection that you want to loop through. This field accepts any collection variable.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Direction`	Determines whether the flow starts with the first item or the last item in the collection variable.
`Loop Variable`	The variable that the flow uses to contain the current item's values during a loop iteration. • If `Collection Variable` is set to a non-record collection variable, this field accepts a variable with the same data type. • If `Collection Variable` is set to a record collection variable, this field accepts a record variable with the same object type.

Usage

After you add a Loop element and the elements that you want the loop to include, from the Loop element:

- Determine which element to execute first when a new item's values are copied into the loop variable by using the "For each item" connector.
- Determine which element to execute after the loop has processed all the items in the collection by using the "After last item" connector.

Sample Flow That Loops Through a Collection
Transfer ownership of accounts from one user to another by using record collection variables and loops. The flow already has the required user IDs.

SEE ALSO:

Move and Connect Elements to Change a Flow Route

Flow Elements

Flow Resource: Variable

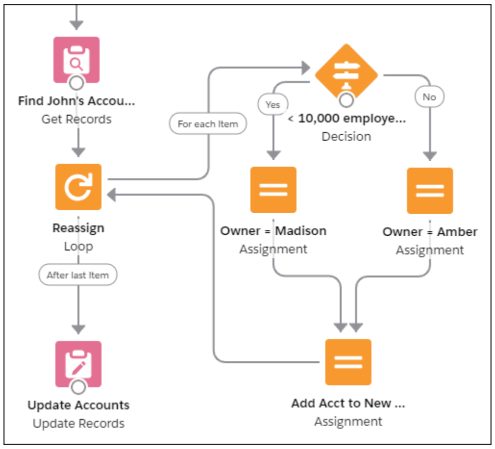
Sample Flow That Loops Through a Collection

Transfer ownership of accounts from one user to another by using record collection variables and loops. The flow already has the required user IDs.

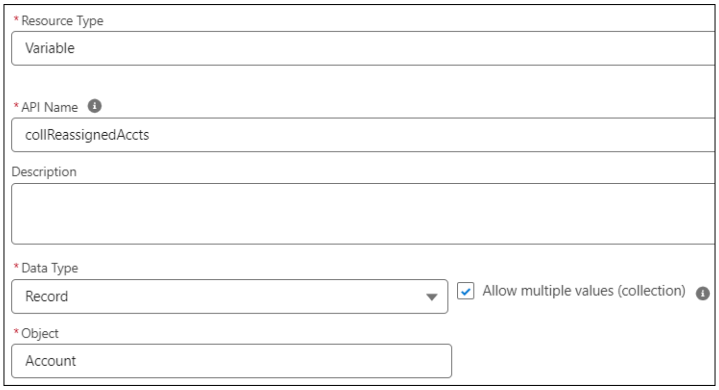
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



First, create an Account-based record collection variable called `collReassignedAccts`.



* Resource Type
Variable

* API Name
collReassignedAccts

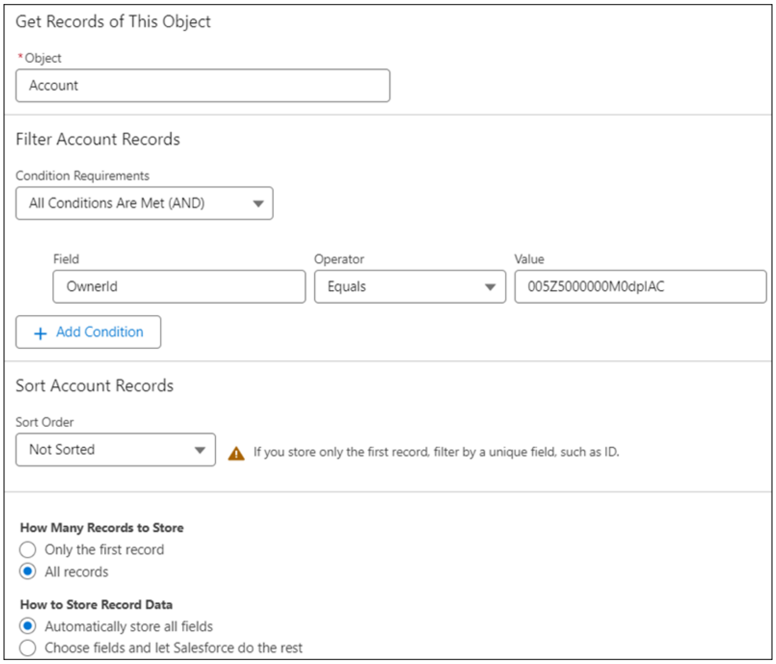
Description

* Data Type
Record

☑ Allow multiple values (collection)

* Object
Account

Next, add the Get Records element to get all account records that John Smith owns.



Get Records of This Object

* Object
Account

Filter Account Records

Condition Requirements
All Conditions Are Met (AND)

Field	Operator	Value
OwnerId	Equals	005Z5000000M0dpIAC

+ Add Condition

Sort Account Records

Sort Order
Not Sorted

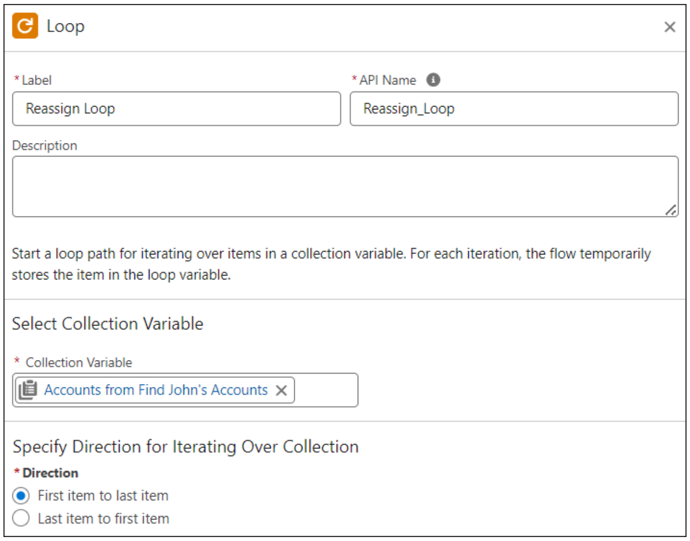
If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store
○ Only the first record
● All records

How to Store Record Data
● Automatically store all fields
○ Choose fields and let Salesforce do the rest

Then create a loop that iterates through the collection. For each item in the collection, the loop:

1. Assigns the collection item to the loop variable.



Loop

* Label: Reassign Loop

* API Name: Reassign_Loop

Description

Start a loop path for iterating over items in a collection variable. For each iteration, the flow temporarily stores the item in the loop variable.

Select Collection Variable

* Collection Variable: Accounts from Find John's Accounts

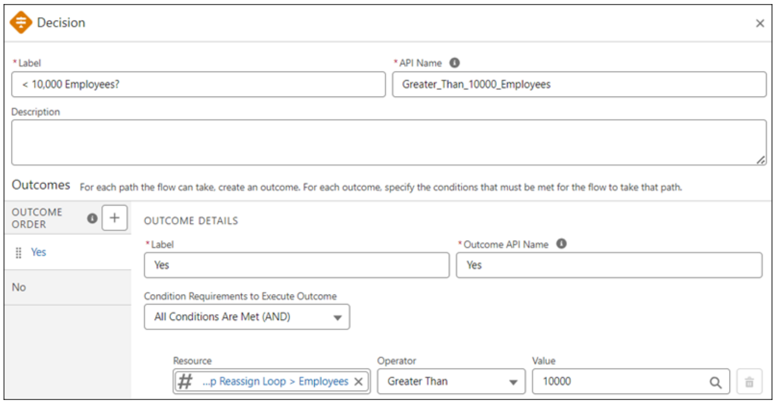
Specify Direction for Iterating Over Collection

* **Direction**

● First item to last item

○ Last item to first item

2. Evaluates whether the account has more than 10,000 employees.



Decision

* Label: < 10,000 Employees?

* API Name: Greater_Than_10000_Employees

Description

Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

OUTCOME ORDER: Yes, No

OUTCOME DETAILS

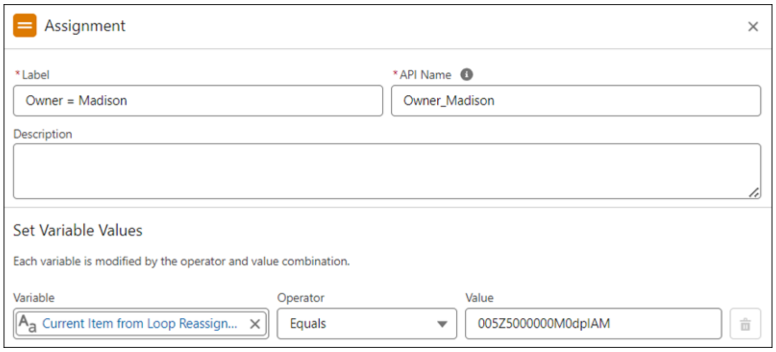
* Label: Yes

* Outcome API Name: Yes

Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource	Operator	Value
...p Reassign Loop > Employees	Greater Than	10000

3. If the account has more than 10,000 employees, assigns Madison's user ID to the `OwnerId` field in the loop variable.



Assignment

* Label: Owner = Madison

* API Name: Owner_Madison

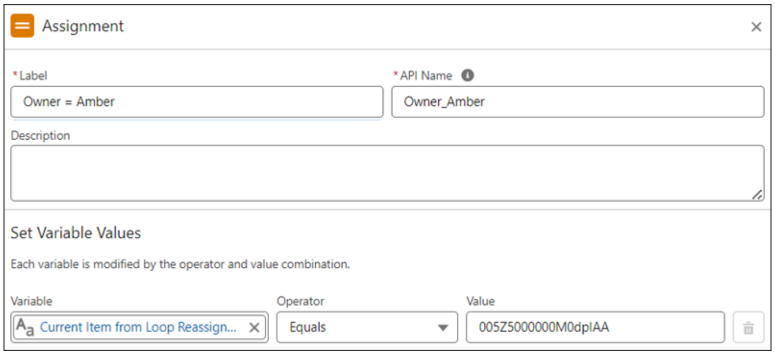
Description

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
Current Item from Loop Reassign...	Equals	005Z5000000M0dpIAM

4. If the account doesn't have more than 10,000 employees, assigns Amber's user ID to the `OwnerId` field in the loop variable.



Assignment

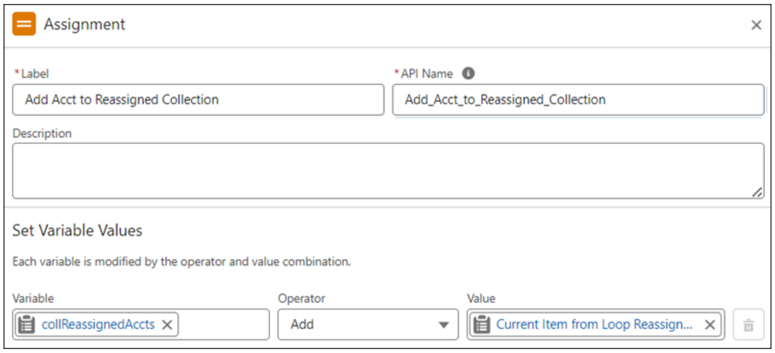
*Label: Owner = Amber
*API Name: Owner_Amber
Description:

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
Current Item from Loop Reassign...	Equals	005Z5000000M0dpIAA

5. Adds the loop variable's values as a new item in the `collReassignedAccts` collection.



Assignment

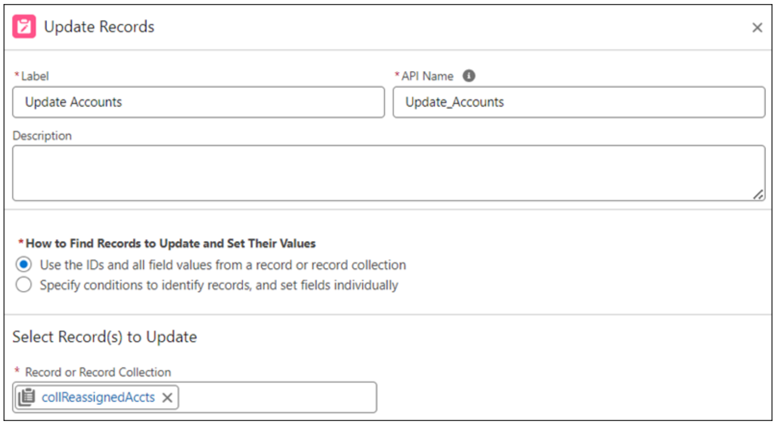
*Label: Add Acct to Reassigned Collection
*API Name: Add_Acct_to_Reassigned_Collection
Description:

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
collReassignedAccts	Add	Current Item from Loop Reassign...

Finally, add an Update Records element to update the accounts in `collReassignedAccts` with the new `OwnerId` after the loop finishes iterating through the collection.



Update Records

*Label: Update Accounts
*API Name: Update_Accounts
Description:

*How to Find Records to Update and Set Their Values
● Use the IDs and all field values from a record or record collection
○ Specify conditions to identify records, and set fields individually

Select Record(s) to Update

* Record or Record Collection: collReassignedAccts

This section of the flow uses a single query to look up the list of accounts and a single DML statement to update those accounts. If you updated the records by setting the fields individually, you would use:

- One Update Records element to find all accounts that John owns and have more than 10,000 employees (1 query). Then update those records' `OwnerId` to Madison's Id (1 DML statement).
- One Update Records element to find all accounts that John owns and don't have more than 10,000 employees (1 query). Then update those records' `OwnerId` to Amber's Id (1 DML statement).

Flow Elements: Wait

Resume a flow interview after specific conditions are met, a specified amount of time passes, or until a specific date.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note:

- Flows that contain Wait elements must be autolaunched. If a flow includes Wait elements and screens, choice, or choice sets, you can't activate or run it.
- Before you add a Wait element to your flow, understand the special behavior and limitations. See Paused Flow Interview Considerations on page 267 for details.

Flow Element: Wait for Conditions

Resume a flow interview after specific conditions are met.

Flow Element: Wait for Amount of Time

Resume a flow interview after a specific amount of time.

Flow Element: Wait Until Date

Resume a flow interview after a specific date.

SEE ALSO:

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Elements

Define Conditions in a Flow

Marketing Cloud Growth Campaign Flow Element: Wait Until Event

Configure the Process Trigger

Flow Element: Wait for Conditions

Resume a flow interview after specific conditions are met.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Each wait configuration corresponds to a wait connector on the canvas. When a flow pauses, it waits for one or more resume events. For the first resume event that occurs, the flow resumes and executes the connector for the associated pause configuration.

Flow Wait Conditions

Each wait configuration that you define in a flow has optional wait conditions. At run time, these conditions determine whether the flow waits for the associated resume event.

Flow Resume Events

Define the event to wait for if the wait conditions are met. When an event occurs, the flow resumes and takes the path associated with this wait configuration.

Sample Flows That Wait for Events

Configure a flow to wait for events in one of four ways.

Flow Wait Conditions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Each wait configuration that you define in a flow has optional wait conditions. At run time, these conditions determine whether the flow waits for the associated resume event.

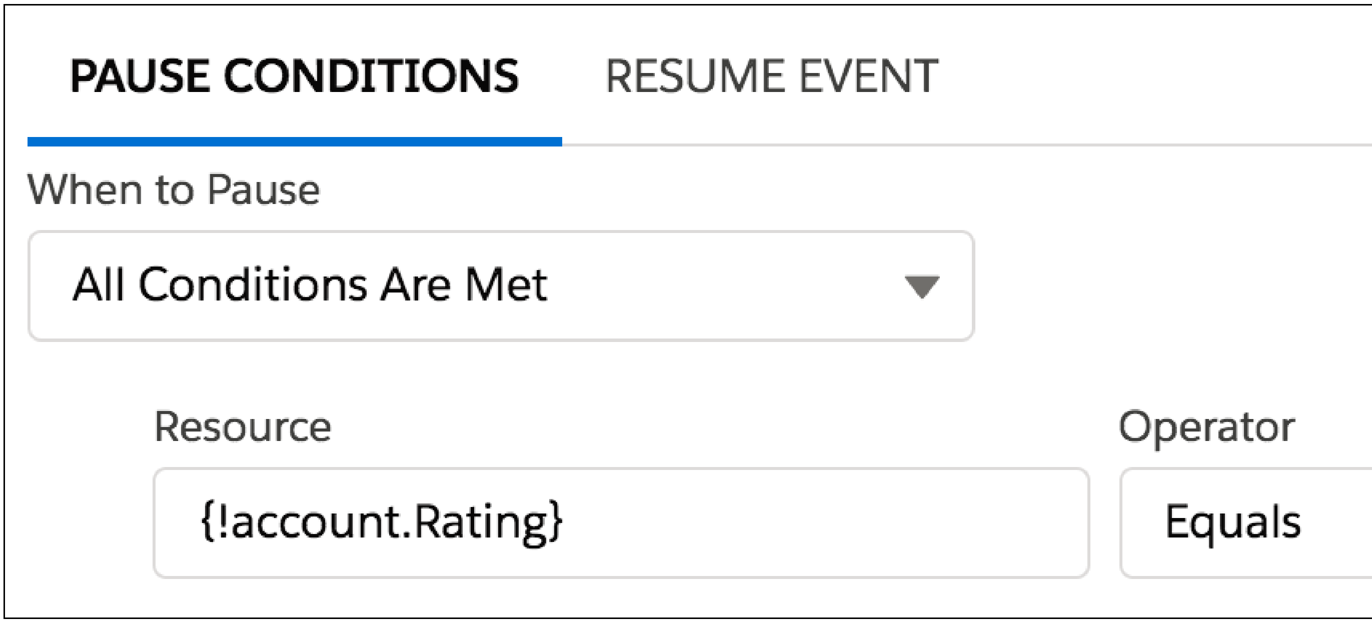
If the wait conditions aren't met for a resume event, the interview doesn't wait for that event. If all resume events have unmet wait conditions, the interview doesn't pause. Instead, it executes the default path.

Example: Use wait conditions when:

- The flow waits for different events based on a field value on a given record.

  For example, send an email reminder to a contract's owner before the contract's end date. However, the date on which you send the email depends on the rating of the contract's account. If the account is hot, send the email a month before the end date. If the account isn't hot, send the email two weeks before the end date.

  For this example, you create two events. The event for hot accounts occurs 30 days before the contract's end date. Its wait conditions check whether the rating for the contract's account is equal to "Hot."



When to Pause: All Conditions Are Met

Resource: {!account.Rating}

Operator: Equals

  The second event occurs 14 days before the contract's end date. Its wait conditions check whether the rating for the contract's account is not equal to "Hot." If the account is hot, the interview doesn't wait for the second event.

- The flow waits for multiple events to occur, such as to send periodic email reminders. For an example of this scenario, see Sample Flow That Pauses Until Multiple Resume Events Occur on page 352.

SEE ALSO:

Define Conditions in a Flow

Flow Elements: Wait

Flow Resume Events

Define Conditions in a Flow

Flow Elements: Wait

Flow Resume Events

Flow Resume Events

Define the event to wait for if the wait conditions are met. When an event occurs, the flow resumes and takes the path associated with this wait configuration.

Flow Resume Event: Specific Time

Resume the paused flow when a specific time occurs.

Flow Resume Event: Platform Event Message

Resume the flow interview when it receives a platform event message.

SEE ALSO:

Flow Elements: Wait

Flow Wait Conditions

Flow Resume Event: Specific Time

Resume the paused flow when a specific time occurs.

Make sure to familiarize yourself with Paused Flow Interview Considerations.

Define Resume Time: Flow-Based Time

When the time source is a specific time, configure the resume time with these fields.

Parameter	Description
`Base Time`	A date/time value. You can manually enter a date/time value or reference a merge field or flow resource.
`Offset Number`	Optional. The number of days or hours to offset the selected field’s value. Required if you provide an offset unit. Manually enter the integer. You can’t use a merge field or flow resource for this value. To resume the flow before the base time, use a negative number. To resume the flow after the base time, use a positive number.

Parameter	Description
`Offset Unit`	Optional. The unit to offset the selected field's value. Required if you provide an offset number. Manually enter *`Days`* or *`Hours`*. You can't use a merge field or flow resource for this value.

Example: To resume the flow 3 days after the flow paused, use the `$Flow.CurrentDate` global variable as the base time, set the offset number to 3, and set the offset unit to Days.

Define Resume Time: Record-Based Time

When the time source is a record field, configure the resume time with these fields. The base resume time is a date/time field value on a record.

Parameter	Description
`Object`	The API name of the object to use for the base resume time. Manually enter the string. You can't use a merge field or flow resource for this value.
`Field`	The API name for a date/time field. The field must belong to the specified object. Manually enter the string. You can't use a merge field or flow resource for this value.
`Record ID`	The ID of the record to use for the base time. The record's object must match what's entered in `Object`. The record's value for the selected field is used as the base resume time. You can enter a string or select a merge field or flow resource.
`Offset Number`	Optional. The number of days or hours to offset the selected field's value. Required if you provide an offset unit. Manually enter the integer. You can't use a merge field or flow resource for this value. To resume the flow before the base time, use a negative number. To resume the flow after the base time, use a positive number.
`Offset Unit`	Optional. The unit to offset the selected field's value. Required if you provide an offset number. Manually enter *`Days`* or *`Hours`*. You can't use a merge field or flow resource for this value.

Example: You want to resume the flow 3 days before a contract ends. To identify the base resume time, set `Object` to *`Contract`*, `Field` to *`EndDate`*, and `Record ID` to {!varContractId}. To offset the base resume time, set the offset number to *`-3`*, and set the offset unit to *`Days`*.

Store Output Values in Variables

Reference information from the resume event in your flow by storing its outputs in flow variables.

Parameter	Description	Example
`Resume Time`	The actual time at which the event occurred and the flow interview resumed.	`11/26/2014 10:12 AM`

Parameter	Description	Example
`Event Delivery Status`	The status of the event when the flow interview resumed. After a resume event occurs, Salesforce delivers data from the event to the flow that's waiting for it so that the flow knows to resume. Valid values are: • Delivered—The event was successfully delivered. • Invalid—An error occurred during delivery, but the flow successfully resumed.	`Delivered`

Record-Based Time: Supported Objects

You can configure a flow to wait for a record-base time for any custom object or the following standard objects.

- Account
- AccountContactRelation
- AccountRelationship
- ActionPlan
- ActiveScratchOrg
- ActivityMetric
- ActivityMetricRollup
- Address
- AgentWork
- AgentWorkSkill
- AiImageDetectedObject
- AiImageObject
- AiImageTrainingObject
- Asset
- AssetRelationship
- AssignedResource
- AssistantProgress
- BusinessAccount
- Campaign
- CampaignInfluence
- CampaignMember
- CareBarrier
- CareBarrierDeterminant
- CareBarrierType
- CareDeterminant
- CareDeterminantType
- CareDiagnosis
- CareInterventionType

- CarePreauth
- CarePreauthItem
- CareProgram
- CareProgramCampaign
- CareProgramEnrollee
- CareProgramTeamMember
- CareRequest
- CareRequestDrug
- CareRequestItem
- Case
- CaseComment
- Certification
- CertificationDef
- CertificationSectionDef
- CertificationStep
- CertificationStepDef
- ChannelProgram
- ChannelProgramLevel
- ChannelProgramMember
- ChatterActivity
- Claim
- CollaborationGroup
- CollaborationGroupMember
- ConsumptionRate
- ConsumptionSchedule
- Contact
- ContactEmail
- ContactPhone
- ContactPointConsent
- ContactPointTypeConsent
- ContactRequest
- ContactWeb
- Contract
- ContractLineItem
- CoverageBenefit
- CoverageBenefitItem
- CoverageLimit
- CoverageType
- CustomerAssetAuto
- CustomerAssetHome

- DandBCompany
- DataSharingCustomerLocal
- DataUsePurpose
- DigitalSignature
- DuplicateRecordItem
- DuplicateRecordSet
- EmailMessage
- EngagementProgramNode
- EngagementProgramVersion
- Entitlement
- EntitlementContact
- EntityMilestone
- EnvironmentHubMember
- EnvironmentHubMemberRel
- Event
- ExchangeUserMapping
- ExpressionFilter
- ExpressionFilterCriteria
- ExternalEventMapping
- FeedItem
- Goal
- GoalLink
- HealthCareDiagnosis
- HealthCareProcedure
- Idea
- IdentityDocument
- IdentityProvEventLog
- Image
- InStoreLocation
- Individual
- InsurancePolicy
- InsurancePolicyAsset
- InsurancePolicyBeneficiary
- InsurancePolicyCoverageLimit
- InsurancePolicyMember
- InsurancePolicyOwner
- InsuranceProfile
- KeyPerformanceIndicator
- Lead
- LinkedArticle

- LiveAgentSession
- LiveChatTranscript
- LiveChatTranscriptEvent
- LiveChatTranscriptSkill
- Location
- Macro
- MacroAction
- MacroInstruction
- MaintenanceAsset
- MaintenancePlan
- MemberPlan
- MessagingEndUser
- MessagingSession
- Metric
- MobileDeviceCommand
- NetworkMember
- OperatingHours
- Opportunity
- OpportunityLineItem
- OpportunityScore
- OpportunitySplit
- OpportunityTeamMember
- Order
- OrderDeliveryGroup
- OrderDeliveryGroupLine
- OrderDeliveryMethod
- OrderItem
- OrderItemSummary
- OrderPriceAdjustDistrLine
- OrderPriceAdjustmentLine
- OrderSummary
- OrgDeleteRequest
- OrgSnapshot
- Organization
- PartnerFundAllocation
- PartnerFundClaim
- PartnerFundRequest
- PartnerMarketingBudget
- PaymentAuthorizationReversal
- PendingServiceRouting

- PersonAccount
- PersonEducation
- PersonEmployment
- PersonLifeEvent
- PlanBenefit
- PlanBenefitItem
- Producer
- Product2
- ProductCategoryProduct
- ProductConsumed
- ProductCoverage
- ProductCoverageLimit
- ProductItem
- ProductItemTransaction
- ProductRequest
- ProductRequestLineItem
- ProductRequired
- ProductTransfer
- ProfileSkill
- ProfileSkillEndorsement
- ProfileSkillUser
- PurchaserPlan
- PurchaserPlanAssn
- Question
- QuickText
- Quote
- QuoteLineItem
- Reply
- RequestsForAccessSIQ
- ResourceAbsence
- ResourcePreference
- RetailLocationGroup
- RetailStore
- RetailStoreKpi
- RetailStoreVisitTemplate
- RetailVisitKpi
- RetailVisitTemplate
- RetailVisitTemplateWorkTask
- RetailVisitWorkTask
- RetailWorkTask

- RetailWorkTaskKpi
- ReturnOrder
- ReturnOrderLineItem
- SOSSession
- SOSSessionActivity
- SalesAgreement
- SalesAgreementProduct
- SalesAgreementProductSchedule
- ScoreIntelligence
- ScratchOrgInfo
- ServiceAppointment
- ServiceAppointmentCapacityUsage
- ServiceContract
- ServiceCrew
- ServiceCrewMember
- ServiceReport
- ServiceResource
- ServiceResourceCapacity
- ServiceResourceSkill
- ServiceTerritory
- ServiceTerritoryLocation
- ServiceTerritoryMember
- ServiceTerritoryWorkType
- SettingUsageMap
- Shipment
- SignupRequest
- Site
- SkillRequirement
- SocialPersona
- SocialPost
- Solution
- SsoUserMapping
- StreamActivityAccess
- StreamingChannel
- Survey
- SurveyInvitation
- SurveyPage
- SurveyQuestion
- SurveyQuestionChoice
- SurveyQuestionResponse

- SurveyQuestionScore
- SurveyResponse
- SurveySubject
- SurveyVersion
- Task
- TimeSheet
- TimeSheetEntry
- TimeSlot
- Topic
- TopicAssignment
- UsageEntitlementPeriod
- User
- UserLicense
- UserProvisioningRequest
- UserServicePresence
- Visit
- WebStore
- WebStoreNetwork
- WebStorePricebook
- WorkBadge
- WorkBadgeDefinition
- WorkCapacityLimit
- WorkCapacityUsage
- WorkCoaching
- WorkFeedback
- WorkFeedbackQuestion
- WorkFeedbackQuestionSet
- WorkFeedbackRequest
- WorkFeedbackTemplate
- WorkGoal
- WorkOrder
- WorkOrderLineItem
- WorkPerformanceCycle
- WorkReward
- WorkRewardFund
- WorkRewardFundType
- WorkThanks
- WorkType
- WorkTypeGroup
- WorkUpgradeAction

- WorkUpgradeCustomer
- WorkUpgradeUser
- *article*__kav

SEE ALSO:

Flow Resume Events

Flow Resume Event: Platform Event Message

Resume the flow interview when it receives a platform event message.

Make sure to familiarize yourself with Paused Flow Interview Considerations.

Filter Platform Event Messages

If you leave the condition requirements set to **No Conditions**, the flow interview resumes when it receives any platform event message, regardless of field values. The fields are defined in the platform event definition.

Store Output Values in Variables

When a platform event message resumes a flow, the message provides one output value. The output value includes the values for every field on the platform event message that resumed the flow. To use values from the message, store Platform Event Message in a record variable. Make sure that the record variable's object matches the platform event.

For example, to reference Expected Delivery Date from a Vendor Response platform event, store Platform Event Message in the `{!vendorResponse}` record variable. Then reference `{!vendorResponse.Expected_Delivery_Date__c}` to get the specific field value.

Note: To create a record variable to store values from the platform event message, you must have the Customize Application permission.

SEE ALSO:

Define and Manage Platform Events

Flow Resume Events

Sample Flows That Wait for Events

Configure a flow to wait for events in one of four ways.

Sample Flow That Pauses Until a Single Event Occurs

This flow waits for a single event. The base time for the event in this example, which uses a flow-based resume time, is the `{!$Flow.CurrentDateTime}` global variable.

Sample Flow That Pauses Until Only the First Resume Event Occurs

This flow waits for the first of multiple events to occur before proceeding. The base times for these events are field values, so this example's resume events use record-based time.

Sample Flow That Pauses Until Multiple Resume Events Occur

This flow waits for many resume events to occur, rather than just the first one. The base times for these events are field values, so this example's resume events use record-based time.

Sample Flow That Pauses Until a Platform Event Message is Received

You're designing a flow that places a supply order and waits for shipment confirmation from the vendor. Then it assigns an installation task the day after the supplies are expected to be delivered.

SEE ALSO:

Flow Elements: Wait

Sample Flow That Pauses Until a Single Event Occurs

This flow waits for a single event. The base time for the event in this example, which uses a flow-based resume time, is the `{!$Flow.CurrentDateTime}` global variable.

You're designing a flow that requests feedback from customers after a contract is activated, but you want to delay the email by a day.

EDITIONS

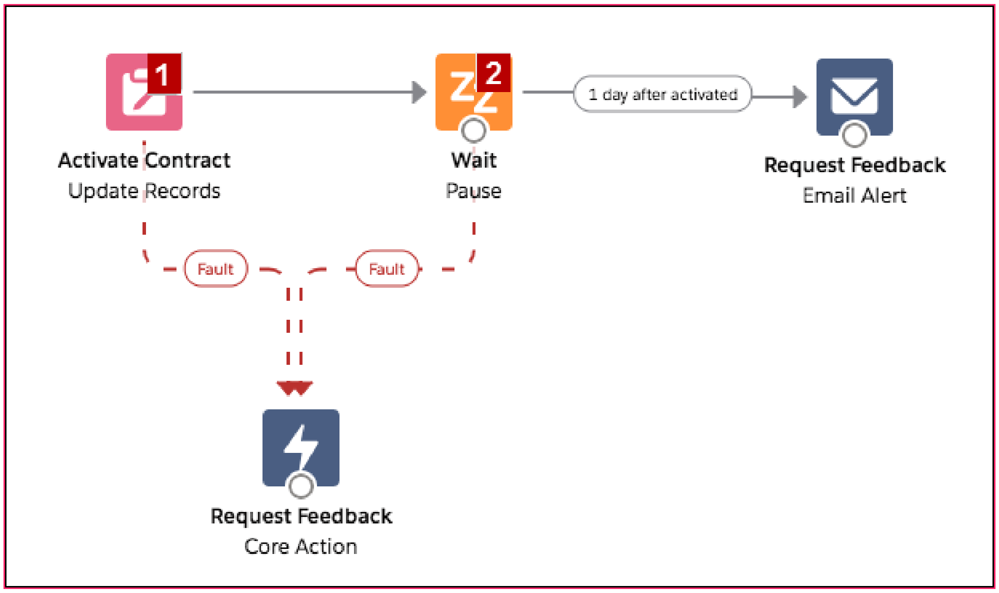
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

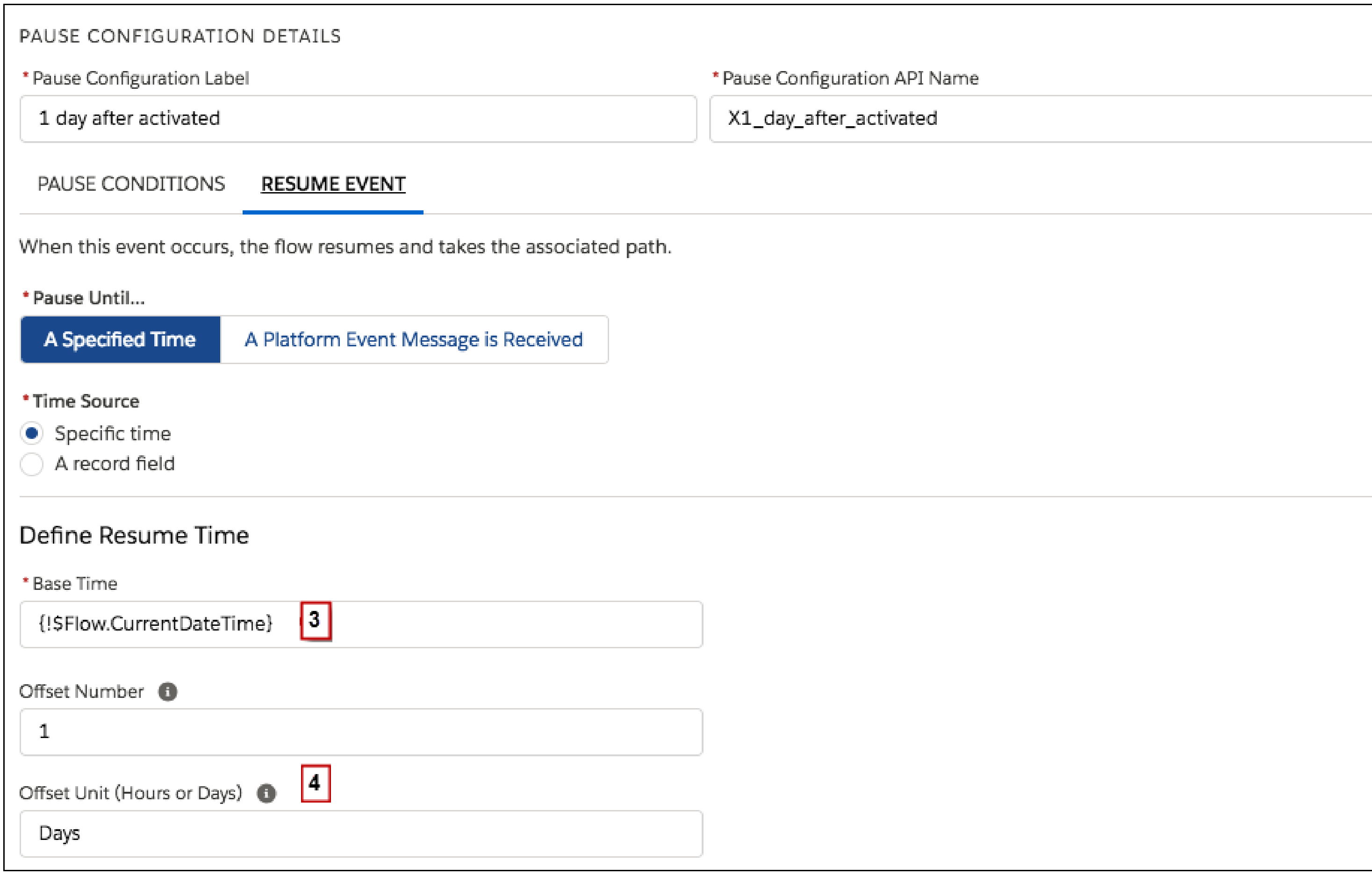
Example

This flow already contains the following populated variables. The flow activates a contract (1) and then pauses (2).

- `{!customerEmail}` contains the email address for the customer
- `{!creatorEmail}` contains the email address for the flow's creator



Within the Wait element, a single resume event is defined (1 day after activated). The flow sends the feedback request one day after the contract is activated, so configure a flow-based resume time. The base time is the `{!$Flow.CurrentDateTime}` global variable (3), and the offset is one day (4).



PAUSE CONFIGURATION DETAILS

* Pause Configuration Label: 1 day after activated

* Pause Configuration API Name: X1_day_after_activated

PAUSE CONDITIONS | RESUME EVENT

When this event occurs, the flow resumes and takes the associated path.

* Pause Until...

A Specified Time | A Platform Event Message is Received

* Time Source

● Specific time

○ A record field

Define Resume Time

* Base Time: {!$Flow.CurrentDateTime}

Offset Number: 1

Offset Unit (Hours or Days): Days

Because there’s only one wait configuration and you only want the feedback request to be sent one time, don’t set any wait conditions for this wait configuration. However, just in case something goes wrong, don’t forget to set a fault path. In this example, the fault path sends an email that contains the fault message to the user who created the flow.

SEE ALSO:

Flow Elements: Wait

Flow Resume Event: Specific Time

Sample Flow That Pauses Until Only the First Resume Event Occurs

This flow waits for the first of multiple events to occur before proceeding. The base times for these events are field values, so this example's resume events use record-based time.

You’re designing a flow that reminds account owners to follow up with their customers a week before either the account renews or the contract ends. The flow sends a reminder email for whichever date occurs first.

Example

This flow already contains these populated variables. Before the flow executes the Wait element, it looks up and stores the contract’s `ID`, its parent account’s `ID` and `OwnerId`, and the account owner’s `Email`.

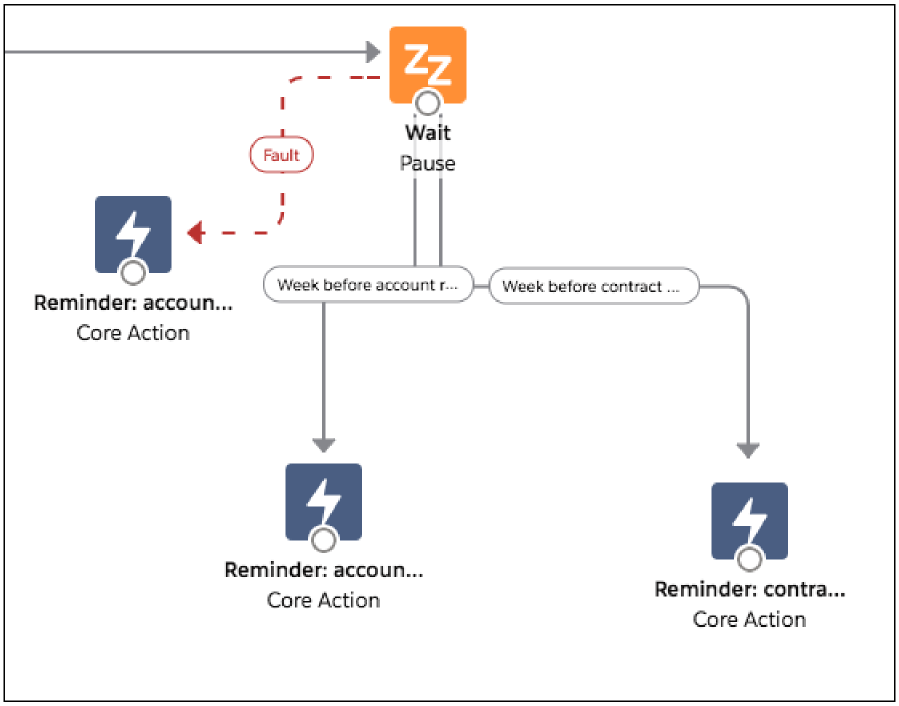
- `{!accountId}` contains the ID for the account

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

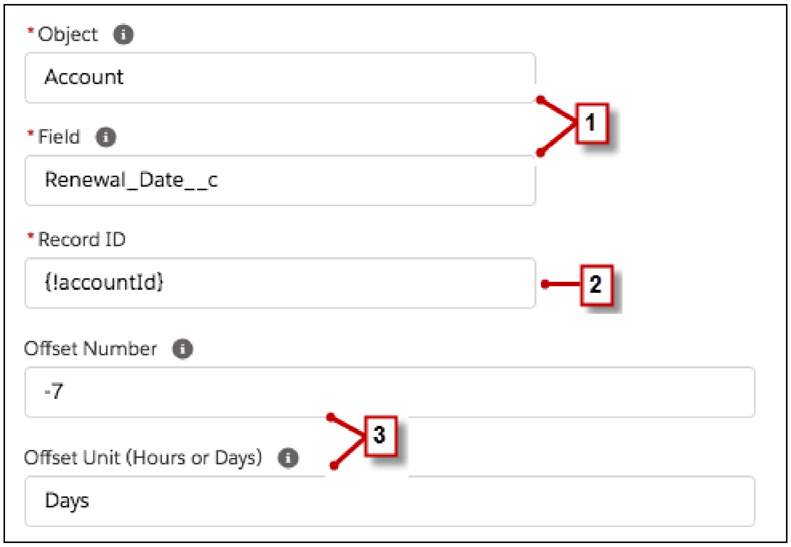
- `{!contractId}` contains the ID for the contract
- `{!accountOwner}` contains the ID for the account's owner
- `{!ownerEmail}` contains the account owner's email address



The Wait element defines two time-based resume events.

Tip: Every time-based resume event consists of a base time and an offset. For record-based time, the flow needs three pieces of information to determine the base time: the object, the date/time field, and the specific record. The offset for record-based time works the same as it does for flow-based time. The flow must know the unit (either *`Days`* or *`Hours`*) and the number of those units. For both of these events, the base time is offset by -7 days, because weeks isn't an acceptable offset unit.

The base time for the first event ("Week before account renews") is the value of `Account.Renewal_Date__c` (1) on the record whose ID is stored in `{!accountId}` (2). The offset is -7 days (3).



*Object

Account

*Field

Renewal_Date__c

*Record ID

{!accountId}

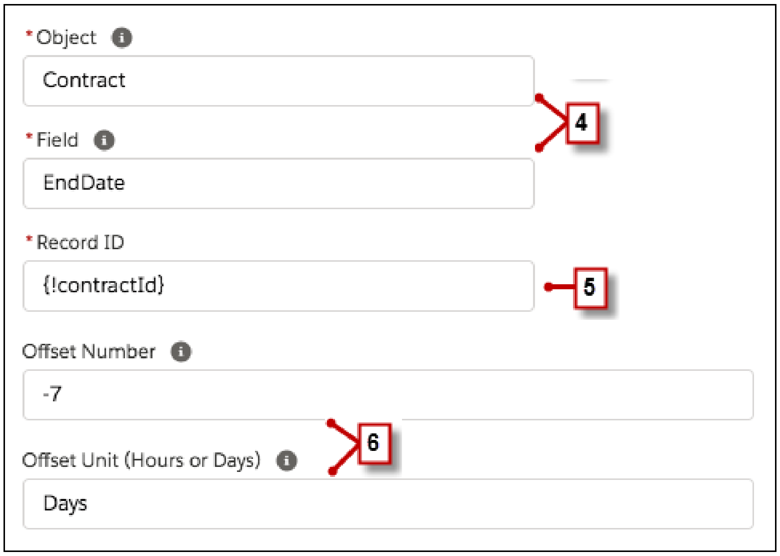
Offset Number

-7

Offset Unit (Hours or Days)

Days

The base time for the second event ("Week before contract expires") is the value of `Contract.EndDate` (4) on the record whose ID is stored in `{!contractId}` (5). The offset is -7 days (6).



*Object

Contract

*Field

EndDate

*Record ID

{!contractId}

Offset Number

-7

Offset Unit (Hours or Days)

Days

You only want to send one follow-up reminder and the flow always waits for both events, so neither of these events need wait conditions. However, just in case something goes wrong, set a fault path. In this example, the fault path sends an email that contains the fault message to the user who created the flow.

SEE ALSO:

Flow Elements: Wait

Flow Resume Event: Specific Time

Flow Wait Conditions

Sample Flow That Pauses Until Multiple Resume Events Occur

This flow waits for many resume events to occur, rather than just the first one. The base times for these events are field values, so this example's resume events use record-based time.

You're designing a flow that reminds contract owners to follow up with their customers before the contract ends. Rather than sending just one reminder, however, the flow sends them regularly. This example shows how to use one Wait element to send a reminder two weeks before and then again one week before the contract ends. You could easily extend this flow to send reminders at more intervals, such as three days and one day before the contract ends.

EDITIONS

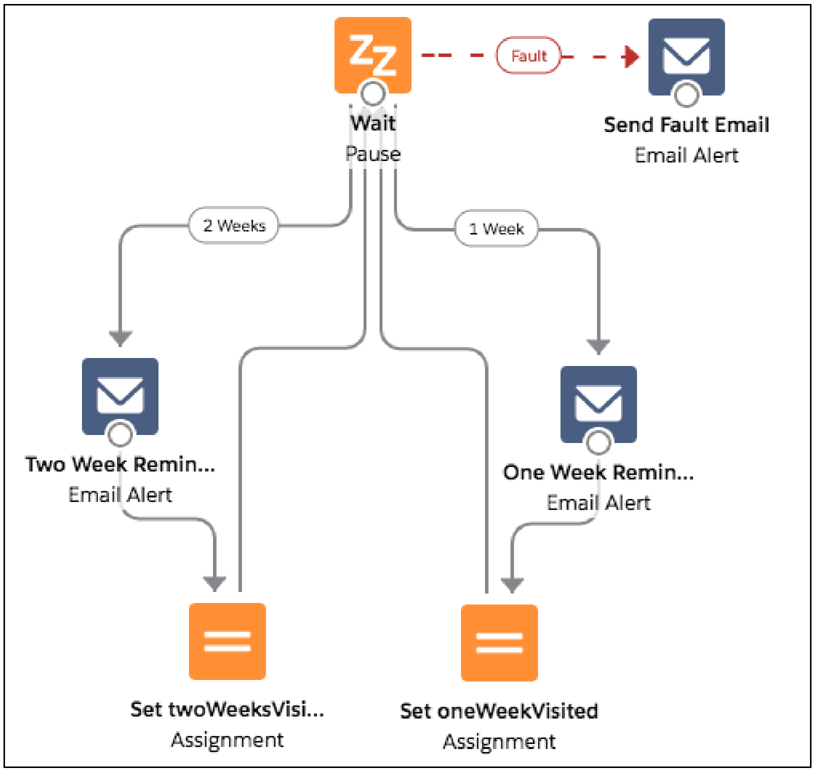
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example

This flow already contains these populated variables. Before the flow executes the Wait element, it looks up and stores the contract's `EndDate` and `OwnerId`.

- `{!contract}` is a record variable that contains the contract's `EndDate` and `OwnerId`
- `{!contractId}` is a text variable that contains the contract's `Id`
- `{!oneWeekVisited}` is a Boolean variable whose default value is *`{!$GlobalConstant.False}`*
- `{!twoWeeksVisited}`is a Boolean variable whose default value is *`{!$GlobalConstant.False}`*

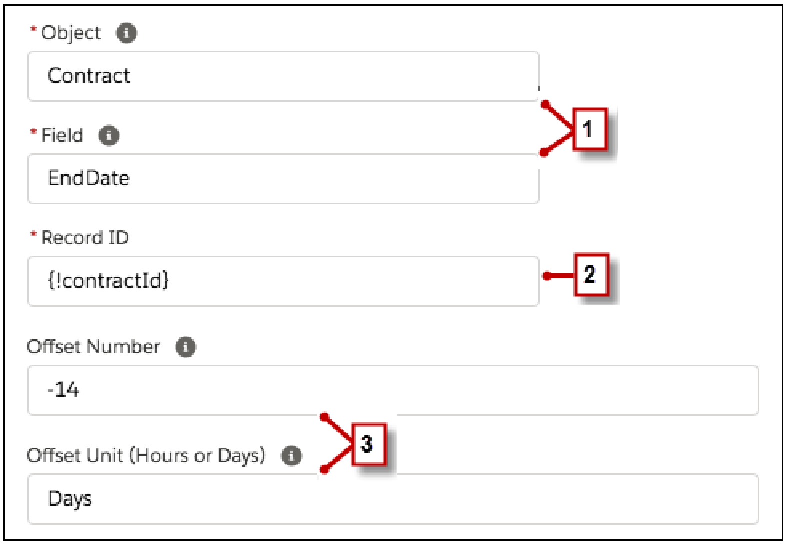


Because the flow sends the reminder emails both two weeks and a week before the contract's end date, the Wait element defines two time-based resume events that use record-based time.

Tip: Every time-based resume event consists of a base time and an offset. For record-based time, the flow needs three pieces of information to determine the base time: the object, the date/time field, and the specific record. The offset for record-based time works the same as it does for flow-based time. The flow must know the unit (either *`Days`* or *`Hours`*) and the number of those units. To wait for a number of days or hours before the record field, set `Offset Number` to a negative integer.

For both of these events, the offset is declared in *`Days`*, because weeks isn't an acceptable offset unit.

The base time for the first event (“2 Weeks”) is the value of `Contract.EndDate` (1) on the record whose ID is stored in `{!contractId}` (2). The offset is -14 days (3) to represent two weeks.



*Object: Contract

*Field: EndDate

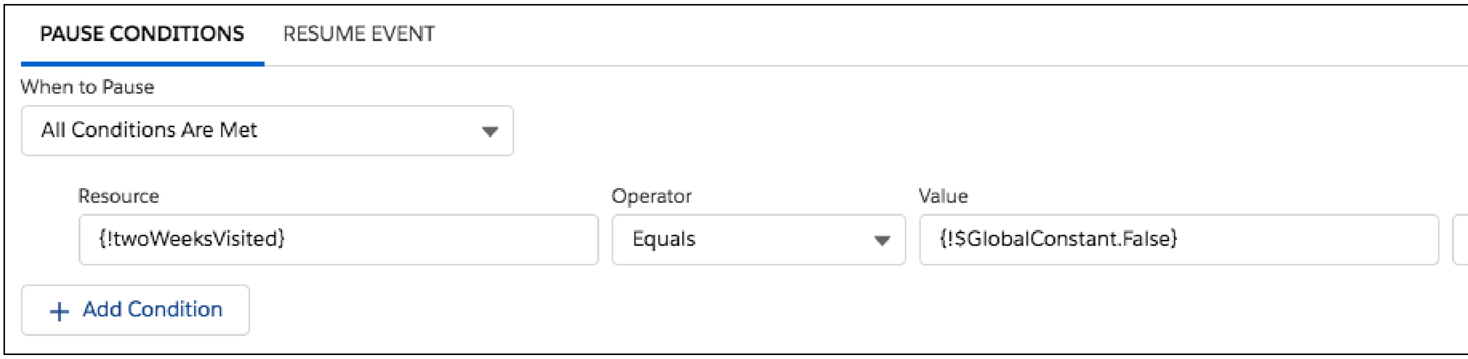
*Record ID: {!contractId}

Offset Number: -14

Offset Unit (Hours or Days): Days

You want to use the same Wait element for every reminder, so after a flow interview sends one email reminder, it returns to the Wait element. But first, to ensure that the interview doesn’t send the same email again and again, use *wait conditions*. When an interview executes a Wait element, it first checks the conditions for each wait configuration to determine whether to wait for those events. If a wait configuration has conditions set and those conditions aren’t met, the interview doesn’t wait for the associated resume event.

For the first resume event, the interview checks whether the Boolean variable `{!twoWeekVisited}` is set to false. The variable’s default value is set to *`{!$GlobalConstant.False}`*, so the flow waits for the event until the variable’s value is changed.

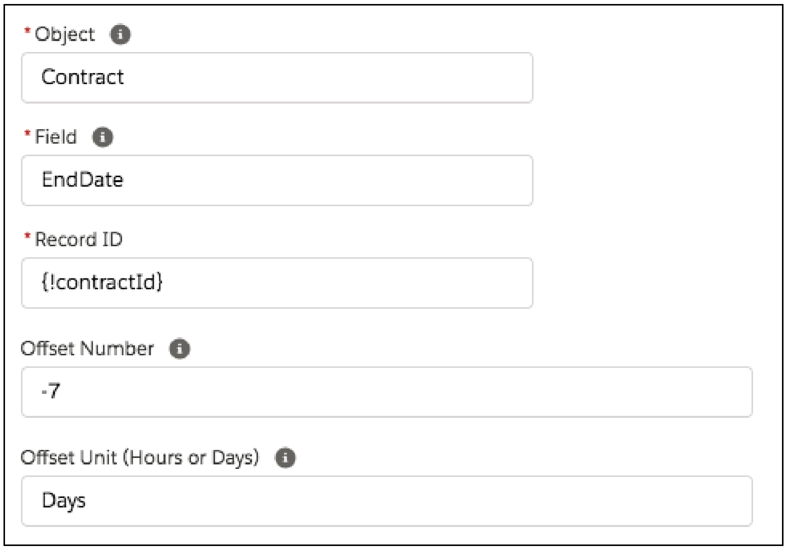


When to Pause: All Conditions Are Met

Resource	Operator	Value
{!twoWeeksVisited}	Equals	{!$GlobalConstant.False}

Indicate what the flow does when the “2 Weeks” event occurs by connecting the Wait element to other elements. Then, before you return the flow path to the Wait element, change the value of `{!twoWeeksVisited}` to *`{!$GlobalConstant.True}`*. You can do so with an Assignment element. If the value for `{!twoWeeksVisited}` isn’t false when the Wait element is executed, the flow doesn’t wait for the “2 Weeks” event to occur. Essentially, the interview checks whether the first resume event has occurred yet, since the variable is changed to true only in the associated wait configuration’s path. If that resume event has occurred (and the variable isn’t set to false), the interview knows not to wait for that event.

The second event (“1 Week”) has the same base time as the first event; the offset is -7 days to represent a week.



* Object
Contract

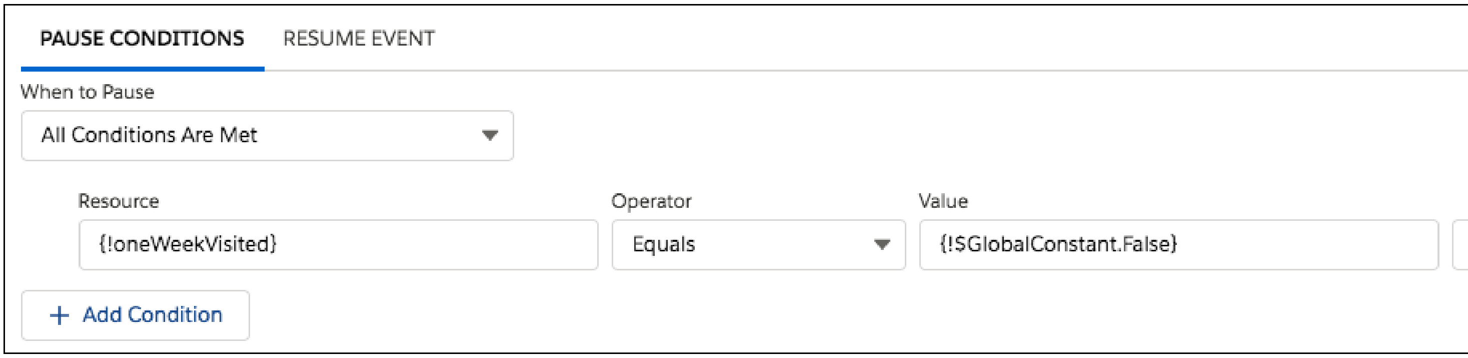
* Field
EndDate

* Record ID
{!contractId}

Offset Number
-7

Offset Unit (Hours or Days)
Days

For the second event, the flow checks whether the Boolean variable `{!oneWeekVisited}` is set to false. If it isn't, the flow doesn't wait for this event.



When to Pause
All Conditions Are Met

Resource	Operator	Value
{!oneWeekVisited}	Equals	{!$GlobalConstant.False}

Like with the first wait configuration, use an Assignment element to change the value of `{!oneWeekVisited}` to *`{!$GlobalConstant.True}`* before the flow path returns to the Wait element. As long as `{!oneWeekVisited}` isn't false, the flow doesn't wait for the "1 Weeks" event to occur.

Tip: When a flow executes a Wait element and all the wait configurations have conditions that aren't met, the flow executes the *default path*. Because this flow is finished after it sends the final reminder, don't connect the default path to another element.

Just in case something goes wrong, set a fault path. In this example, the fault path sends an email that contains the fault message to the user who created the flow.

Sample Flow That Pauses Until a Platform Event Message is Received

EDITIONS

Available in both Salesforce Classic and Lightning Experience

Available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions

You're designing a flow that places a supply order and waits for shipment confirmation from the vendor. Then it assigns an installation task the day after the supplies are expected to be delivered.

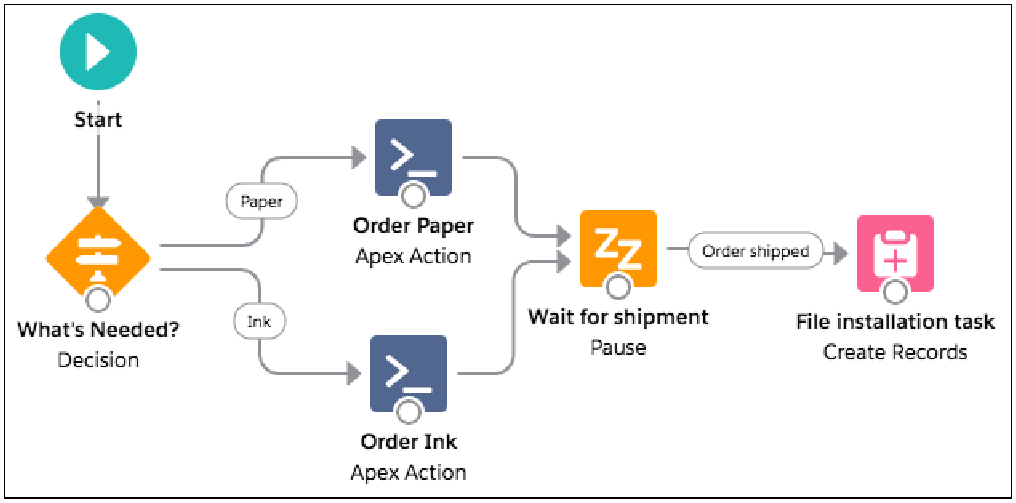
The vendor that you buy supplies from has set up a platform event for you to subscribe to. This platform event, called Vendor Response, includes the order number, order status, and expected delivery date.

Note: This flow is part of a larger example. It gets launched by a process that starts when a Printer Status platform event message is received. For details about the process, see Sample Process: Printer Management.

The Order Printer Supplies flow starts when the Printer Management process launches it. The process populates the following variables in the flow.

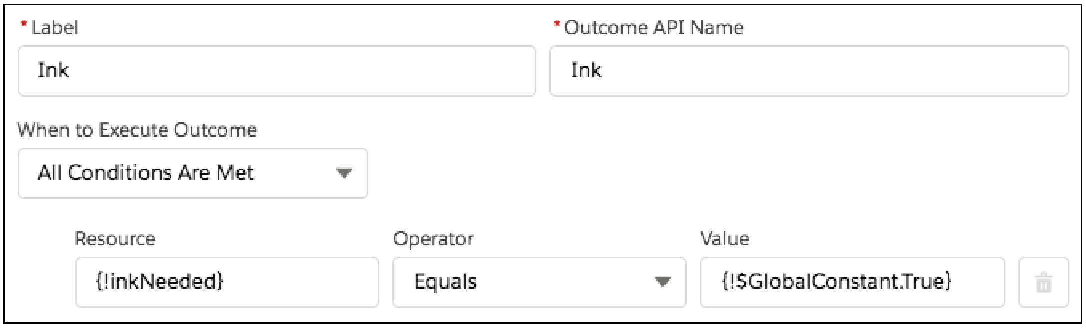
- `{!assetId}`—The asset's ID
- `{!assetOwner}`—The asset's owner
- `{!inkManufacturer}`—The manufacturer of the printer's ink
- `{!inkNeeded}`—Whether the printer needs more ink
- `{!inkType}`—Specific type of ink that the printer uses
- `{!paperNeeded}`—Whether the printer needs more paper
- `{!paperSize}`—Paper size that the printer uses
- `{!serialNumber}`—The asset's serial number

First, the flow determines whether to order ink or paper. Based on the decision, it submits an order of ink or paper with the vendor by using an Apex action. Then it pauses until the vendor sends a platform event message that says the order has been shipped. When Salesforce receives the specified event message, the flow resumes and creates a task for the asset's owner to install the new supplies.



Decision Element

The decision includes two outcomes: Ink and Paper. The Ink outcome is true if the variable `{!inkNeeded}` is true. The Paper outcome is true if the variable `{!paperNeeded}` is true.

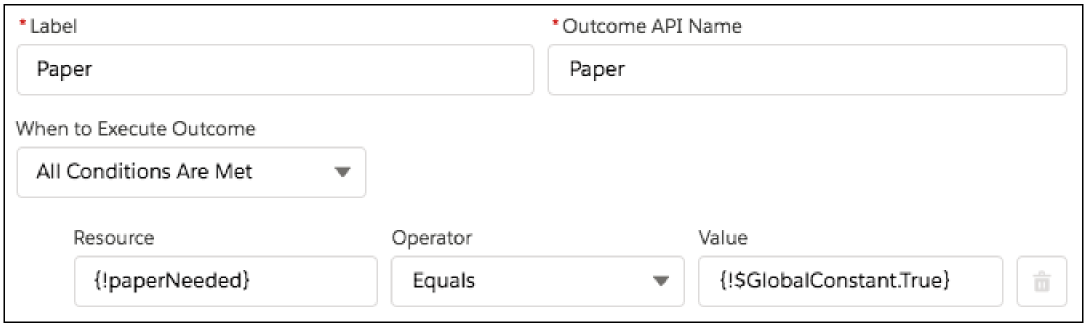


*Label: Ink

*Outcome API Name: Ink

When to Execute Outcome: All Conditions Are Met

Resource	Operator	Value
{!inkNeeded}	Equals	{!$GlobalConstant.True}



*Label: Paper

*Outcome API Name: Paper

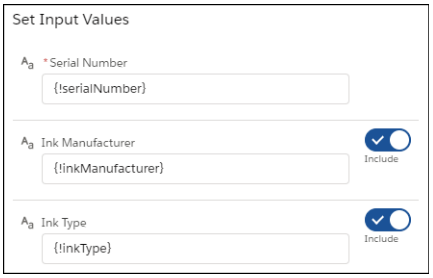
When to Execute Outcome: All Conditions Are Met

Resource	Operator	Value
{!paperNeeded}	Equals	{!$GlobalConstant.True}

Apex Action Elements

The flow includes two Apex actions that submit a supply order with a vendor but provide different information to it based on whether the flow executed the Ink outcome or Paper outcome. All the variables used for input values (like `{!serialNumber}` and `{!paperSize}`) are set when a process launches the flow.

The first Apex action provides information about which ink to order.



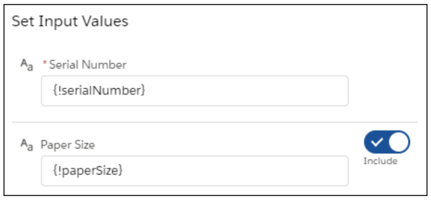
Set Input Values

*Serial Number: {!serialNumber}

Ink Manufacturer: {!inkManufacturer} — Include

Ink Type: {!inkType} — Include

The second Apex action provides information about which paper to order.

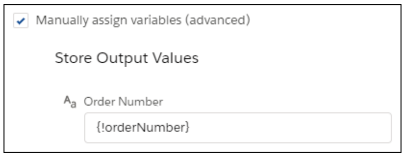


Set Input Values

*Serial Number: {!serialNumber}

Paper Size: {!paperSize} — Include

In both Apex actions, the action returns an order number. The flow stores that value in the `{!orderNumber}` variable to reference in the Wait element.



☑ Manually assign variables (advanced)

Store Output Values

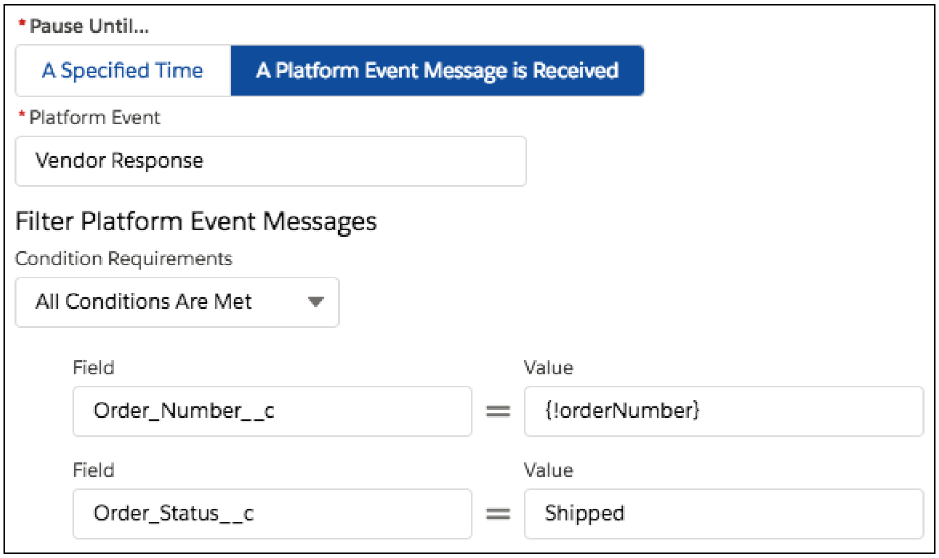
Order Number

{!orderNumber}

Wait Element

After the Apex action submits the supply order, the flow waits for confirmation that the order has been shipped. That confirmation is received through the Vendor Response platform event.

The flow pauses until Salesforce receives a Vendor Request event message with specific values. The order number must be the same as the order number that the Apex action provided. And the order status must be Shipped.



* Pause Until...

A Specified Time | A Platform Event Message is Received

* Platform Event

Vendor Response

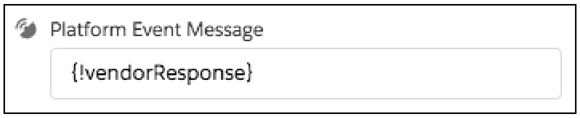
Filter Platform Event Messages

Condition Requirements

All Conditions Are Met

Field		Value
Order_Number__c	=	{!orderNumber}
Order_Status__c	=	Shipped

When the correct event message is received and the flow resumes, the flow stores the event message's data in a record variable. That way, you can reference the expected delivery date to calculate when the supplies are scheduled to be installed.

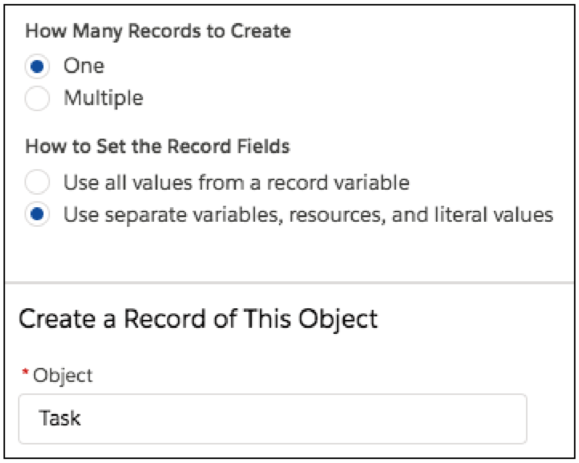


Platform Event Message

{!vendorResponse}

Create Records Element

When the flow resumes, it creates a task for the asset owner to install the new supplies.



How Many Records to Create

● One

○ Multiple

How to Set the Record Fields

○ Use all values from a record variable

● Use separate variables, resources, and literal values

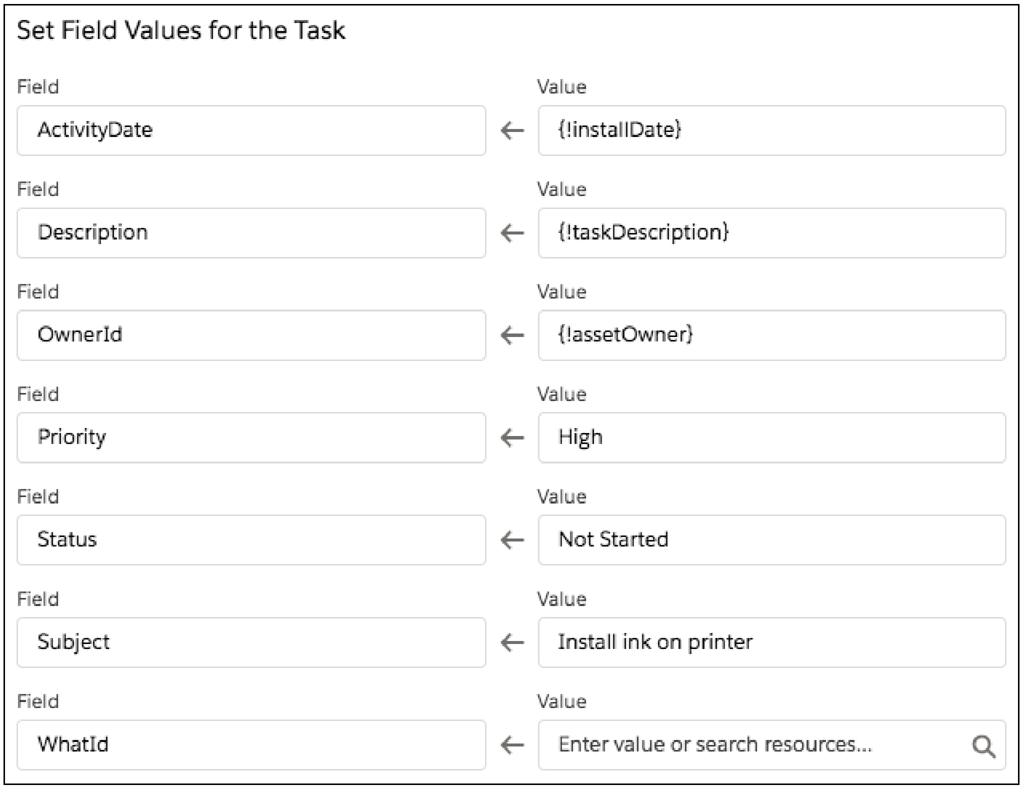
Create a Record of This Object

* Object

Task

For the task's field values, the flow uses these resources.

- `{!installDate}`—A formula that calculates the day after the event's expected delivery date.
- `{!taskDescription}`—A text template that gives more details about the installation.
- `{!assetOwner}`—Provided by the process that launches the flow
- `{!assetId}`—Provided by the process that launches the flow



Set Field Values for the Task

Field		Value
ActivityDate	←	{!installDate}
Description	←	{!taskDescription}
OwnerId	←	{!assetOwner}
Priority	←	High
Status	←	Not Started
Subject	←	Install ink on printer
WhatId	←	Enter value or search resources...

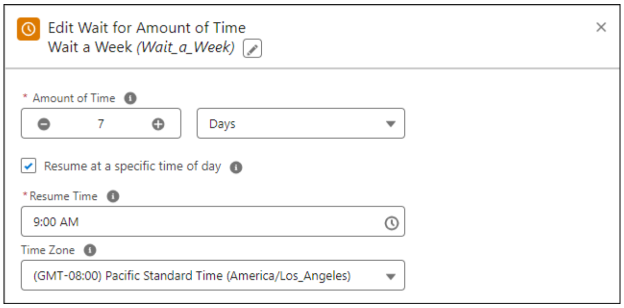
Flow Element: Wait for Amount of Time

Resume a flow interview after a specific amount of time.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Edit Wait for Amount of Time
Wait a Week *(Wait_a_Week)*

* Amount of Time
7
Days

☑ Resume at a specific time of day

* Resume Time
9:00 AM

Time Zone
(GMT-08:00) Pacific Standard Time (America/Los_Angeles)

Field	Description
`Amount of Time`	How long to wait until resuming the flow.
`Resume at a specific time of day`	Specifies whether to wait to resume a flow until a specific time of day. When enabled, if the specified amount of time expires after the resume time, the flow doesn't resume until the resume time on the following day.
`Resume Time`	When to resume the flow.
`Time Zone`	The time zone used to resume the flow.

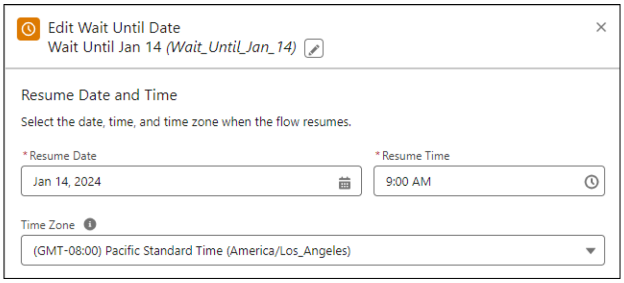
Flow Element: Wait Until Date

Resume a flow interview after a specific date.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Edit Wait Until Date
Wait Until Jan 14 *(Wait_Until_Jan_14)*

Resume Date and Time

Select the date, time, and time zone when the flow resumes.

* Resume Date: Jan 14, 2024

* Resume Time: 9:00 AM

Time Zone: (GMT-08:00) Pacific Standard Time (America/Los_Angeles)

Field	Description
`Resume Date`	The date to resume the flow.
`Resume Time`	The time to resume the flow.
`Time Zone`	The time zone used to resume the flow.

Flow Element: Recommendation Assignment

Generate Einstein Next Best Action recommendations by combining data from records in the recommendation object, records in other objects, values in collections, and manually entered values.

The Recommendation Assignment element is similar to the Assignment element: both set values in variables. However, there are important differences:

- Use Recommendation Assignment to output a new recommendation collection variable. In contrast, use Assignment to add or change values in an existing variable.
- Recommendation Assignment can't update values in existing variables.
- Recommendation Assignment can set a field's value across all recommendations in the output collection. For example, if you set the AcceptanceLabel to `Accept`, the AcceptanceLabel for all records in the output collection is set to Accept.
- Recommendation Assignment can create recommendations from another object's records. For example, use leads in the source collection to create recommendations that have the same names as those leads.
- Recommendation Assignment is available only in Recommendation Strategy flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Source Collection

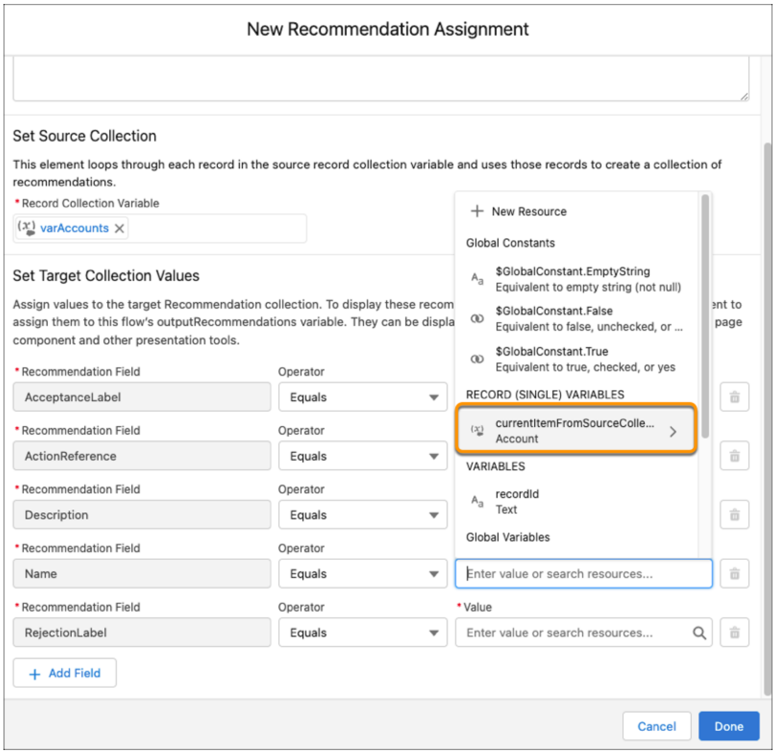
Select a record collection variable with the data that you want to use to create recommendations. Recommendation Assignment creates one recommendation for every record in the source collection.

If you select a record collection variable that contains recommendations, you can choose the fields that set values in the target collection. If you select a record collection variable that contains any other object's records, the element requires certain recommendation fields.

Set Target Collection Values

Assign values to recommendation fields in the output recommendation collection. Values defined in this section are set in every record in the output collection. If there are values in the source collection variable that aren't set here, those values are passed to the output recommendation collection.

To use the source collection records' values from a specific field, select `currentItemFromSourceCollection` in the Value column, then select the desired field.



New Recommendation Assignment

Set Source Collection

This element loops through each record in the source record collection variable and uses those records to create a collection of recommendations.

* Record Collection Variable: varAccounts

Set Target Collection Values

Assign values to the target Recommendation collection. To display these recom… ent to assign them to this flow's outputRecommendations variable. They can be displa… page component and other presentation tools.

* Recommendation Field	Operator	* Value
AcceptanceLabel	Equals	
ActionReference	Equals	
Description	Equals	
Name	Equals	Enter value or search resources...
RejectionLabel	Equals	Enter value or search resources...

New Resource
Global Constants
$GlobalConstant.EmptyString
Equivalent to empty string (not null)
$GlobalConstant.False
Equivalent to false, unchecked, or ...
$GlobalConstant.True
Equivalent to true, checked, or yes
RECORD (SINGLE) VARIABLES
currentItemFromSourceColle...
Account
VARIABLES
recordId
Text
Global Variables
Add Field
Cancel
Done

Each value is modified by the operator and value combination.

Column Header	Description
Recommendation Field	The recommendation field whose value you want to change. Select an existing field.
Operator	Select `Equals` to set the value in a field. Select `Add` to add the text that you enter or select for `Value` to the end of `Recommendation Field`.
Value	`Recommendation Field` and `Value` in the same row must have compatible data types. Options:

Einstein Next Best Action always requires certain recommendation fields to display a recommendation. However, the Recommendation Assignment element only requires these fields if you select a source collection that contains non-recommendation records. If you select a source collection with recommendations, Recommendation Assignment doesn't require the fields because it's possible that your source collection already has values in those fields. However, the fields are still required to display the recommendation.

- AcceptanceLabel
- ActionReference
- Description
- Name
- RejectionLabel

Usage

Recommendation Assignment outputs a collection with the assembled recommendation records and doesn't change the contents of the source collection. The output collection is null until its corresponding Recommendation Assignment element runs.

Note: A Recommendation Assignment's output collection is named after its API name. For example, if a Recommendation Assignment element is named `CreateRecs`, its output collection is called `Recommendations from CreateRecs`.

However, recommendations display to users only if they're in the outputRecommendations variable. To add recommendations to the outputRecommendations collection, use an Assignment element.

SEE ALSO:

Get Started with Einstein Next Best Action

Create Recommendations

Recommendation Fields

Einstein Next Best Actions Considerations

Flow Element: Screen

Collect information from or display information to a user who runs the flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Screen Properties

When you don't have a screen component or record field selected, the properties pane shows the entire screen.

- Configure Frame—Control whether the header and footer are displayed for this screen. These options are supported only in Lightning runtime. If you hide the footer, use a custom screen component to let the user navigate between screens.
- Control Navigation—Deselect the navigation options that you want to disable for this screen. By default, navigation options appear as buttons in the screen footer. The Next action is available when an element is in the flow after the screen. The Finish action is available when the screen is the last element in the flow. The Previous action is available when a screen is before this screen. The Pause action is available when **Let users pause flows** is enabled in your org's Process Automation settings.

  For example, a flow prompts a user to enter information and uses that information to get a Contact record. If no matching contact is found, the flow displays a screen to tell the user to go back and try again.

  Tip:

  - If you hide the footer but want to let the user navigate between screens, expose the actions with Lightning components.
  - If a data element precedes the screen element, such as Update Records, or an action, such as Post to Chatter, deselect **Previous**.
  - To force the flow user to go back, such as to correct an earlier input, deselect **Next or Finish**.

  If the Pause action is enabled:

  - Use **Pause Confirmation Message** to tell the user where to resume the flow. For the components that list a user's paused flows, see Make It Easy for Users to Find Their Paused Flow Interviews.
  - Customize the flow's interview label.
  - In your org's Process Automation settings, enable **Let users pause flows**.

- Provide Help—Give your users more context for this screen. The text you enter is available in an info bubble in the screen's header. If you hide the header but want to expose the help text, use a custom screen component.

Screen Components

The Components tab contains all the standard input components, standard display components, and custom components that are available for the screen. Click and drag a component to add it to the screen.

Tip: If you have many custom components, enter text in the search field to find the one you need. You can access third-party custom components on AppExchange using the button at the bottom of the pane.

Record Fields

Build screen flows faster by adding your existing Salesforce record fields directly from the Fields tab. When you add a record field to a flow screen, the field's name, data type, help text, requiredness, and existing values are automatically configured for you. To add a record field, select or create a record variable, then click and drag a field to add it to the screen.

Make Flow Screens Dynamic with Conditional Visibility

You can control when screen components appear with conditional visibility.

Validate User Input on Flow Screens

Validate user input on a flow screen at run time by using a Boolean formula expression, and provide a custom error message to guide the user.

Adding Record Fields to Flow Screens

Build flow screens faster by adding fields directly from your Salesforce objects. When you add a record field to a flow screen, its name, data type, help text, requiredness, and, in certain cases, existing values are automatically configured. Record fields use a record variable to determine which fields can be placed on a flow screen and their configuration.

Flow Screen Actions

Screen actions are used in Screen elements to retrieve or process data by triggering an autolaunched flow. The output from the autolaunched flow is then made available to all components on that same screen, reducing the number of screens needed in a flow. Screen actions are triggered by an Action Button (beta) component.

SEE ALSO:

Flow Elements

Build Rich Screens with Custom Screen Components

Move and Connect Elements to Change a Flow Route

Standard Flow Screen Components

Make Flow Screens Dynamic with Conditional Visibility

You can control when screen components appear with conditional visibility.

1. For the screen component, expand the Set Component Visibility section.
2. Define the conditions for when the component is visible.
3. Define the filter logic if you entered multiple conditions.

SEE ALSO:

Flow Conditional Visibility Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Validate User Input on Flow Screens

Validate user input on a flow screen at run time by using a Boolean formula expression, and provide a custom error message to guide the user.

The formula expression used to validate user input must return a Boolean value (`true` or `false`). If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the custom error message appears below the component.

Note: If the user leaves a required field blank, the flow shows the default error message, not your custom error message.

Tip: In regular expressions, use a double backslash to escape any characters that start with a slash. For example, *\d* becomes *\\d*.

To add input validation to a flow screen component:

1. In Flow Builder, on a screen, add a screen input component, and then expand the Validate Input section.
2. Customize the error message that appears if the user enters an invalid value.

   To format the error message, use HTML tags.
3. Define the values allowed for the component by entering a Boolean formula. In the formula, reference the correct output for the component.
   a. If the component has one output, reference the component itself. For example, for a Text component labeled Cancellation Reason, reference `{!Cancellation_Reason}`.
   b. If the component has multiple outputs, reference the specific output of the component. For example, for an Email component labeled Contact Email, reference `{!Contact_Email.value}`.

   Tip: For a component to reference itself in the Validate Input section, you must click away from the component configuration pane after you add it to the screen to save its state before you attempt to reference it.

Use these example formulas as a guide.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

- This formula validates the format of an email address in a Text component.

  ```
  REGEX({!Email_Address},"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}")
  ```

- This formula validates the format of a ZIP code in a Text component.

  ```
  REGEX({!Zipcode},"\\d{5}(-\\d{4})?")
  ```

- This formula validates that a user uploaded at least one file in a File Upload component.

  ```
  NOT({!fileUpload.contentDocIds} = "[]")
  ```

- This formula validates that a specific account is selected from a list of accounts in a Lookup component

  ```
  CONTAINS({!myLookup.recordIds},{!getSpecificAccount.Name})
  ```

Note: Validating record collections or Apex-defined type collections isn’t supported.

Note: At run time, if a user leaves a component blank, the component's value isn't validated for these components: Checkbox, Checkbox Group, Choice Lookup, Currency, Date, Date & Time, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, Text.

SEE ALSO:

Salesforce Help: Formula Operators and Functions by Context

Flow Formula Considerations

Adding Record Fields to Flow Screens

Build flow screens faster by adding fields directly from your Salesforce objects. When you add a record field to a flow screen, its name, data type, help text, requiredness, and, in certain cases, existing values are automatically configured. Record fields use a record variable to determine which fields can be placed on a flow screen and their configuration.

EDITIONS

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Place Record Fields Directly on Flow Screens

To add record fields to a screen flow, follow these steps.

Record Fields on Flow Screens Considerations

Before you add fields from your Salesforce objects directly to your flow screens, consider record field behaviors.

Place Record Fields Directly on Flow Screens

To add record fields to a screen flow, follow these steps.

1. Create or edit a screen element.
2. On the Fields tab, select a record variable or create a variable with the Record data type and the object that contains the desired field.

   Alternatively, select a record variable automatically created by a Get Records element or a Loop element.
3. From the list of fields that appear, drag a field to the screen canvas.
4. To use a record field’s existing value, use a Get Records element for the record variable.

Note: If a field contains a value in the record variable, that value is set on the screen at run time as a default value.

Record Fields on Flow Screens Considerations

Before you add fields from your Salesforce objects directly to your flow screens, consider record field behaviors.

Account Ticker Field

Placing the standard Ticker Symbol field on Account records in your screen prevents your flow from saving.

Creating and Updating Records

Just like other fields on a screen, record fields don't save data automatically. To save the record field data, use the associated record variable in a Create Records or Update Records element.

Default Values

Record fields don't support the default values of their source fields. If a record field's source field has a default value, the record field is blank.

If the field contains a value in the record variable, that value is set in the screen at runtime as a default value.

Event and Task Record Fields

Some event and task record fields aren't supported. The supported field types are Date, Date/Time, Checkbox, Number, Text, Text Area, and Text Area (Long).

At run time, event and task record fields behave differently on a screen than other types of record fields.

- Checkbox field labels aren't shown in line with the checkbox. They're shown above or below the checkbox.
- When edited, the record field doesn't have a yellow background, and the undo button isn't shown.

Lookup

- Creating a record from Lookup isn't supported in these environments: Flows launched from URLs (such as List Buttons), Flow Debugger, Lightning Out, Digital Experiences (LWR), and Embedded Service Flows.
- Lookup fields aren't compatible with mobile devices.
- To view or change the value of a Master-Detail relationship for existing records, the Allow reparenting setting must be enabled for that field.
- Lookup fields with filters applied don't immediately display errors from the flow runtime when a user running the flow attempts to create a record. Errors display only when creating or updating the records that reference the newly created record.
- UI API must support the object where the Lookup resides.

Lookup Filters

If a lookup filter relies on the field values of the current record in the flow, fields used in the lookup filter must be added on the same screen. If you must limit the available records in your Lookup based on the actions taken by a user in a flow, use the Choice Lookup component.

Multi-Select Picklists

Using the Add Item operator in assignments allows for duplicate values in multi-select picklist fields.

Name Field

If the value of a Name subfield is changed at runtime, the new value is set, and the subfield shows the new value. But the full Name record field still shows the original value.

Permissions

System context doesn't apply to record fields. If a user doesn't have access to a record field, they can't see it in a flow.

If no ID is set for the record variable, then the running user needs both `Read` and `Create` access to change any record field values.

If an ID is set, the running user needs both `Read` and `Edit` access to change any record field values.

In the record field details, the values in Update Compatible and Create Compatible reflect the properties of the field and your permissions. The values don't reflect the permissions of the users running the flow.

Screen flows don't display record fields that are set to read only at the object level.

Person Accounts

If PersonAccount is enabled and Account record fields have been added to a screen:

- When the record type is a Person Account, only Person* fields and common fields display.
- If the record type is a business Account, only Business account fields and common fields are displayed.

All account record fields are displayed and the record type is ignored.

The `Account.Name` displays as a `PersonName` field if the record type is a Person Account and as a Text field if the record type is a business account. `Account.Name` displays as a Text field.

Picklist

Record fields support dependent picklists only if the dependent field is on the same screen as its controlling field.

If a record field has a record type ID set when entering the screen, its values are filtered by record type. If the record type ID isn't set, all values are displayed.

Referencing Record Fields

You can't reference record fields in other areas of your flow, such as formulas, decisions, and conditional visibility. Instead, reference the record variable that you used to create the record fields.

Runtime Environments

Record fields aren't supported in screen flows on Experience Cloud sites that use Lightning Web Runtime (LWR).

Supported Field Data Types

Record fields support these field data types: Address, Checkbox, Date, Date and Time, Email, Lookup, Name, Number, Phone, Picklist, Text, Text Area, and Text Area (Long).

Other Considerations

- Record fields use the labels returned by the User Interface (UI) API, which can differ from the labels that appear on record pages and in Object Manager. For record fields of the Name data type, the label appears as Full Name instead of Name for most objects.
- Record fields aren't supported in the Repeater screen component.
- Validating user input isn't supported.

Flow Screen Actions

Screen actions are used in Screen elements to retrieve or process data by triggering an autolaunched flow. The output from the autolaunched flow is then made available to all components on that same screen, reducing the number of screens needed in a flow. Screen actions are triggered by an Action Button (beta) component.

To use a screen action, add an Action Button component to a Screen element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Flow Screen Input Component: Action Button

Flow Element: Start

Connect the Start element to the flow element that you want to execute first at run time. In an autolaunched flow, you can open the Start element to add a trigger that launches the flow. Without a trigger, you must set up other things to invoke the autolaunched flow, such as custom buttons, processes, Apex classes, or Einstein Bots.

Note: In event-triggered flows, you can set the flow to run as the default workflow user. If the default workflow user gets unset, the flow runs as the automated process user.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Schedule Triggers for Flows That Run for Batches of Records

Record Triggers for Flows That Make Before-Save Updates

Creating Flow Formulas with Flow Formula Builder

Flow Element: Subflow

Launch another active flow that's available in your org. A flow launched by another flow is called the *referenced flow*.

Add a Subflow element to your flow and then, using the label or API name of the flow, search for a flow to configure. The flow you select is the referenced flow. To open the referenced flow in a new window, click the action menu. You can't reference a screen or a template-triggered prompt flow from an autolaunched flow. A template-triggered prompt flow can only be referenced from another template-triggered prompt flow. For each flow, the list shows the label and API name of the active version. If a flow doesn't have an active version, the list displays the label and API name of the latest version. You can't call flows that contain wait elements.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Tip: Create smaller flows that perform common tasks. For example, build utility flows to capture address and credit card information or to authorize a credit card purchase amount. Then call those flows as needed from multiple product-ordering flows.

Note: Only flow admins can run inactive flows. For other users, the flow fails at run time if a Subflow element calls a flow with no active version.

Set Input Values

To set the input variables for the referenced flow, use values from earlier in the flow. In the Set Input Values tab, the Subflow elements displays only input variables from the active version and the latest version of the referenced flow.

The values are assigned at run time when the flow calls the referenced flow. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of `null` is converted to an empty string. in the referenced flow.

Store Output Values in Flow Types Except Prompt Flows

To use the referenced flow's outputs later in the flow, store them in variables. The values are assigned when the referenced flow finishes running. In the Store Output Values tab, the Subflow elements show only output variables from the active version and the latest version of the referenced flow.

Store Output Values in Prompt Flows

To use the outputs of the referenced flow later in the flow, either reference the output of the Subflow element or store them as manually assigned variables.

Usage

- Flow Builder doesn't display descriptions for input and output values. For details about each variable in the referenced flow, ask the admin who built the flow for more information.
- In API version 61.0 and later, screen flows and record-triggered flows call the active version of a referenced flow by default. If a referenced flow has no active version, the flow calls the latest version of the referenced flow. In previous API versions, screen flows call the latest version of a referenced flow by default. To run only the latest version of each referenced flow, you have two options.
  - In Flow Builder, click **Debug**, select **Run the latest version of each flow called by subflow elements**, and then click **Run**. This option isn't available when you debug a flow in the Flow Builder canvas.
  - Append the URL for the parent flow with `?latestSub=true`.
- In API version 62.0 and later, the prompt flow type supports the Subflow element. The prompt flow can call the active version of a referenced prompt flow by default. If a referenced prompt flow has no active version, the flow calls the latest version of the referenced prompt flow.

Flow Example: Subflow in Prompt Flow

Let's build a flow that uses a referenced flow to create meeting invites for each marketing event generated by the sales email template-triggered prompt flow.

SEE ALSO:

Flow Elements

Add and Edit Elements

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Lightning Runtime Considerations

Flow Example: Subflow in Prompt Flow

Let's build a flow that uses a referenced flow to create meeting invites for each marketing event generated by the sales email template-triggered prompt flow.

Template-triggered prompt flows aren't compatible with prompt templates created in Winter '24.

Let's say you're a Salesforce admin. You create a flow that sends instructions to the prompt template to get Marketing events in the same city and state where your contacts live. Now, you can add a prompt flow as a referenced flow in the parent flow. This new flow creates a meeting invite for your contacts for each marketing event.

Before creating your parent flow, you create a flow that retrieves a list of marketing events. See Example of Sales Email Template-Triggered Prompt Flow on page 45.

1. Create the flow.
   a. Click **New Flow**.
   b. From Flow Builder, select **Start from Scratch**, and then click **Next**.
   c. Select **Template-Triggered Prompt Flow**, and then click **Create**.
2. Configure the flow.
   a. Select **Automatic Inputs**.
   b. For Prompt Template Type, select **Sales Email Template**.

      Each prompt template type is associated with its prompt template type in Prompt Builder.
   c. For Recipient, select **Contact**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, and **Unlimited** Editions with the Einstein for Sales, Einstein for Platform, or Einstein for Service add-on

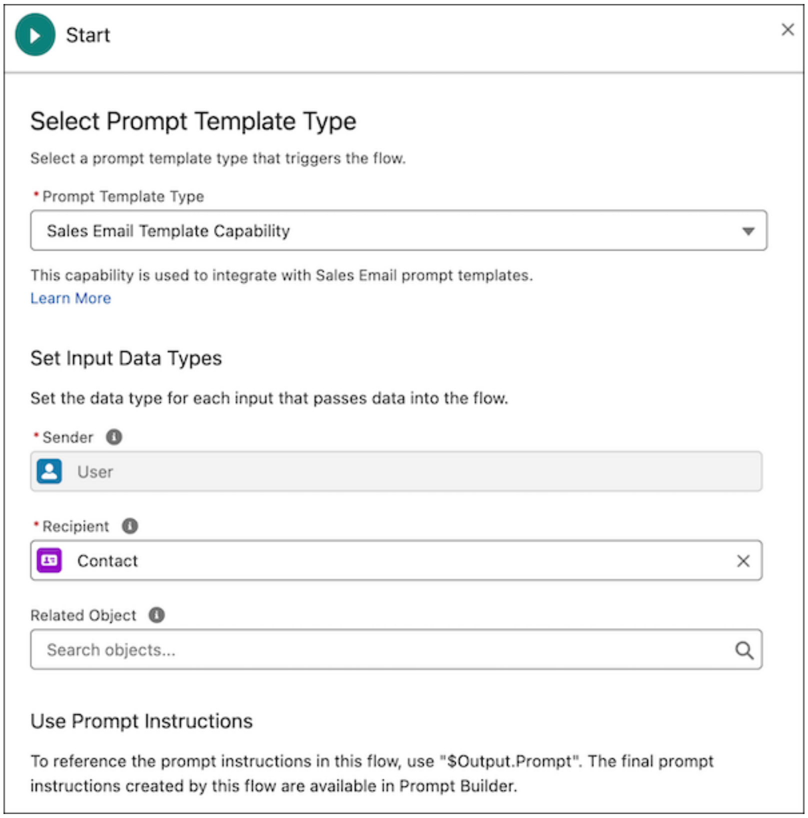
USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and manage prompt templates in Prompt Builder:

- Prompt Template Manager permission set



Select Prompt Template Type

Select a prompt template type that triggers the flow.

*Prompt Template Type: Sales Email Template Capability

This capability is used to integrate with Sales Email prompt templates.

Set Input Data Types

Set the data type for each input that passes data into the flow.

*Sender: User

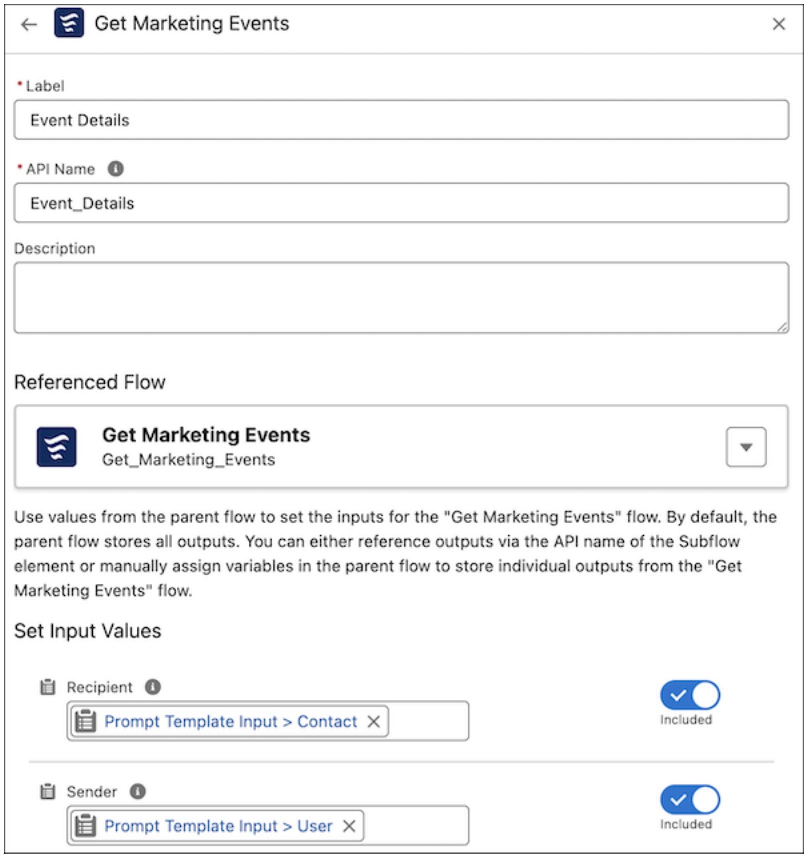
*Recipient: Contact

Related Object: Search objects...

Use Prompt Instructions

To reference the prompt instructions in this flow, use "$Output.Prompt". The final prompt instructions created by this flow are available in Prompt Builder.

3. Add a Subflow element and configure it.
   a. Click ⊕, and then click the **Subflow** element.
   b. For Label, enter `Event Details`, and use the default API name `Event_Details`.
   c. For Recipient, select **Prompt Template Input** > **Contact**.
   d. For Sender, select **Prompt Template Input** > **User**.



Get Marketing Events

* Label

Event Details

* API Name

Event_Details

Description

Referenced Flow

Get Marketing Events
Get_Marketing_Events

Use values from the parent flow to set the inputs for the "Get Marketing Events" flow. By default, the parent flow stores all outputs. You can either reference outputs via the API name of the Subflow element or manually assign variables in the parent flow to store individual outputs from the "Get Marketing Events" flow.

Set Input Values

Recipient

Prompt Template Input > Contact

Included

Sender

Prompt Template Input > User

Included

4. Add the Add Prompt Instructions element and configure it.
   a. Click ⊕, and then click the **Add Prompt Instructions** element.
   b. For Label, enter `Meeting Invite`, and use the default API name `Meeting_Invite`.
   c. For Prompt Instructions, enter `Send a meeting invite to {!$Input.Recipient.Name} with {!Event_Details.Prompt}`.

5. Save your flow.
   a. For Flow Label, enter `Send Marketing Events Invite`, and use the default API name `Send_Marketing_Events_Invite`.
   b. Save your work.

6. Before activating the flow, click **Debug** to troubleshoot for any flow errors.

7. Set the debug options and input variables.

   If you want to run the flow as another user, ensure that **Let admins debug flows as other users** is enabled in **Process Automation Settings**.

8. Click **Run**.

   The debug details for the run appear in a panel on the right.

9. To see the results, review the debug details. If the flow fails, troubleshoot the flow errors on page 233.

10. Activate the flow.

Now that you completed the flow, you create a sales email prompt template in Prompt Builder. You add the parent flow to the prompt template. When you preview the prompt template in Prompt Builder, it triggers the parent flow, which then runs the referenced flow. The parent flow sends its prompt instructions to the prompt template.

SEE ALSO:

Example of Sales Email Template-Triggered Prompt Flow

Trailhead: Run a Flow Within a Flow

Flow Element: Transform

Select the flow resources for mapping and transforming source data to target data. You can use the Transform element in screen flows, autolaunched flows with no triggers, and record-triggered flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
Source Data	The data to transform. - Resource—Flow resources that are available to the flow. You can add multiple resources.
Target Data	The data after it's transformed - Allow multiple values (collection)—Select if the target data is a collection. The source data must contain one or more collections. - Apex Class—The Apex class to use for the target data structure. - Data Type—The data type of the flow resource for the target data. - Object—The object to use for the target data structure.
Formula	The formula expression that transforms the data. The formula expression can: - Set a fixed value for the target data field. - Reference up to two source collections that are nested within one another. For example, a source data field contains collection A, which contains collection B. The formula expression can reference additional flow resources as source data if the flow resources don't reference a collection. The formula result must be compatible with the target field data type. In the formula, the [$EachItem] merge field syntax represents each item in a collection. For example, the merge field references a collection of orders, and each order item has a Customers field. Each

Field	Description
	Customers field contains a Name field: {!Orders[$EachItem].Customers[$EachItem].Name}

Usage

To use the Transform element, understand these general limitations.

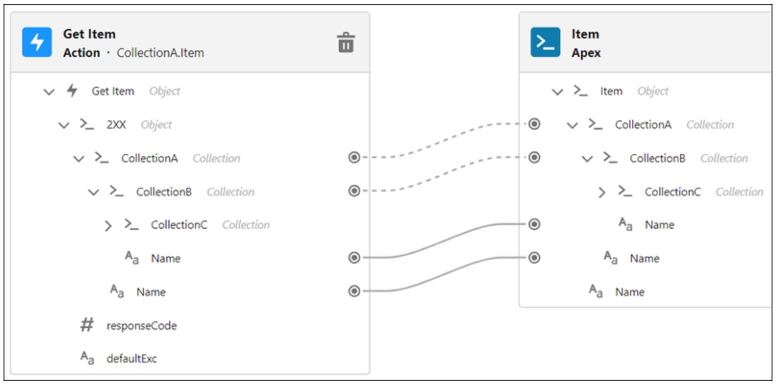
- When you transform a collection, the transformation can't include joining, sorting, or filtering data. To filter or sort a collection, you can use the Collection Filter or Collection Sort element in the flow instead.
- Viewing the debug details of the source and target data in a rich and interactive format is supported only for autolaunched flows with no triggers and record-triggered flows.
- When the resource for target data is an Apex class from an external service registration, the debug details in Flow Builder show modified field names.

  Note: When an external service is registered, Salesforce creates an Apex class that stores the input and output of the service.

  - If a field uses a reserved name, `z0` is prepended to the field name, for example, z0type. When the flow calls out to the external service, the original field name—in this example, type—is used. See Apex Reserved Keywords.
  - Fields that append `_set`, such as name_set, are added automatically to the dynamic Apex class. The fields appear only in the flow debug details and when you inspect the dynamic Apex class. See External Service Registrations in Apex.
- Accessing related records via lookup fields on standard, custom, and external objects isn't supported.
- The Checkbox Group, Picklist, and Choice Lookup screen components aren't supported as a flow resource for the source or target data.

Understand these rules that preserve the integrity of the data structure in collections.

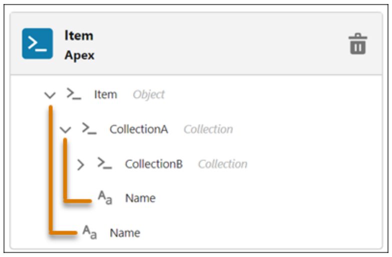
- When mapping a field in a source collection to a field in the target collection, both collections must be at the same hierarchical level in their respective resources. For example, the collection A in the source data and collection A in the target data aren't within any other collections. Because they're both the top collections in their respective resources, you can map fields between them.
- Before you map a field in a collection that's within another collection, map the field in the parent collection. For example, the flow resources for the source and target data each contain collection A, which has the same data structure. Collection A contains collection B. Before you map fields in collection B, map a field in collection A, and then map a field in collection B.



At run time, if a target data field isn't mapped or is null, it's removed from the flow resource that's generated from the Transform element.

Limits

- The Transform element supports mapping up to one nested collection.
- A field on an Apex-defined flow resource can reference up to 10 levels of Apex-defined fields within it. For example, the Name field on CollectionA is the first level. A field on CollectionB is the second level, and so on.



- Debug details show up to 20 records in a collection.
- In the Transform element, a formula expression can contain up to 255 characters. Characters that exceed 255 are truncated. To enter more than 255 characters in a formula expression, you can create a formula resource in the flow. The formula resource can exceed 255 characters in its formula expression. In the Transform element, you can select the formula resource when you use a formula to transform data.

SEE ALSO:

Transform Data in a Flow

Sum or Count Items in Collections with the Transform Element

Flow Element: Update Records

Identify Salesforce records to update, and set the values to change in those records. To do so, use the IDs and field values stored in a record variable or record collection variable, or use specify conditions to identify the records and set the field values individually.

> Note: Looking for the Fast Update and Record Update elements from Cloud Flow Designer? The Update Records element combines the functionality from both elements. For the equivalent of a Fast Update element, choose to use the IDs and field values from a record variable or record collection variable. For the equivalent of a Record Update element, choose to specify conditions to identify the records and set the field values individually.

In the Update Records element, your selection for how to identify the records or related records to update and set their values determines what to enter in the rest of the element.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Use a record variable or record collection variable

If you've populated a record variable or record collection variable with the field values to change, choose to use the IDs and field values from a record variable or record collection variable, and then select the variable to use. To update the field values in a record variable or record collection variable, configure an Assignment element earlier in the flow.

> Important: For the variable that you select, make sure that each record's ID value is set. That ID value is how the flow identifies which records to update.

When you use a record collection variable to update multiple records at once, you reduce the number of DML requests in your flow. That means you're more likely to stay within your org's limits. For more information, see Flow Bulkification in Transactions.

Use conditions and set fields individually

Otherwise, choose to use conditions and set fields individually. Choose the object whose records or related records you want to update, add conditions to filter down the list of records, and set the field values to change for those records. You can update any field on the record, but the Update Records element doesn't know which fields are required for this object.

Important: Configure at least one filter condition, or the flow updates all the records for the object.

Example: On an opportunity record, when a user clicks the "Won" button, a flow updates the opportunity's stage.

Considerations for Defining Filter Criteria

- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

  For example, your filters check whether a case's Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At run time, the filters are combined to `(1 OR 2) AND 3`.

- The available filter operators depend on the data type of the selected fields. For details, see Flow Operators in Data Elements and Record Choice Sets.

Usage

Note: At run time, the record isn't updated until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

SEE ALSO:

Flow Operators in Data Elements and Record Choice Sets

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Elements

Flow Builder Elements for Marketing Cloud

A Flow Builder element represents an action that a flow can execute. Examples include decisions based on criteria and creating and deleting Salesforce data. Some Flow Builder elements are available only in Marketing Cloud, such as Send Email Message and Send SMS Message.

In Marketing Cloud, Flow Builder builds flows in auto-layout. In auto-layout, click ⊕ to display the types of elements that you can add.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Starter**, **Enterprise**, and **Unlimited** Editions with Marketing Cloud **Growth** Edition

Marketing Cloud Growth Campaign Flow Element: Send Email
The Send Email Message action in a Marketing Cloud Growth campaign flow sends an email from your Salesforce CMS to an audience segment. You can configure the action to track clicks and opens, send messages to your opt-in list only, and get help from Einstein for send-time optimization (STO) and identifying which clicks and opens are real.

Marketing Cloud Growth Campaign Flow Element: Send SMS
The Send SMS Message action in a Marketing Cloud Growth campaign flow sends an SMS message from your Salesforce CMS to an audience segment. You can configure the action to track clicks and opens, send messages to your opt-in list only, and get help from Einstein for send-time optimization (STO) and identifying which clicks and opens are real.

Flow Element: Path Experiment

Experiment with up to 10 different versions of a customer journey to determine the most effective path. The Path Experiment element randomly assigns individuals to paths for an unbiased outcome.

Marketing Cloud Growth Campaign Flow Element: Wait Until Event

Resume a flow interview after an engagement event.

Marketing Cloud Growth Campaign Flow Element: Send Email

The Send Email Message action in a Marketing Cloud Growth campaign flow sends an email from your Salesforce CMS to an audience segment. You can configure the action to track clicks and opens, send messages to your opt-in list only, and get help from Einstein for send-time optimization (STO) and identifying which clicks and opens are real.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Starter**, **Enterprise**, and **Unlimited** Editions with Marketing Cloud **Growth** Edition

Set Input Values

You select the segment from the campaign record or from the flow's Start element. You can customize the contents of the email from the campaign record.

Input Parameter	Description
`Email`	The email content to be sent. This field is populated with the email template that was created when you created the campaign. If you remove the email from this field, you can select a different one from your content workspace in your CMS.
`Einstein Send Time Optimization`	Optional. Einstein Send Time Optimization (STO) determines the best time to send a message. Using machine learning, Einstein predicts optimal send times so that a user is more likely to engage with your message. A Salesforce admin must enable this feature.
`Einstein Metrics Guard`	Optional. Einstein Metrics Guard works behind the scenes to filter out most email security scanner clicks and opens without blocking legitimate visitor activity. A Salesforce admin must enable this feature.
`Select Sender`	The sender for the email message. Sender addresses are configured in Setup in Organization-Wide Addresses.
`Track Clicks`	Optional. If enabled, the flow tracks how many times email recipients click links that are included in your message.
`Track Opens`	Optional. If enabled, the flow tracks whether recipients open your emails.
`Communication Subscription`	Optional. If you select a communication subscription, the message is sent only to recipients who opt in to receiving it.

Usage

The email is sent based on the schedule configured in the Start element. If there are multiple Send Email elements separated by Wait elements, the flow doesn't send the email until the Wait conditions are met.

The monthly limit of email sends is 15,000.

Considerations

Emails sent using the Send Email Message element don't include email signatures from My Email Settings. To include a signature, add one to the email template.

Marketing Cloud Growth Campaign Flow Element: Send SMS

The Send SMS Message action in a Marketing Cloud Growth campaign flow sends an SMS message from your Salesforce CMS to an audience segment. You can configure the action to track clicks and opens, send messages to your opt-in list only, and get help from Einstein for send-time optimization (STO) and identifying which clicks and opens are real.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions with Marketing Cloud **Growth** Edition

Set Input Values

You select the segment from the campaign record or from the flow's Start element. You customize the contents of the SMS message from the campaign record.

Input Parameter	Description
`SMS`	The SMS message content to be sent. This field is populated with the SMS content that was created when you created the campaign. If you remove the SMS message content from this field, you can select different content from your content workspace in your CMS.
`Sender ID`	The sender for the SMS message. This address is configured in Unified Messaging Setup.
`Track Clicks`	Optional. If enabled, the flow tracks how many times message recipients click links that are included in your message.
`Communication Subscription`	Optional. If you select a communication subscription, the message is sent only to recipients who opt in to receiving it.
`Communication Subscription Channel Type ID`	The ID of the communication subscription channel type that points to the communication subscription.

Usage

The SMS message is sent based on the schedule configured in the Start element. If there are multiple Send SMS Message elements separated by Wait elements, the flow doesn't send the SMS message until the Wait conditions are met.

Flow Element: Path Experiment

Experiment with up to 10 different versions of a customer journey to determine the most effective path. The Path Experiment element randomly assigns individuals to paths for an unbiased outcome.

Prerequisites

To add the Path Experiment element to a flow, the user requires either the Marketing Cloud Manager or Marketing Cloud Admin permission set. The Path Experiment element also requires Einstein Personalization. To set up Einstein Personalization, use Marketing Cloud Assisted Setup. Einstein Personalization setup is found in Reporting and Optimization, on the Customer Engagement tab. After setting up Einstein Personalization, return to the Customer Engagement tab. Under the Configure Basic Personalization section, select a data graph.

Path Experiment is available only for segment-triggered flows.

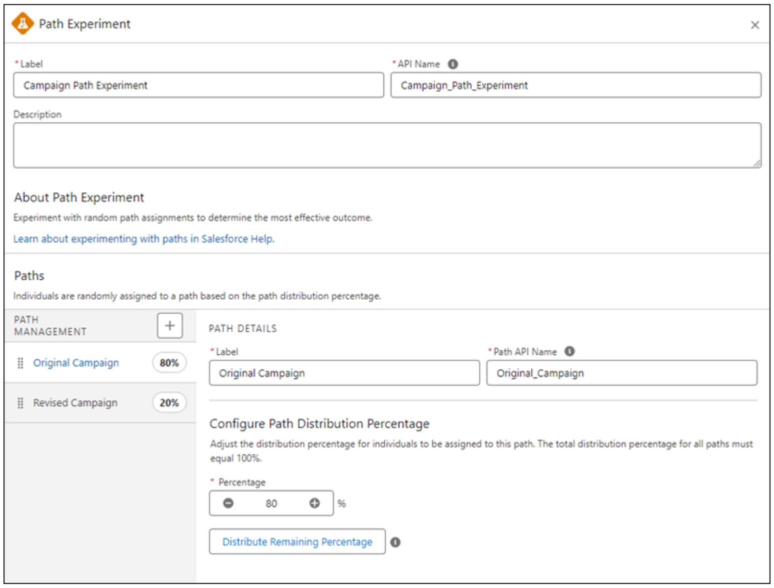
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions with **Marketing Cloud Advanced Edition**.

Paths

Create a path for each customer journey you want to test. For each path, specify what percentage of the total audience you want to go down that path. Individuals are randomly assigned an Experiment cohort and flow path based on the path's distribution percentage. If an individual reenters the experiment in any way, for example through a loop or Go To connector, they're always assigned the same path as their first assignment.



Path Experiment

*Label: Campaign Path Experiment

*API Name: Campaign_Path_Experiment

Description:

About Path Experiment

Experiment with random path assignments to determine the most effective outcome.

Learn about experimenting with paths in Salesforce Help.

Paths

Individuals are randomly assigned to a path based on the path distribution percentage.

PATH MANAGEMENT

Original Campaign 80%

Revised Campaign 20%

PATH DETAILS

*Label: Original Campaign

*Path API Name: Original_Campaign

Configure Path Distribution Percentage

Adjust the distribution percentage for individuals to be assigned to this path. The total distribution percentage for all paths must equal 100%.

* Percentage: 80 %

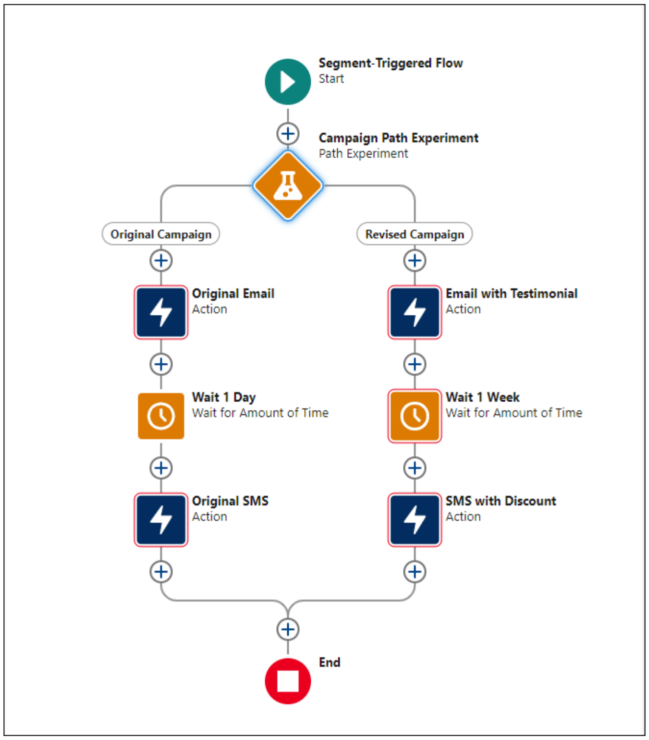
Field	Description
`Label`	Identifies the path on the canvas.
`Path API Name`	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same API name, provided they're used in different flows. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Percentage`	The distribution percentage of individuals to send down this path. The total distribution percentage across all paths must equal 100%.

Example: With the Path Experiment element, you can test the effectiveness of two subscription renewal campaigns.

- Send 80% of the individuals to Path 1, and send them a renewal email. The flow waits for one day before sending a follow-up SMS.
- Send 20% of the individuals to Path 2, and send them a renewal email that includes testimonials. The flow waits for one week before sending a follow-up SMS with a discount code.

To see more details about the results of a Send Email action or other type of message engagement, create a Data Cloud report using one of the Message Engagement DMOs, filtered by the flow element ID.

Example:



SEE ALSO:

Salesforce Help: Data Cloud Reports and Dashboards

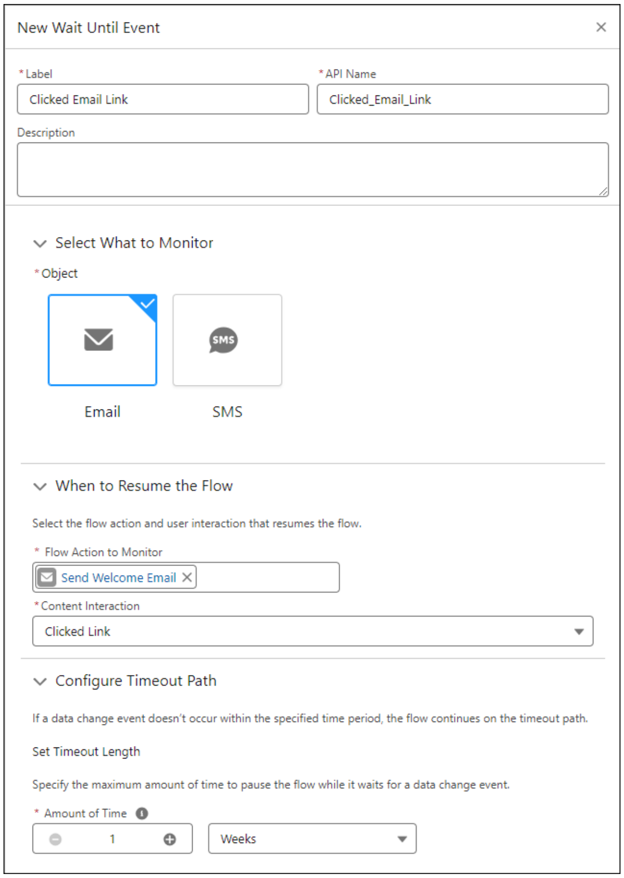
Marketing Cloud Growth Campaign Flow Element: Wait Until Event

Resume a flow interview after an engagement event.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise** and **Unlimited** Editions with Marketing Cloud **Growth** Edition



New Wait Until Event

* Label: Clicked Email Link

* API Name: Clicked_Email_Link

Description

Select What to Monitor

* Object

When to Resume the Flow

Select the flow action and user interaction that resumes the flow.

* Flow Action to Monitor: Send Welcome Email

* Content Interaction: Clicked Link

Configure Timeout Path

If a data change event doesn't occur within the specified time period, the flow continues on the timeout path.

Set Timeout Length

Specify the maximum amount of time to pause the flow while it waits for a data change event.

* Amount of Time: 1 Weeks

Field	Description
`Object`	The object to monitor for engagements.
`Flow Action to Monitor`	The flow action to monitor for engagements.
`Content Interaction`	The kind of engagement that resumes the flow.

Field	Description
`Amount of Time`	The maximum amount of time to wait. If the duration is exceeded, the flow resumes and follows a timeout path.

SEE ALSO:

Flow Elements: Wait

Provided Flow Core Actions

Perform an action outside of the flow. Choose from Salesforce-provided actions, like Submit for Approval or Send Email, or from your org's quick actions and local actions. To add one of these actions to your flow, add an Action element. Then, in the Action field, search for the appropriate action.

Flow Core Action: Activate Session-Based Permission Set

Activate a session-based permission set for the running user.

Flow Core Action: Deactivate Session-Based Permission Set

Deactivate a session-based permission set for the running user.

Flow Core Action: Einstein Discovery

Get predictive and prescriptive intelligence directly in your flows with Einstein Discovery-powered models. Select the row or fields to use for your predictions and let Einstein Discovery generate predictions, suggested ways to improve predicted outcomes, and other details.

Flow Core Action: Get Forecast Context

Get forecast context for a specific user. To be used in the Forecast Guidance Flow as part of the Get Forecast Guidance copilot action.

Flow Core Action: Get Forecast Opportunities

Get forecast opportunities for a user that matches the specified criteria. To be used in the Forecast Guidance flow as part of the Get Forecast Guidance copilot action.

Flow Core Action: Get Record Prioritization Data

Get record data and field metadata to prioritize records for a user. This action is used in the Get Opportunity Details flow as part of the Prioritize Opportunities copilot action.

Flow Core Action: Lock Record

Locks or unlocks a workflow-enabled or approval-enabled record for editing during an approval and specifies who can edit the record while it's locked.

Flow Core Action: Post to Chatter

Post a message to a specified feed, such as a Chatter group or a case record. The message can contain mentions and topics, but only text posts are supported.

Flow Core Action: Prompt Template Actions

Creates a response based on the large language model (LLM) response for the specified prompt template and inputs.

Flow Core Action: Global or Object-Specific Action

Call an object-specific or global action that's already been configured in your org. Only Create, Update, and Log a Call actions are available.

Flow Core Action: Run a Batch Data Transform in Data Cloud

Run a batch data transform.

Quip Flow Core Actions

Quip provides several core actions for organizing, creating, and copying your Quip content in flows. To add one of these actions to your flow, add an Action element. Then select the **Quip** category, and search for the appropriate action.

B2B Commerce Checkout Flow Core Actions

The B2B Commerce Checkout Flow provides several core actions for implementing a successful checkout process within your Commerce org. To add one of these actions to your flow, add an Action element. Then select the **B2B Commerce** category, and search for the appropriate action.

Commerce Checkout Flow Core Actions

The Commerce Checkout Flow provides several core actions for implementing a successful checkout process within your Commerce org. To add one of these actions to your flow, add an Action element. Then select the **Commerce** category, and search for the appropriate action. Cart actions aren't available in flows for B2B stores built on an Aura template.

Salesforce Order Management Flow Core Actions

Salesforce Order Management provides several core actions for implementing order management functionality in flows. To add one of these actions to your flow, add an Action element. Then select the **Order Management** category, and search for the appropriate action.

Salesforce Omnichannel Inventory Flow Core Actions

Salesforce Omnichannel Inventory provides several core actions for implementing inventory functionality in flows. To add one of these actions to your flow, add an Action element. Then select the **Omnichannel Inventory Service** category, and search for the appropriate action.

Flow Core Actions: Send Conversation Messages

Send a messaging component to one or more messaging users in enhanced WhatsApp, enhanced Apple Messages for Business, enhanced SMS, or Messaging for In-App.

Flow Core Action: Send Custom Notification

Add the Send Custom Notification action to a flow, then add recipients and content.

Flow Core Action: Send Email

Send and optionally log an email by specifying the email content and recipients in a flow. If you're using Marketing Cloud Growth, use the Send Email Message on page 377 element to send an email to your audience segment.

Flow Core Action: Send Notification Actions

Call a notification type to send. Each Send Notification action corresponds to a supported notification type. Send Notification actions are available only for Slack-enabled custom notification types and certain Slack-enabled standard notification types.

Flow Core Action: Send Surveys

Create an action to send an active survey by specifying the name, subject, recipients, and invitation link options in the flow.

Flow Core Action: Perform Survey Sentiment Analysis

Get insights into the sentiments that underlie survey responses.

Flow Core Action: Get Assessment Response Summary

Create a printable summary view of assessments taken. This action enables you to extract responses saved in an assessment and create a flow to generate a document.

Slack Flow Core Actions

Manage Slack channels, channel members, and messages from flows. As your Salesforce records change, a flow can trigger changes in Slack.

Flow Core Action: Submit for Approval

Submit one Salesforce record for approval.

Salesforce Anywhere Core Flow Actions (Beta)

Salesforce Anywhere provides several core actions for implementing Salesforce Anywhere functionality in flows. To add one of these actions to your flow, add an Action element. Then select the Salesforce Anywhere category, and search for the appropriate action.

SEE ALSO:

Add and Edit Elements

Flow Core Action: Activate Session-Based Permission Set

Activate a session-based permission set for the running user.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Permission Set`*, and select **Activate Session-Based Permission Set**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to identify the permission set to activate.

Important: You can run queries, but don't modify Salesforce data in flows that also activate session-based permission sets.

Field	Description
`Permission Set Name`	The developer name of the permission set. This parameter accepts single-value resources of any type. That value is treated as text.
`Permission Set Namespace`	Optional. The permission set's namespace. This parameter accepts single-value resources of any type. That value is treated as text.

Example: A junior buyer in your org occasionally requires access to your Contracts object. Create a session-based permission set with access to the object. Then create a flow that uses the Activate Session-Based Permission Set core action. Configure the action to activate the permission set.

The junior buyer runs the flow to access contracts during the current user session. The action activates the permission set for the junior buyer during the current session.

SEE ALSO:

Flow Core Action: Deactivate Session-Based Permission Set

Create a Flow That Can Activate or Deactivate a Session-Based Permission Set

Plan for Success in Flow Builder

Add and Edit Elements

Flow Core Action: Deactivate Session-Based Permission Set

Deactivate a session-based permission set for the running user.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Permission Set*, and select **Deactivate Session-Based Permission Set**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to identify the permission set to deactivate.

Field	Description
Permission Set Name	The developer name of the permission set. This parameter accepts single-value resources of any type. That value is treated as text.
Permission Set Namespace	The permission set's namespace. This parameter accepts single-value resources of any type. That value is treated as text.

SEE ALSO:

- Flow Core Action: Activate Session-Based Permission Set
- Create a Flow That Can Activate or Deactivate a Session-Based Permission Set
- Plan for Success in Flow Builder
- Add and Edit Elements

Flow Core Action: Einstein Discovery

Get predictive and prescriptive intelligence directly in your flows with Einstein Discovery-powered models. Select the row or fields to use for your predictions and let Einstein Discovery generate predictions, suggested ways to improve predicted outcomes, and other details.

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Set Input Values

Note: To view Einstein Discovery predictions, improvements, and other details, users must have the **View Einstein Discovery Recommendations** system permission. To learn more, see Assign Einstein Discovery Permission Sets to Users .

Use values from an Einstein Discovery model to set the inputs for the action.

Field	Description
Action	Search for the deployed models to which you have access.
Label	Descriptive label for the action.
API Name	API name for the action.
Description	Description for the action.

Field	Description
What to Store	
`Predictions`	Enable output from a predictive model to be stored in a flow resource.
`Top Predictors`	Enable top predictors to be stored in a flow resource.
`Top Improvements`	Enable suggested actions to be stored in a flow resource. Applies only to numeric and binary classification models.
`Select Object Record ID Field`	Generate predictions based on a Salesforce object record.
`Map Fields`	Generate predictions using flow resources.
`Record ID Field`	ID of the record to use for the prediction.
`Model Variable`	Map the prediction model variables to flow resources.
`Advanced`	Optionally, for predictions associated with multiclass classification models, expand Advanced, select **Manually assign variables**, and selectively store output values (class probabilities, the prediction, and top predictors).

SEE ALSO:

About Models

Add and Edit Elements

Flow Core Action: Get Forecast Context

Get forecast context for a specific user. To be used in the Forecast Guidance Flow as part of the Get Forecast Guidance copilot action.

In Flow Builder, add an Action element to your Flow. In the New Action window, search for Get Forecast Context, and then select **Get Forecast Context**.

EDITIONS

Available in: Lightning Experience

Available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Field	description
User ID	The ID of the user to provide forecasting context for.

Store Output Values

Field	description
Can Access Target User Forecasts	A Boolean value that indicates whether the running user has access to the specified user's forecasts.
Current Period ID	The ID of the period that the forecasting context applies to.

Field	description
Cumulative Forecasts	A Boolean value that indicates whether the Cumulative Forecasts setting is enabled.
Most Likely	A Boolean value that indicates whether the Most Likely setting is enabled.
Commit Label	The label that was used for the commit category.
Most Likely Label	The label that was used for the Most Likely category.
Opportunity Score	A Boolean value that indicates whether the Opportunity Score setting is enabled.
Push Count	A Boolean value that indicates whether the Push Count setting is enabled.
Domain URL	The URL to use to generate links to user and opportunity forecast contexts.

Usage

The Get Forecast Context action is a part of the Forecast Guidance Flow, designed to retrieve forecasting context for a specific user. It's primarily used in conjunction with the Get Forecast Guidance copilot action.

Limitations

This action is limited to usage within Flow. It can't be invoked through other methods such as Apex, REST API, Copilot, Prompt Studio, or Einstein Bots. While actions can typically be invoked through various frameworks like InvocableActionService, this specific action is restricted solely to Flow.

SEE ALSO:

Salesforce Help: Copilot Action: Get Forecast Guidance

Flow Core Action: Get Forecast Opportunities

Get forecast opportunities for a user that matches the specified criteria. To be used in the Forecast Guidance flow as part of the Get Forecast Guidance copilot action.

In Flow Builder, add an Action element to your flow. In the New Action window, search for Get Forecast Opportunities, and then select **Get Forecast Opportunities**.

EDITIONS

Available in: Lightning Experience

Available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

field	description
User ID	The ID of the user to provide forecasting guidance context for.

field	description
Forecasting Type ID	The ID of the forecasting type to include in returned forecasting opportunities.
Period ID	The ID of the time period that opportunities must fall within to be included in the returned forecasting opportunities.
Forecast Category	The forecast category of opportunities to include in returned forecasting opportunities. Valid values when a user has single forecast category rollups are Pipeline, Best Case, Commit, and Closed. Valid values when a user has cumulative forecasts are Open Pipeline, Best Case Forecast, Commit Forecast, and Closed Only. Valid value when Most Likely is enabled is Most Likely.
Forecasting Opportunity Count	The maximum number of forecasting opportunities to return. Valid values are from 1 to 500.

Store Output Values

field	description
Forecast Opportunity IDs	A collection of IDs of opportunity records that match the specified criteria for the user.

Usage

The Get Forecast Opportunities retrieves opportunity ID records for a user based on specified criteria. It's designed mainly to be used within the Forecast Guidance flow as part of the Get Forecast Guidance copilot action.

Limitations

This action is limited to usage within a flow. It can't be invoked through other methods such as Apex, REST API, Copilot, Prompt Studio, or Einstein Bots. While actions can typically be invoked through various frameworks like InvocableActionService, this specific action is restricted solely to Flow.

SEE ALSO:

Salesforce Help: Copilot Action: Get Forecast Guidance

Flow Core Action: Get Record Prioritization Data

Get record data and field metadata to prioritize records for a user. This action is used in the Get Opportunity Details flow as part of the Prioritize Opportunities copilot action.

In Flow Builder, add an Action element to your flow. In the New Action window, search for *Prioritization*, and select **Get Record Prioritization Data**.

EDITIONS

Available in: Lightning Experience

Available in: **Professional**, **Enterprise**, **Performance**, Unlimited, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

field	description
Layout Type	The type of layout, which determines the fields to retrieve data for. Valid values are: • Full: Based on the default Full layout record. • Compact: Based on the primary Compact layout record. • Search: Based on all Search layout records. The default value is Compact.
Optional Fields	The optional fields to retrieve data for.
Record ID	Required. The record IDs to retrieve data for.

Store Output Values

field	description
Field Metadata	The field metadata for the record IDs.
Record Data	The record data for the record IDs.

Usage

The Get Record Prioritization Data action is a part of the Get Opportunity Details flow, which is designed to retrieve record data and field metadata record prioritization. It’s primarily used with the Prioritize Opportunities copilot action.

Limitations

This action is limited to usage within flow. It can’t be invoked through other methods such as Apex, REST API, Copilot, Prompt Studio, or Einstein Bots. While actions can typically be invoked through frameworks such as InvocableActionService, this specific action is restricted to flow.

Flow Core Action: Lock Record

Locks or unlocks a workflow-enabled or approval-enabled record for editing during an approval and specifies who can edit the record while it's locked.

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

In Flow Builder, add an Action element to your flow. In the New Action window, search for *`Lock Record`*, and then select **Lock Record**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Field	Description
Action	Required. Specifies the action to perform on the record. Valid values are: • `Lock` • `Unlock`
Record ID	Required. The ID of the record to be locked or unlocked.
Allowed ID	Optional. The ID of a user, group, queue. or user role record that represents who can edit the record while it's locked.

Usage

This action is available in flows running in API version 61.0 and later.

To lock a record, a user must have view access to the record.

To edit a locked record, an allowed user must have edit access to the record, but system admins can always edit a locked record.

If a user role is specified for the Allowed ID parameter, a user must have a user role on or above the specified user role in the role hierarchy to be able to edit the locked record.

A group or queue specified for the Allowed ID parameter must be a group or queue of users. The specified group or queue can't contain:

- roles
- roles and internal subordinates.

An attempt to lock a record that's already locked results in an error with the UNABLE_TO_LOCK_RECORD error code.

A system admin, the user who locked a record, or someone who's an allowed user for the locked record can unlock the record. An attempt by any other user results in an error with the INSUFFICIENT_ACCESS_OR_READ_ONLY error code.

The action completes without error when the action attempts to unlock a record that's already unlocked.

Limitations

If a record is locked with this action and the record is submitted to an approval process, then the approval process overwrites the record lock.

- If the approval process overwrites the lock from this action, then the allowed users specified by this action can no longer edit the record.
- The approval process can be interrupted when this action unlocks a record locked by the approval process.

SEE ALSO:

Salesforce Help: Approval Processes

Flow Core Action: Post to Chatter

Post a message to a specified feed, such as a Chatter group or a case record. The message can contain mentions and topics, but only text posts are supported.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Chatter*, and select **Post to Chatter**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the Chatter post.

Input Parameter	Description
Message	The text that you want to post. - To mention a user or group, enter @[*reference*], where *reference* is the ID for the user or group that you want to mention. The reference can be a literal value, a merge field, or a flow resource. For example: @[{!UserId}]. - To add a topic, enter #[*string*], where *string* is the topic that you want to add. For example: #[Action Required]. This parameter accepts single-value resources of any type. That value is treated as plain text and is limited to 10,000 characters.
Target Name or ID	Reference to the user, Chatter group, or record whose feed you want to post to. - To post to a user's feed, enter the user's ID or username. For example: *jsmith@salesforce.com* - To post to a Chatter group, enter the group's name or ID. For example: *Entire Organization* - To post to a record, enter the record's ID. For example: *001D000000JWBDx* This parameter accepts single-value resources of any type. That value is treated as text.
Community ID	ID of an Experience Cloud site to post to. Valid only if Digital Experiences is enabled. Required if posting to a user or Chatter group that belongs to an Experience Cloud site.

Input Parameter	Description
	This parameter accepts single-value resources of any type. That value is treated as text.
`Target Type`	Required only if `Target Name or ID` is set to a username or a Chatter group name. The type of feed that you want to post to. Valid values are: - *`User`*—If `Target Name or ID` is set to a user's username, enter this value. - *`Group`*—If `Target Name or ID` is set to a Chatter group's name, enter this value.
`Visibility`	Specifies whether this feed item is available to Experience Cloud site users. To display this feed item only to internal users, set it to *`internalUsers`*. Valid only if Digital Experiences is enabled. Valid values are: - *`allUsers`* - *`internalUsers`*

Store Output Values

To use the Chatter post's ID later in the flow, store it in a variable. The values are assigned when the Chatter post is created.

Output Parameter	Description
`Feed Item ID`	Assigns the created post's ID to a resource in the flow. This parameter accepts any single-value variables of type Text, Picklist, or Multi-Select Picklist.

Usage

At run time, the Chatter post isn't created until the interview's transaction completes. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

SEE ALSO:

Flow Run Context

Flow Elements

Flow Core Action: Prompt Template Actions

Creates a response based on the large language model (LLM) response for the specified prompt template and inputs.

In Flow Builder, add an Action element to your flow. In the New Action window, select the **Prompt Template** category, and then select the name of the prompt template to use.

The API name for each action is prefixed with `generatePromptResponse`.

EDITIONS

Available in: Lightning Experience

Available in: Enterprise, Unlimited, and Developer Editions

Set Input Values

Field	Description
Citation	A picklist that specifies how citations are returned: • post_generation: Return citations after generation • off: Don't return citations
Prompt Response	The prompt response generated by the action based on the specified prompt template and input.

Additional input values are based on the input variables specified for the prompt template.

Store Output Values

Field	Description
Citation	The citations for material used during prompt generation.
Prompt Response	The prompt response generated by the action based on the specified prompt template and input.

Usage

This action is available only if the following are true. Otherwise, the action fails.

- Prompt Builder is enabled.
- The user who runs the flow has the Prompt Template User permission.
- The specified prompt template has an active version.

SEE ALSO:

Prompt Builder

Flow Core Action: Global or Object-Specific Action

Call an object-specific or global action that's already been configured in your org. Only Create, Update, and Log a Call actions are available.

In Flow Builder, add an Action element to your flow. In the Action field, select the object-specific or global action to configure.

The API name for each object-specific action is prefixed with the object it's associated with, such as `quickAction-Task.UpdatePriority`. The API name for each global action has no prefix, such as `quickAction-NewAccount`.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Field	Description
`Related Record ID`	Only for object-specific actions. The ID of the record from which the action executes. For example, the action creates a case that's associated with a given account. Assign the ID for that account to `Related Record ID`. This parameter accepts single-value resources of any type. That value is treated as text.
`Input Parameter`	Varies for each action. The action layout determines which parameters are required. Required parameters appear by default and can't be removed. If a required field has a default or predefined value, that field is optional in object-specific and global actions in the flow. If you later remove the field's default or predefined value and you didn't set a value in the flow, the interview fails at run time. The value must be compatible with the parameter.

Example: Your org has an object-specific action that creates a case record on an account. The flow calls that action at run time and uses values from earlier in the flow to identify the account ID.

Note: At run time, the record isn't created or updated until the interview's transaction completes. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

SEE ALSO:

Add and Edit Elements

Flow Elements

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Core Action: Run a Batch Data Transform in Data Cloud

Run a batch data transform.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Run Batch Transform**.

Set Input Values

Use values from earlier in the flow to set an input for the action.

Input Parameter	Description
`Batch Transform Name`	Required. The name or the record ID of the batch data transform to run.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Quip Flow Core Actions

Quip provides several core actions for organizing, creating, and copying your Quip content in flows. To add one of these actions to your flow, add an Action element. Then select the **Quip** category, and search for the appropriate action.

EDITIONS

Available in: **Lightning Experience**

Important: Quip core actions don't support system-run flows or automated process users. Quip core actions execute in the context of the user, who is also referred to as the context user. The flow has access to whatever the context user has access to.

Flow Core Actions for Quip: Create Quip Document

Create a document, spreadsheet, or slide deck, and add content to it.

Flow Core Action for Quip: Create Quip Chat

Create a chat room, and send a message to its members.

Flow Core Action for Quip: Create Quip Folder

Create a private folder, or add it to existing folders.

Flow Core Action for Quip: Copy Quip Document

To use a document as a template, create a copy. By default, copied documents are added to the running user's Private folder in Quip.

Flow Core Action for Quip: Copy Quip Content (Retired)

Copy content from a source slide deck, and paste it in a target slide.

Flow Core Action for Quip: Copy with Live Paste

Copy content from a source document, and paste it with Live Paste in a new document. When the source content is updated, documents with the live pasted content stay up to date.

Flow Core Action for Quip: Attach Quip Document to Record

Attach a document to a Salesforce record. Linked documents show up in the Quip Associated Documents component.

Flow Core Action for Quip: Edit Quip Document

Edit content in a document, spreadsheet, or slide. Add or replace content based on a document section.

Flow Core Action for Quip: Lock Quip Document

To mark a document as complete, lock document edits.

Flow Core Action for Quip: Lock Quip Section

To mark sections of a document as complete or to keep them safe from accidental edits, lock them.

Flow Core Action for Quip: Export Quip Document to PDF

To mark a document as complete and to keep a copy for your records, export it as a PDF. You can attach the exported PDF to a document or to a Salesforce record.

Flow Core Action for Quip: Send Message in Quip Chat

Send a message in a chat room.

Flow Core Action for Quip: Send a Message in a Document

Add a message to the conversations pane of a document.

Flow Core Action for Quip: Add Quip Document to Folder

Add a document to a folder to organize and share your documents.

Flow Core Action for Quip: Add Members to Document

Add members with different levels of access to a document.

Flow Core Action for Quip: Add Members to Quip Chat
Add users to a chat room.

Flow Core Action for Quip: Remove Quip Document from Folder
Remove a document from a folder. Make a shared document private again.

Flow Core Action for Quip: Remove Members from Quip Document
To rescind access to a document for certain users, remove them from the document.

Flow Core Action for Quip: Remove Members from Quip Chat
Remove users from a chat room.

SEE ALSO:

Add and Edit Elements

Flow Core Actions for Quip: Create Quip Document

EDITIONS

Available in: **Lightning Experience**

Create a document, spreadsheet, or slide deck, and add content to it.

Warning: Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Create Quip Document**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document Title`	The title of the new document. Only string values are supported.
`Add Members by Email Address`	Optional. A list of user emails separated by commas to add to the new document. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com* • *person1@quip.com*
`Company Link Address`	Optional. Link sharing settings for the new document. By default, new documents are set to edit-access. Valid values are: • *view*—To let users view the document, enter this value. • *edit*—To let users view and edit the document, enter this value. • *none*—To block user access to the document, enter this value.
`Content Type`	Optional. Format of content added to the document. By default, content format is set to html. Valid values are: • *html*—To format text added to `Document Content` with html, enter this value. • *liveapp*—To add a live app to your document, enter this value. Only valid if `Document Type` is set to *document*.

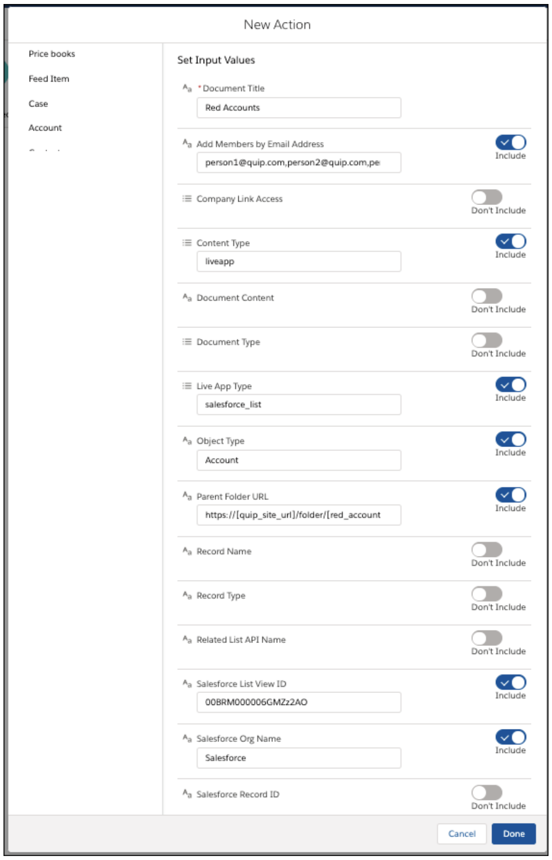
Input Parameter	Description
`Document Content`	Optional. Content added to the new document. Valid only when `Content Type` is set to html. By default, the document title is used for the document's content. Valid values are: - String values - *@[Salesforce user ID]*–To @mention a Salesforce user in the document, enter the Salesforce ID. If the user's Salesforce email is connected to Quip, the user ID is replaced with a Quip user @mention. If not, the Salesforce ID is replaced with the user's Salesforce email. - *@[person1@quip.com]*–To @mention a user by email, enter this value. - *@Everyone*–To send a notification to all users added to document, enter this value.
`Document Type`	Optional. Type of document created including documents and spreadsheets. By default, new documents are created as documents. Valid values are: - *document*–To create a document, enter this value. - *spreadsheet*–To create a spreadsheet, enter this value.
`Live App Type`	Type of live app added to the document. Required if `Content Type` is set to *liveapp*. Only documents support live apps. Valid values are: - *salesforce_record*–To add the Salesforce Record live app to the document, enter this value. - *salesforce_list*–To add the Salesforce List live app to the document, enter this value.
`Object Type`	Type of object used by the Salesforce List live app. Required if `Live App Type` is set to *salesforce_list*. Only string values are supported. For example: *Account*, *Opportunity*, or *CustomObject__c*.
`Parent Folder URL`	Optional. A list of Quip folder URLs separated by commas to add the new document to. By default, the document is added to the user's Private folder in Quip. Valid values are: - *https://[quip_site_url]/folder/[folder1_name], https://[quip_site_url]/folder/[folder2_name]* - https://[quip_site_url]/folder/[folder_name] For example: *https://salesforce.quip.com/folder/account-plans*
`Record Name`	Optional. Name of the record added to the document through the Salesforce Record live app. Valid only if the `Live App Type` is set to *salesforce_record*. Only string values are supported.
`Record Type`	Optional. Type of object used by the Salesforce Record live app. Valid only if `Live App Type` is set to *salesforce_record*. Only string values are supported. For example: *Account*, *Opportunity*, or *CustomObject__c*.
`Salesforce List View ID`	ID of the Salesforce list view added to the document. Required if `Live App Type` is set to *salesforce_list*.

Input Parameter	Description
`Salesforce Org Name`	Optional. Salesforce org name used in the live app. Valid only if `Content Type` is set to *`liveapp`*. Only string values are supported. For example: *`Acme`*.
`Salesforce Record ID`	ID of the Salesforce record added to the document. Required if `Live App Type` is set to *`salesforce_record`*.

Store Output Values

Output Parameter	Description
`Document Title`	Title of the new document
`Document ID`	ID of the new document
`Document Link`	URL of the new document

Example: A sales manager wants to create a document at the end of each quarter to identify which accounts are at risk of attrition. You can create a flow that uses the Create Quip Document core action to create a document called Red Accounts. Add a Salesforce list view through the Salesforce List live app that shows all accounts in the red. Then add the document a Red Accounts folder.



Set Input Values

- *Document Title: Red Accounts
- Add Members by Email Address: person1@quip.com,person2@quip.com,pe — Include
- Company Link Access — Don't Include
- Content Type: liveapp — Include
- Document Content — Don't Include
- Document Type — Don't Include
- Live App Type: salesforce_list — Include
- Object Type: Account — Include
- Parent Folder URL: https://[quip_site_url]/folder/[red_account — Include
- Record Name — Don't Include
- Record Type — Don't Include
- Related List API Name — Don't Include
- Salesforce List View ID: 00BRM000006GMZz2AO — Include
- Salesforce Org Name: Salesforce — Include
- Salesforce Record ID — Don't Include

SEE ALSO:

Flow Elements

Flow Core Action for Quip: Create Quip Chat

Create a chat room, and send a message to its members.

In Flow Builder, add an Action element to your flow. In the Action field, enter `Quip`, and select **Create Quip Chat**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Message`	Chat message to get the chat room started. Valid values are: • String values

Input Parameter	Description
	- *@[Salesforce user ID]*—To @mention a Salesforce user in the chat room, enter the Salesforce ID. If the user's Salesforce email is connected to Quip, the user ID is replaced with a Quip user @mention. If not, the Salesforce ID is replaced with the user's Salesforce email. - *@[person1@quip.com]*—To @mention a user by email, enter this value. - *@Everyone*—To send a notification to all chat room members, enter this value.
`Add Members by Email Address`	Optional. A list of user emails separated by commas to add to the new chat room. Valid values are: - *person1@quip.com, person2@quip.com, person3@quip.com* - *person1@quip.com*
`Chat Title`	The title of the chat room. Only string values are supported.

Store Output Values

Output Parameter	Description
`Chat ID`	ID of the new chat room
`Chat Link`	URL of the new chat room
`Chat Title`	Title of the new chat room

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Create Quip Folder

Create a private folder, or add it to existing folders.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Create Quip Folder**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Folder Name`	Name of the new folder. Only string values are supported.
`Folder Color`	Optional. Color of the new folder. Valid values are: - *yellow* - *red* - *orange*

Input Parameter	Description
	- *green* - *blue* - *purple* - *manila* - *light red* - *light orange* - *light green* - *light blue* - *light purple*
`Parent Folder URL`	Optional. A list of folder URLs separated by commas to add the new folder to. By default, the folder is added to the user's Private folder in Quip. Valid values are: - *https://[quip_site_url]/folder/[folder1_name], https://[quip_site_url]/folder/[folder2_name]* - https://[quip_site_url]/folder/[folder_name] For example: *https://salesforce.quip.com/folder/account-plans*

Store Output Values

Output Parameter	Description
`Created Folder Title`	Title of the new folder
`Folder ID`	ID of the new folder

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Copy Quip Document

To use a document as a template, create a copy. By default, copied documents are added to the running user's Private folder in Quip.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Copy Quip Document**.

Important: Newly copied documents aren't automatically attached to the record. To attach the newly created document to the record and use Synced Sharing, use the Attach Document to Record action after the Copy Quip Document action and set the Salesforce Record ID to be the ID of the variable.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

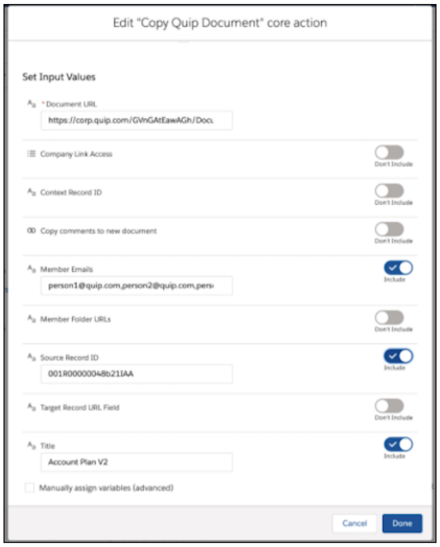
Input Parameter	Description
`Document URL`	The URL of the document that you want to copy. For example: *https://salesforce.quip.com/GVnGbtEasAGa*
`Company Link Access`	Optional. Link sharing settings for the copied document. By default, copied documents are set to edit-access. Valid values are: - *view*–To let users view the copied document, enter this value. - *edit*–To let users view and edit the copied document, enter this value. - *none*–To block user access to the copied document, enter this value.
`Context Record ID`	Optional. ID of the record that you want to update with the copied document's URL. Including the Context Record ID doesn't attach the document to a record. Valid only if the Quip Document component is set up on the record layout. The `Target Record URL Field` is required to use `Context Record ID`.
`Copy comments to new document`	Optional. This input determines whether to copy comments from the source document to the copied document. Valid values are: - *true*–To copy the source document's comments and annotations, enter this value. - *false*–To copy the source document's content without comments, enter this value.
`Member Emails`	Optional. A list of user emails separated by commas to add to the copied document. Valid values are: - *person1@quip.com, person2@quip.com, person3@quip.com* - *person1@quip.com*
`Member Folder URLs`	Optional. A list of folder URLs separated by commas to add the copied document to. Valid values are: - *https://[quip_site_url]/folder/[folder1_name], https://[quip_site_url]/folder/[folder2_name]* - https://[quip_site_url]/folder/[folder_name] For example: *https://salesforce.quip.com/folder/account-plans*
`Source Record ID`	Optional. ID of the record that you want to use in the place of mail merge syntax. For example, to replace the copied document's `Account.Name` merge field with the record's account name, enter the record ID.
`Target Record URL Field`	Optional. Reference to the URL field on a record used by the `Context Record ID`. This field is updated with the copied document URL and adds the copied document to the record's Quip Document component. Valid values are: - *API name of the field*–For example: *QuipDocumentURL__c* The `Context Record ID` is required to use the `Target Record URL Field`. Including the Target Record URL Field doesn't attach the newly created document to the record.
`Title`	Optional. The title of the copied document. Only string values are supported.

Store Output Values

Output Parameter	Description
`Created Document Title`	The title of the copied document.
`Document ID`	ID of the copied document.
`Document Link`	The URL of the copied document.

Example: Watch an Account Plan Automation Demo (2 minutes)

A sales rep wants to create an Account Plan and share it with the regional sales managers to close a large opportunity. You can create a flow that uses the Copy Quip Document core action to copy an Account Plan template when the Opportunity stage is set to Proposal/Quote. Configure the action to replace merge fields with data from the account, add the Account Plan to a folder, and share the folder with the regional sales managers.



Edit "Copy Quip Document" core action

Set Input Values

* Document URL: https://corp.quip.com/GVnGAtEawAGh/Docu

Company Link Access: Don't Include

Context Record ID: Don't Include

Copy comments to new document: Don't Include

Member Emails: person1@quip.com,person2@quip.com,pers — Include

Member Folder URLs: Don't Include

Source Record ID: 001R00000048b21IAA — Include

Target Record URL Field: Don't Include

Title: Account Plan V2 — Include

□ Manually assign variables (advanced)

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Copy Quip Content (Retired)

Copy content from a source slide deck, and paste it in a target slide.

Warning: Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

Drag a Core Action element onto the canvas. In the Core Action field, enter `Quip`, and select **Copy Quip Content**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

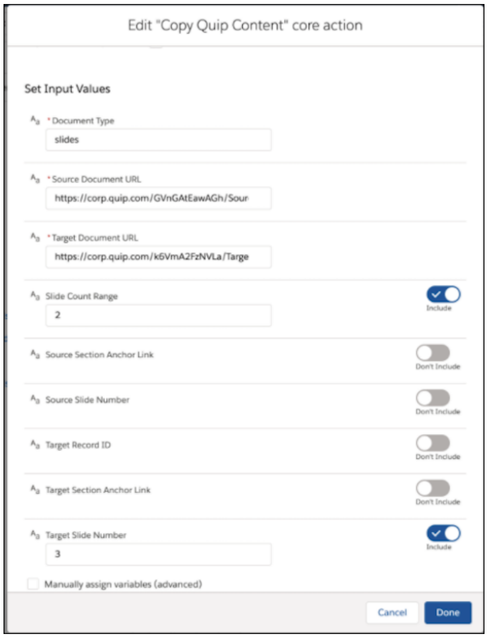
Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document Type`	Type of document you want to copy. Valid values are: - *`slides`*–To copy slides to another slide deck, enter this value.
`Source Document URL`	URL of the slide deck you want to copy content from.
`Target Document URL`	URL of the slide deck where you want to add copied content.
`Slide Count Range`	Optional. Number of slides to copy from the source slide deck. By default, `Slide Count Range` is set to *`1`*.
`Source Section Anchor Link`	URL of a section in the source slide deck that you want to copy content from.
`Source Slide Number`	Optional. The slide index to copy content from. For example, to copy content from the first slide of a deck, enter *`1`*.
`Target Record ID`	Optional. ID of the record you want to use in the place of mail merge syntax. For example, to replace the copied document's [[Account.Name] merge field with the record's account name, enter the record ID.
`Target Section Anchor Link`	URL of a section in the target slide deck where you want to paste copied content.
`Target Slide Number`	Optional. The slide index to copy content to. For example, to paste copied content to the first slide of a deck, enter *`1`*.

Store Output Values

Output Parameter	Description
`Document ID`	ID of the target slide deck content was copied to.
`Document Link`	URL of the target slide deck content was copied to.
`Created Document Title`	The title of the slide deck content was copied to. Only string values are supported.

Example: A sales rep wants to update a slide deck with the latest sales numbers to prepare for a customer pitch. The sales rep wants to use the slides from another deck that their manager keeps up to date with the latest numbers. You can create a flow that uses the Copy Quip Content core action to copy content slides 1 and 2 from their manager's slide deck and replace the content in slides 3 and 4 of the customer-facing slide deck.



Edit "Copy Quip Content" core action

Set Input Values

* Document Type: slides

* Source Document URL: https://corp.quip.com/GVnGAtEawAGh/Sour

* Target Document URL: https://corp.quip.com/k6VmA2FzNVLa/Targe

Slide Count Range: 2 — Include

Source Section Anchor Link — Don't Include

Source Slide Number — Don't Include

Target Record ID — Don't Include

Target Section Anchor Link — Don't Include

Target Slide Number: 3 — Include

□ Manually assign variables (advanced)

Flow Core Action for Quip: Copy with Live Paste

Copy content from a source document, and paste it with Live Paste in a new document. When the source content is updated, documents with the live pasted content stay up to date.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Quip`*, and select **Copy with Live Paste**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

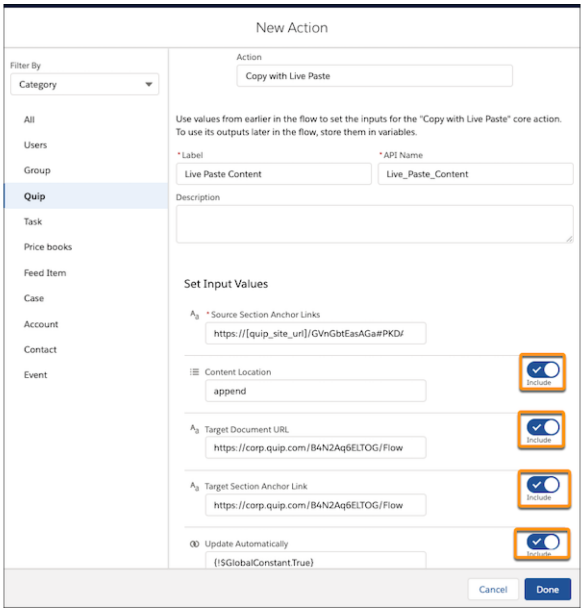
Input Parameter	Description
`Source Section Anchor Links`	URLs of the sections in the source document that you want to copy content from. Anchor links must be from the same document and separated with commas. Valid only if `Content Type` is set to anchor link.
`Content Location`	Optional. Location in the document where you want to live paste your copied content. Valid values are: • *`append`*—To live paste content to the end of the document, enter this value. • *`prepend`*—To live paste content to the beginning of the document, enter this value. • *`after_section`*—To live paste content after a designated section, enter this value. Valid only if `Target Section Anchor Link` is specified. • *`before_section`*—To live paste content before a designated section, enter this value. Valid only if `Target Section Anchor Link` is specified. • *`replace_section`*—To replace an existing section with live pasted content, enter this value. Valid only if `Target Section Anchor Link` is specified.

Input Parameter	Description
	- *after_document_range*–To live paste content after a named document range, enter this value. Valid only if Target Document Range Heading Text is specified. - *before_document_range*–To live paste content before a named document range, enter this value. Valid only if Target Document Range Heading Text is specified. - *replace_document_range*–To replace a named document range with live pasted content, enter this value. Valid only if Target Document Range Heading Text is specified. By default, Content Location is set to *append*.
Content Type	Type of content that you want to copy. Valid values are: - *anchor_link*–To copy content based on a section anchor link URL, enter this value. - *document_range*–To copy content from a template based on a document range name, enter this value.
Source Document Range Heading	Heading text from the document range that you want to copy. Valid only if Content Type is set to *document_range*.
Target Document URL	URL of the document that you want to copy live pasted content to.
Target Section Anchor Link	Optional. URL of the section in the target document where you want to copy live pasted content to.
Target Document Range Heading Text	Heading text from the document range where you want to live paste your copied content.
Update Automatically	Optional. Automatically update the target document when the source content is updated and Live Paste is on. Valid values are: - *true* - *false* By default, Update Automatically is set to *false*.

Store Output Values

Output Parameter	Description
Document ID	ID of the document where live pasted content was added.
Document Link	URL of the document where live pasted content was added.
Document Title	Title of the document where live pasted content was added.

Example: A sales manager wants to add instructions to all Account Plans to teach their sales reps what to do next. But the sales manager doesn't want to update each one individually. You can create a flow that uses the Copy with Live Paste core action to add the updated instructions to the end of the Account Plan.



New Action

Filter By: Category

Action: Copy with Live Paste

Use values from earlier in the flow to set the inputs for the "Copy with Live Paste" core action. To use its outputs later in the flow, store them in variables.

*Label: Live Paste Content

*API Name: Live_Paste_Content

Description:

Set Input Values

*Source Section Anchor Links: https://[quip_site_url]/GVnGbtEasAGa#PKD/

Content Location: append

Target Document URL: https://corp.quip.com/B4N2Aq6ELTOG/Flow

Target Section Anchor Link: https://corp.quip.com/B4N2Aq6ELTOG/Flow

Update Automatically: {!$GlobalConstant.True}

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Attach Quip Document to Record

EDITIONS

Available in: **Lightning Experience**

Attach a document to a Salesforce record. Linked documents show up in the Quip Associated Documents component.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Attach Quip Document to Record**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document URL`	The URL of the document that you want to attach to a Salesforce record. For example: *https://salesforce.quip.com/GVnGbtEasAGa*
`Salesforce Record ID`	ID of the Salesforce record that you want to attach your document to.

Store Output Values

Output Parameter	Description
`Content Document Link ID`	The ID of the link between the document and the record where it's attached. The Attach Quip Document to Record flow action creates a Content Document object that references the document. It also creates a Content Document Link object that maps the record to the Content Document object.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Edit Quip Document

Edit content in a document, spreadsheet, or slide. Add or replace content based on a document section.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Quip`*, and select **Edit Quip Document**.

Warning: Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Content Location`	Location in the document where you want to add content. Valid values are: • *`append`*–To add content to the end of the document, enter this value. • *`prepend`*–To add content to the beginning of the document, enter this value. • *`after_section`*–To add content after a designated section, enter this value. Valid only if `Section Anchor Link` is specified. • *`before_section`*–To add content before a designated section, enter this value. Valid only if `Section Anchor Link` or `Section Match Type` is specified. • *`replace_section`*–To replace an existing section with new content, enter this value. Valid only if `Section Anchor Link` or `Section Match Type` is specified. • *`after_document_range`*–To add content to a template after a document range, enter this value. Valid only if `Document Range Heading` is specified. • *`before_document_range`*–To add content to a template before a document range, enter this value. Valid only if `Document Range Heading` is specified. • *`replace_document_range`*–To replace existing content based on a document range, enter this value. Valid only if `Document Range Heading` is specified.

Input Parameter	Description
`Document URL`	URL of the document that you want to edit. For example: *`https://salesforce.quip.com/GVnGbtEasAGa`*
`Edit Document Type`	Type of document that you want to edit. Valid values are: - *`document`*—To edit a document, enter this value. - *`spreadsheet`*—To edit a spreadsheet, enter this value.
`Content`	Optional. Content added to the document that you want to edit. Valid only if `Content Format` is set to *`html`*. Valid values are: - String values - *`@[Salesforce user ID]`*—To @mention a Salesforce user in the document, enter the Salesforce ID. If the user's Salesforce email is connected to Quip, the user ID is replaced with a Quip user @mention. If not, the Salesforce ID is replaced with the user's Salesforce email. - *`@[person1@quip.com]`*—To @mention a user by email, enter this value. - *`@Everyone`*—To send a notification to everyone in the document, enter this value.
`Content Format`	Optional. Format of content added to the document. By default, content format is set to html. Valid values are: - *`html`*—To format text added to `Document Content` with html, enter this value. - *`liveapp`*—To add a live app to your document, enter this value. Only valid if `Document Type` is set to *`document`*.
`Disable Extra Lines in Quip`	Optional. Boolean parameter that prevents Quip from inserting an extra line between paragraphs. The default is *`false`*, meaning that by default extra lines *are* inserted.
`Document Range Heading`	Optional. Heading text that marks the start of the document range.
`Element Type`	Optional. The type of spreadsheet element to edit. Only valid if `Document Type` is set to *`spreadsheet`*. Valid values are: - *`row`*—To edit a spreadsheet row, enter this value. - *`column`*—To edit a spreadsheet column, enter this value.
`Image Number`	Optional. Image index of an image on a slide. Only valid if `Document Type` is set to *`slides`*. Valid values are: - *`Image integers`*—Integers represent an image index on a slide. Images are ordered from top to bottom. Images closer to the top of a slide have an image integer of 1. Images closer to the bottom have the biggest integers. If there are multiple images on a slide with the same vertical positions, the image numbers are ordered from left to right. If there are multiple images on a slide with the same horizontal and vertical positions, the image that is behind the other one has an image number of *`1`*. The image in front has an image number of *`2`*. Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

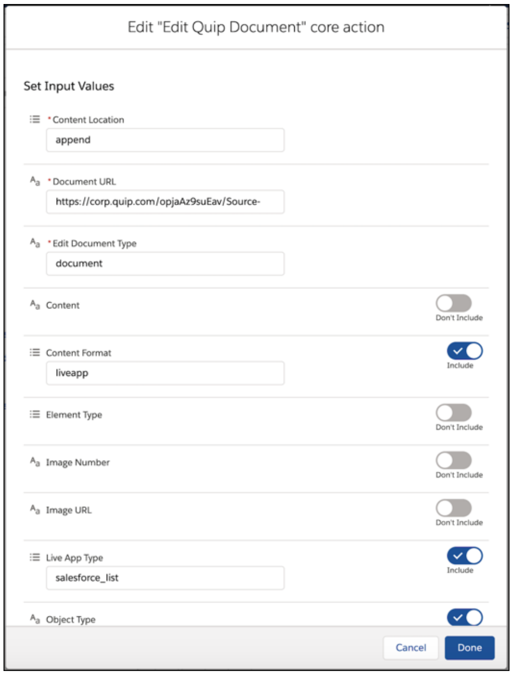
Input Parameter	Description
`Image URL`	Optional. The URL of an image in a slide deck. Only valid if `Document Type` is set to *`slides`*. Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More
`Live App Type`	Type of live app added to the document. Required if `Content Format` is set to *`liveapp`*. Only documents support live apps. Valid values are: - *`salesforce_record`*–To add the Salesforce Record live app to the document, enter this value. - *`salesforce_list`*–To add the Salesforce List live app to the document, enter this value.
`Object Type`	Type of object used by the Salesforce List live app. Required if `Live App Type` is set to *`salesforce_list`*. Only string values are supported. For example: *`Account`*, *`Opportunity`*, or *`CustomObject__c`*.
`Record Name`	Optional. Name of the record added to the document through the Salesforce Record live app. Valid only if the `Live App Type` is set to *`salesforce_record`*. Only string values are supported.
`Record Type`	Optional. Type of object used by the Salesforce Record live app. Valid only if `Live App Type` is set to *`salesforce_record`*. Only string values are supported. For example: *`Account`*, *`Opportunity`*, or *`CustomObject__c`*.
`Salesforce List View ID`	Optional. ID of the Salesforce list view added to the document. Valid only if `Live App Type` is set to *`salesforce_list`*.
`Salesforce Org Name`	Optional. Salesforce org name used in the live app. Valid only if `Content Format` is set to *`liveapp`*. Only string values are supported. For example: *`Acme`*.
`Salesforce Record ID`	Optional. ID of the Salesforce record added to the document. Valid only if `Live App Type` is set to *`salesforce_record`*.
`Section Anchor Link`	URL of a section in the document where you want to add or replace content. Valid only if `Content Location` is set to *`before_section`*, *`after_section`*, or *`replace_section`*. For example: *`https://[quip_site_url]/GVnGAtEawAGh/Source-Slide#JUJACAuc0ps`*, where Source-Slide#JUJACAuc0ps is a specific slide in the slide deck.
`Section Match Type`	Placement of keywords used to identify the section where you want to add or replace content. Valid only if `Content Location` is set to *`before_section`*, *`after_section`*, or *`replace_section`*, and the `Document Type` is *`document`*. Keywords aren't case-sensitive and ignore HTML tags. Valid values are: - *`prefix`*–To find a keyword in a document based on the first part of a word, enter this value. For example, *`hello`* is the prefix for *`helloworld`*. - *`suffix`*–To find a keyword in a document based on the end of a word, enter this value. For example, *`world`* is the suffix for *`helloworld`*.

Input Parameter	Description
`Section Style`	Format of the document section where you want to add or replace content. Required if `Section Type` is set to *`textmatch`*. Valid values are: - *`paragraph`*—To find a keyword in a paragraph, enter this value. - *`heading`*—To find a keyword in a heading, enter this value. - *`list`*—To find a keyword in a list, enter this value.
`Section Text`	Keywords to identify the section where you want to add or replace content. Required if `Section Match Type` is specified. Only string values are supported.
`Section Type`	Optional. Determines how a section is edited. Valid values are: - *`anchorlink`*—To edit a document section based on its anchor link, enter this value. Valid only if `Section Anchor Link` is set up. - *`textmatch`*—To edit a document section based on a keyword, enter this value.
`Slide Layout`	Optional. The slide element to edit. Only valid if `Document Type` is set to *`slides`*. Valid values are: - *`single_column`*—To edit a slide column, enter this value. - *`image`*—To edit a slide image, enter this value. Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More
`Slide Number`	Optional. The slide index to edit. Only valid if `Document Type` is set to *`slides`*. For example, to edit the first slide of a deck, enter *`1`*. Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

Store Output Values

Output Parameter	Description
`Document ID`	ID of the edited document
`Document Link`	URL of the edited document
`Document Title`	Title of the edited document

Example: A sales rep wants to update their Account Plan and add a list view of open opportunities related to the account. You can create a flow that uses the Edit Document core action to add the Salesforce List live app to the end of the Account Plan.



Edit "Edit Quip Document" core action

Set Input Values

*Content Location
append

*Document URL
https://corp.quip.com/opjaAz9suEav/Source-

*Edit Document Type
document

Content — Don't Include

Content Format — Include
liveapp

Element Type — Don't Include

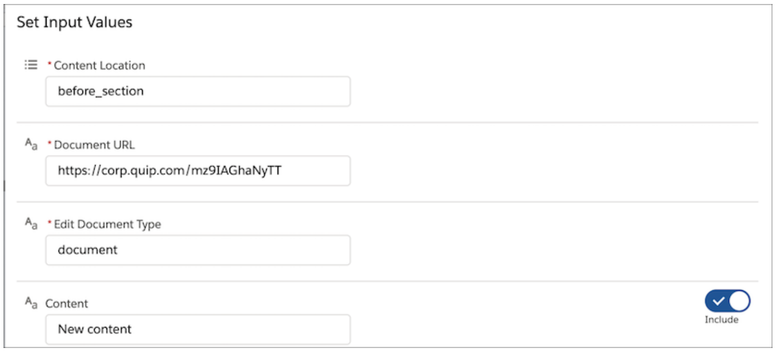
Image Number — Don't Include

Image URL — Don't Include

Live App Type — Include
salesforce_list

Object Type

Example: A service manager wants to update an Account Plan with an account history. The account is up for renewal and the service manager wants to make sure that the sales rep has the necessary background on past cases before contacting the customer. You can create a flow that uses the Edit Document core action to add an account history to an Account Plan and place it before the placeholder lorem.



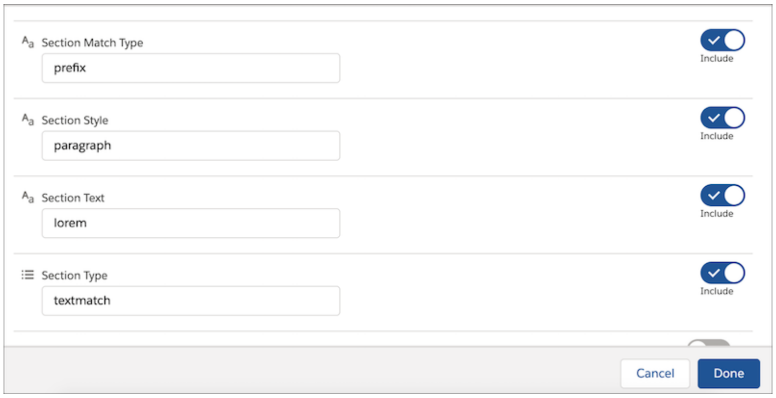
Set Input Values

*Content Location
before_section

*Document URL
https://corp.quip.com/mz9IAGhaNyTT

*Edit Document Type
document

Content — Include
New content



Section Match Type: prefix

Section Style: paragraph

Section Text: lorem

Section Type: textmatch

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Lock Quip Document

To mark a document as complete, lock document edits.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Lock Quip Document**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document URL`	URL of the document that you want to lock.
`Lock Operation`	Action of locking or unlocking the document. Only a user with full access to the document can lock or unlock it. Valid values are: - *lock*—To lock edits to the document, enter this value. - *unlock*—To let users edit a locked document, enter this value.

Store Output Values

Output Parameter	Description
`Document Lock Operation Result`	Success marker of whether the document was locked or unlocked. Possible results are `true` or `false`.

Example: A sales manager wants to lock edits to their reps' Account Plans after a deal is closed. You can create a flow that uses the Lock Quip Document core action to lock edits to Account Plans when the Opportunity stage is set to Closed Won.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Lock Quip Section

EDITIONS

Available in: **Lightning Experience**

To mark sections of a document as complete or to keep them safe from accidental edits, lock them.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Lock Quip Section**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
Section Anchor Link	Anchor link URL of the document section that you want to lock.
Lock Operation	The action of locking or unlocking a document section. Only a user with full access to the document can lock or unlock its sections. Valid values are: • *lock*—To lock edits to the document, enter this value. • *unlock*—To let users edit a locked document, enter this value.

Store Output Values

Output Parameter	Description
Section Lock Operation Result	Success marker of whether the section was locked or unlocked. Possible results are true or false.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Export Quip Document to PDF

EDITIONS

Available in: **Lightning Experience**

To mark a document as complete and to keep a copy for your records, export it as a PDF. You can attach the exported PDF to a document or to a Salesforce record.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Source Document URL`	URL of the document you want to export to PDF. Valid values are: - String values - Field references—To pull a document housed in a custom URL field, enter the field API name.
`Sheet Name`	Name of the sheet in a spreadsheet that you want to export to a PDF. Valid only for spreadsheet files. If no sheet name is entered, defaults to the first sheet.
`Target Document URL`	Optional. URL of the document where you want to attach the created PDF. The PDF is added to the end of the document.
`Salesforce Organization ID`	ID of the Salesforce org where you want to attach your new PDF. You can use the same Salesforce org ID of the triggering record, or attach the PDF to a different org. Valid only if `Target Record ID` is specified.
`Target Record ID`	Optional. ID of the Salesforce record you want to attach your document to. PDFs attached to a record are added to the record's Files component and Notes and Attachments component, and are visible to any user with access to the record. Valid values are: - Alphanumeric series of numbers and letters for a specific Salesforce record. - *`{!$Record.Id}`*To attach the PDF to the same record that triggered the flow, enter this value.

Store Output Values

Output Parameter	Description
`Request ID`	ID to check the status of the PDF export.
`Status message`	Error message that explains why the document wasn't exported to a PDF.
`Status of the request`	Success marker of whether the document was exported to a PDF. Can be `success`, `failure`, or `pending`.

SEE ALSO:

Flow Elements

Flow Core Action for Quip: Send Message in Quip Chat

EDITIONS

Available in: **Lightning Experience**

Send a message in a chat room.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Quip`*, and select **Send Message in Quip Chat**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Chat URL`	URL of the chat room where you want your message to appear.
`Message`	Chat message sent to the chat room. Valid values are: • String values • *@[Salesforce user ID]*—To @mention a Salesforce user in the chat room, enter the Salesforce ID. If the user's Salesforce email is connected to Quip, the user ID is replaced with a Quip user @mention. If not, the Salesforce ID is replaced with the user's Salesforce email. • *@[person1@quip.com]*—To @mention a user by email, enter this value. • *@Everyone*—To send a notification to all chat room members, enter this value.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Send a Message in a Document

EDITIONS

Available in: **Lightning Experience**

Add a message to the conversations pane of a document.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Send Message in Document**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document URL`	URL of the document where you want to add a comment.
`Message`	Message added to the document. Valid values are: • String values • *@[Salesforce user ID]*—To @mention a Salesforce user in the document, enter the Salesforce ID. If the user's Salesforce email is connected to Quip, the user ID is replaced with a Quip user @mention. If not, the Salesforce ID is replaced with the user's Salesforce email. • *@[person1@quip.com]*—To @mention a user by email, enter this value. • *@Everyone*—To send a notification to everyone in the document, enter this value.
`Section Anchor Link`	Optional. URL of a section in the document where you want the message to appear.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Add Quip Document to Folder

EDITIONS

Available in: **Lightning Experience**

Add a document to a folder to organize and share your documents.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Add Quip Document to Folder**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
Document URL	URL of the document that you want to add to a folder.
Folder URL	URL of the folder where you want to add the document.

Store Output Values

Output Parameter	Description
Document ID	ID of the added document
Document Link	URL of the document added to the folder
Document Title	Title of the added document

Example: Watch an Account Plan Automation Demo (2 minutes)

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Add Members to Document

EDITIONS

Available in: **Lightning Experience**

Add members with different levels of access to a document.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Add Members to Document**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
Document URL	URL of the document that you want to add members to.
Add Comment-Access Members by Email Address	Optional. A list of user emails separated by commas that can view and comment on the document. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com*

Input Parameter	Description
	• *person1@quip.com*
`Add Edit-Access Member by Email Address`	Optional. A list of user emails separated by commas that can view, comment on, and edit the document. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com* • *person1@quip.com*
`Add Full-Access Members by Email Address`	Optional. A list of user emails separated by commas that can view, comment on, edit, and share the document. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com* • *person1@quip.com*
`Add View-Access Members by Email Address`	Optional. A list of user emails separated by commas that can view the document. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com* • *person1@quip.com*

Store Output Values

Output Parameter	Description
`Document ID`	ID of the document
`Document Link`	URL of the document
`Document Title`	Title of the document

Example: A service manager wants to add Tier 3 service reps to a Case Swarm document to solve a customer case. The service manager also wants to keep sales reps with open opportunities related to the account up to date. You can create a flow that uses the Add Members to Document core action to add Tier 3 reps to the Case Swarm document and grant the service reps edit-access to the document. Then you can add the sales reps with open opportunities to the document with comment-access so that they can see what's happening and ask questions.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Add Members to Quip Chat

Add users to a chat room.

In Flow Builder, add an Action element to your flow. In the Action field, enter `Quip`, and select **Add Members to Quip Chat**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Add Members by Email Address`	A list of user emails separated by commas to add to the chat room. Valid values are: - *person1@quip.com, person2@quip.com, person3@quip.com* - *person1@quip.com*
`Chat URL`	URL of the chat room that you want to add members to.

Store Output Values

Output Parameter	Description
`Chat ID`	ID of the chat room
`Chat Link`	URL of the chat room
`Chat Title`	Title of the chat room

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Remove Quip Document from Folder

Remove a document from a folder. Make a shared document private again.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Quip`*, and select **Remove Quip Document from Folder**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document URL`	URL of the document that you want to remove from a folder.
`Folder URL`	URL of the folder that you want to remove the document from.

Store Output Values

Output Parameter	Description
`Document ID`	ID of the removed document
`Document Link`	URL of the removed document

Output Parameter	Description
`Document Title`	Title of the removed document

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Remove Members from Quip Document

EDITIONS

Available in: **Lightning Experience**

To rescind access to a document for certain users, remove them from the document.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Quip`*, and select **Remove Members from Quip Document**.

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Document URL`	URL of the document that you want to remove members from.
`Remove Members by Email Address`	A list of user emails separated by commas to remove from the document. Valid values are: • *`person1@quip.com, person2@quip.com, person3@quip.com`* • *`person1@quip.com`*

Store Output Values

Output Parameter	Description
`Document ID`	ID of the document
`Document Link`	URL of the document
`Document Title`	Title of the document

Example: A service manager previously added Tier 3 service reps and sales reps to a Case Swarm document to solve a customer case. Now that the case is closed, the service manager wants to remove user access to the document to preserve its integrity. You can create a flow that uses the Remove Members from Quip Document core action to remove Tier 3 reps and sales reps from the document.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Quip: Remove Members from Quip Chat

Remove users from a chat room.

In Flow Builder, add an Action element to your flow. In the Action field, enter *Quip*, and select **Remove Members from Quip Chat**.

EDITIONS

Available in: **Lightning Experience**

Set Input Values

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
Remove Members by Email Address	A list of user emails separated by commas to remove from the chat room. Valid values are: • *person1@quip.com, person2@quip.com, person3@quip.com* • *person1@quip.com*
Chat URL	URL of the chat room that you want to remove members from.

Store Output Values

Output Parameter	Description
Chat ID	ID of the chat room
Chat Link	URL of the chat room
Chat Title	Title of the chat room

SEE ALSO:

Add and Edit Elements

B2B Commerce Checkout Flow Core Actions

The B2B Commerce Checkout Flow provides several core actions for implementing a successful checkout process within your Commerce org. To add one of these actions to your flow, add an Action element. Then select the **B2B Commerce** category, and search for the appropriate action.

These actions use Apex-defined input and output variables that map to input and output classes in the Apex `ConnectApi` namespace. For more information on using Apex-defined variables in flows, see Considerations for the Apex-Defined Data Type on page 260.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Flow Core Action for Checkout Flow: Activate Order

Activates a Salesforce standard draft order.

Flow Core Action for Checkout Flow: Calculate Cart Promotions

Request a full calculation of all line items in the cart that have a promotion.

Flow Core Action for Checkout Flow: Calculate Cart Shipment Costs

Request the shipping costs of all line items within the cart.

Flow Core Action for Checkout Flow: Calculate Cart Taxes

Request a tax cost calculation for all line items within the cart.

Flow Core Action for Checkout Flow: Cancel Cart Async Operation

Cancel the current checkout so the user can return to an unlocked cart. This operation automatically executes when cart changes are invoked, such as add to cart.

Flow Core Action for Checkout Flow: Cart to Order

Create a Salesforce standard order in draft mode.

Flow Core Action for Checkout Flow: Check Cart Inventory

Request a full inventory availability check of all line items in the cart.

Flow Core Action for Checkout Flow: Checkout Session Action

Get or create a checkout session, and return the ID of the session to the caller.

Flow Core Action for Checkout Flow: Price Cart

Request a reprice of all line items in a cart.

Flow Core Action for Checkout Flow: Update Checkout Session Action

The Update Checkout Session action updates the checkout session state if the current state matches the expected state. This action provides consistency during checkout handling and guarantees that if two browsers attempt to update the state, one succeeds and the other fails validation.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Activate Order

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Activates a Salesforce standard draft order.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Activate Order**. To access this action from the API, use the name `activateOrderAction`.

Set Input Values:

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.
`orderStatus`	A DynamicEnum with the OrderTypeEnum value.

Store Output Values:

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions:

Error Condition	Description
Invalid CartId Input	The cart ID value isn’t accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can’t Invoke Action	The caller doesn’t have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn’t Have Access to the Cart	The buyer doesn’t own the cart, doesn’t have read access to the cart, or the cart isn’t shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
The Cart Isn’t in the Checkout State	The cart status isn’t in the Checkout state and can’t continue. Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn’t Valid or Accessible	The effective account listed isn’t valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn’t a Member of the Store	The user isn’t a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
CartValidationOutput Object Has Errors	The CartValidationOutput object has errors that are related to the cart and must be cleared before order activation. Error Code: INVALID_INPUT HTTP Status Code: 403
CheckoutSession Isn’t in the Activate State	Checkout Session state must be in the Activate state for the order activation to go forward. Error Code: INVALID_INPUT HTTP Status Code: 403

Error Condition	Description
CheckoutSession Is in the Processing State	CheckoutSession can't be in the Processing state. Processing indicates a separate integration is already running. Error Code: INVALID_INPUT HTTP Status Code: 403
Invalid Order Status Input	The Order status input that is passed to the API must be the `ACTIVATE` status code. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
OrderSummary Wasn't Created	The OrderSummary wasn't created due to an internal error. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 403

Usage:

To use the B2B Commerce Activate Order action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The buyer has edit access to the cart.
- An order in `Draft` status is generated from the cartToOrder action, and the `orderId` is in CartCheckoutSession.
- The cart status is `CartStatusEnum.CHECKOUT`.
- `Session.IsProcessing` isn't set to `False`.
- `Session.NextState` is set to `activateOrder`.
- `Session.State` can't be empty.
- `backgroundOperationId` can't be `New` or `Running`.

When the Activate Order action runs, these API interactions occur.

1. The order item is activated, making it read-only.
2. The order summary is generated asynchronously.
3. The CheckoutSession is archived.
4. The cart is archived.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Calculate Cart Promotions

Request a full calculation of all line items in the cart that have a promotion.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Calculate Cart Promotions**. To access this action from the API, use the name `calcCartPromotionsAction`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that you want to reprice.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn't accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn't Have Access to the Cart	The buyer doesn't own the cart, doesn't have read access to the cart, or the cart isn't shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Isn't in the Checkout State	The cart status isn't in the Checkout state and can't continue. Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running.

Error Condition	Description
	Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500

Usage

This action is executed asynchronously using the pricing service configured in StoreIntegratedService.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Calculate Cart Shipment Costs

Request the shipping costs of all line items within the cart.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Calculate Cart Shipment Costs**. To access this action from the API, use the name `calcCartShipmentAction`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn't accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn't Have Access to the Cart	The buyer doesn't own the cart, doesn't have read access to the cart, or the cart isn't shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Isn't in the Checkout State	The cart status isn't in the Checkout state and can't continue. Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500

Usage

This action is executed asynchronously using the pricing service configured in StoreIntegratedService.

To use the B2B Commerce Check Cart Shipment Cost action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The buyer has edit access to the cart.

- The cart status isn't `Closed`.
- `Session.IsProcessing` isn't set to `False`.
- `Session.NextState` is set to `DeliveryMethod`.
- `Session.State` can't be empty.
- `backgroundOperationId` can't be `New` or `Running`.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Calculate Cart Taxes

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Request a tax cost calculation for all line items within the cart.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Calculate Cart Taxes**. To access this action from the API, use the name `calcCartTaxesAction`.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn't accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400

Error Condition	Description
User Doesn't Have Access to the Cart	The buyer doesn't own the cart, doesn't have read access to the cart, or the cart isn't shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Isn't in the Checkout State	The cart status isn't in the Checkout state and can't continue. Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500

Usage

This action is executed asynchronously using the pricing service configured in StoreIntegratedService.

To use the B2B Commerce Calculate Cart Taxes action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The buyer has edit access to the cart.
- The cart status isn't `Closed`.
- `Session.IsProcessing` isn't set to `False`.
- `Session.NextState` is set to `ComputeTaxes`.
- `Session.State` can't be empty.
- `backgroundOperationId` can't be `New` or `Running`.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Cancel Cart Async Operation

Cancel the current checkout so the user can return to an unlocked cart. This operation automatically executes when cart changes are invoked, such as add to cart.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Cancel Cart Async Operation**. To access this action from the API, use the name `cancelCartAsyncOperation`.

Set Input Values:

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Error Conditions:

Error Condition	Description
Cancel a Closed Cart	A cart can’t be canceled after it’s closed. Error Code: INVALID_OPERATION HTTP Status Code: 403

Usage:

To use the B2B Commerce Cancel Cart Async Operation, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The cart status can’t be `Closed`.
- There’s an active CartCheckoutSession associated with the cart.
- There’s a valid `BackgroundOperationId`.

When the Cancel Cart Async Operation runs, these API interactions occur.

1. The background operation is marked as canceled.
2. The cart transitions to the Active state, unlocking the cart for more updates.
3. If it exists, the CartCheckoutSession object is archived.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Cart to Order

Create a Salesforce standard order in draft mode.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Cart To Order** . To access this action from the API, use the name `cartToOrderAction`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.
`runAsync`	Execute the Cart to Order action asynchronously.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn’t accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can’t Invoke Action	The caller doesn’t have the appropriate permissions to call the action, including the MAD or B2BCommerceIntegrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn’t Have Access to the Cart	The buyer doesn’t own the cart, doesn’t have read access to the cart, or the cart isn’t shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Isn’t in the Checkout State	The cart status isn’t in the Checkout state and can’t continue. Error Code: INVALID_OPERATION HTTP Status Code: 403

Error Condition	Description
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500

Usage

To use the B2B Commerce Cart to Order action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The buyer has edit access to the cart.
- The cart status is `CartStatusEnum.CHECKOUT`.
- `Session.IsProcessing` isn't set to `False`.
- `Session.NextState` is set to `cartToOrder`.
- `Session.State` can't be empty.
- `backgroundOperationId` can't be `New` or `Running`.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Check Cart Inventory

Request a full inventory availability check of all line items in the cart.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Check Cart Inventory**. To access this action from the API, use the name `checkCartInventoryAction`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn't accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn't Have Access to the Cart	The buyer doesn't own the cart, doesn't have read access to the cart, or the cart isn't shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
The Cart Isn't in the Checkout State	The cart status isn't in the Checkout state and can't continue. Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY

Error Condition	Description
	HTTP Status Code: 500

Usage

This action is executed asynchronously using the pricing service configured in StoreIntegratedService.

To use the B2B Commerce Check Cart Inventory action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The buyer has edit access to the cart.
- The cart status isn't `Closed`.
- `Session.IsProcessing` isn't set to `False`.
- `Session.NextState` is set to `CheckInventory`.
- `Session.State` can't be empty.
- `backgroundOperationId` can't be `New` or `Running`.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Checkout Session Action

Get or create a checkout session, and return the ID of the session to the caller.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Checkout Session Action**. To access this action from the API, use the name `checkoutSessionAction`.

Set Input Values:

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart that transitions into a checkout.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Store Output Values:

Output Parameter	Description
`requestId`	The ID of the request that processes and then either creates or returns the Checkout Session.

Error Conditions:

Error Condition	Description
Invalid CartId Input	The cart ID value isn't accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn't Have Access to the Effective Account	The user doesn't have access to the effective account either because it isn't a buyer account or they don't have permission to buy for the account. Error Code: BAD_REQUEST HTTP Status Code: 400
User Account Isn't Associated With the Store	The current logged-in buyer account isn't associated with the store and therefore isn't a store member. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Is Already in Progress	The requested cart is already being processed. Error Code: BAD_REQUEST HTTP Status Code: 400
The Session Wasn't Created	The session wasn't created due to an internal service error. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500

Usage:

To use the B2B Commerce Checkout Session Action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The cart status is set to `Active` or `Checkout`.
- The cart must not have any current, active session.

When the Checkout Session Action runs, these API interactions occur.

1. The cart transitions to the Checkout state, preventing more updates to the cart.
2. If it doesn't exist already, the CartCheckoutSession object is created.

3. All errors that are mapped to the input `cartId`, on the CartValidationOutput object, are cleared.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Price Cart

Request a reprice of all line items in a cart.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Price Cart**. To access this action from the API, use the name `priceCart`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`cartId`	The ID of the cart containing the items that you want to reprice.

Store Output Values

Output Parameter	Description
`backgroundOperationId`	The ID of the background operation that processes the total price of all items in the cart.

Error Conditions

Error Condition	Description
Invalid CartId Input	The cart ID value isn’t accepted. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 500
User Can’t Invoke Action	The caller doesn’t have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
User Doesn’t Have Access to the Cart	The buyer doesn’t own the cart, doesn’t have read access to the cart, or the cart isn’t shared with the buyer. Error Code: BAD_REQUEST HTTP Status Code: 400
Cart Isn’t in the Checkout State	The cart status isn’t in the Checkout state and can’t continue.

Error Condition	Description
	Error Code: INVALID_OPERATION HTTP Status Code: 403
Integration Is Already in Progress	Only one integration can be processed at a time. This error indicates when an integration is already running. Error Code: ALREADY_IN_PROCESS HTTP Status Code: 400
Account Associated With the Cart Isn't Valid or Accessible	The effective account listed isn't valid. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500
User Isn't a Member of the Store	The user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 500

Usage

This action is executed asynchronously using the pricing service configured in StoreIntegratedService.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Checkout Flow: Update Checkout Session Action

The Update Checkout Session action updates the checkout session state if the current state matches the expected state. This action provides consistency during checkout handling and guarantees that if two browsers attempt to update the state, one succeeds and the other fails validation.

In Flow Builder, add an Action element to your flow. Select the **B2B Commerce** category, and search for **Update Checkout Session Action**. To access this action from the API, use the name `updateCheckoutSessionAction`.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions with B2B Commerce

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`checkoutSessionId`	The ID of the checkout session.
`nextState`	The state the session moves to after it completes the tasks included in the current state. This input is an Apex-defined variable of enum `CheckoutStateEnum`.

Input Parameter	Description
`expCurrentState`	The current state of the session. This input is an Apex-defined variable of enum `CheckoutStateEnum`.

Store Output Values

Output Parameter	Description
`requestId`	The ID of the request that processes and then either creates or returns the Checkout Session.

Error Conditions

Error Condition	Description
Expected Validation Error	The current state of the checkout session, identified by `checkoutSessionId` parameter, doesn't match the `expectedState` parameter, so the validation fails. HTTP Status Code: 4XX
Invalid Checkout Session ID	Invalid input for the Checkout Session ID. Error Code: UNKNOWN_EXCEPTION HTTP Status Code: 403
Invalid Session or Inadequate User Access	Either the session doesn't exist or the user doesn't have the required permissions. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400
User Can't Invoke Action	The caller doesn't have the appropriate permissions to call the action, including the MAD or B2B Commerce Integrator user perms. Error Code: BAD_REQUEST HTTP Status Code: 400
Account Associated With Cart Is Invalid or Inaccessible	The effective account associated with the cart isn't a valid account. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400
User Isn't a Member of the Store	The buyer user isn't a member of the store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400

Usage

To use the B2B Commerce Update Checkout Session Action, these requirements apply.

- The user has the appropriate permissions to invoke the action.
- The effective account is valid.
- The buyer account is a member of the store.
- The cart status isn't set to `Closed` or `Processing`.
- The `CartcheckoutSession.IsProcessing` field is `false`.

SEE ALSO:

Add and Edit Elements

Commerce Checkout Flow Core Actions

The Commerce Checkout Flow provides several core actions for implementing a successful checkout process within your Commerce org. To add one of these actions to your flow, add an Action element. Then select the **Commerce** category, and search for the appropriate action. Cart actions aren't available in flows for B2B stores built on an Aura template.

EDITIONS

Available in: Lightning Experience

Available in: **Performance**, **Professional**, and **Unlimited** Editions

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

These actions use Apex-defined input and output variables that map to input and output classes in the Apex `ConnectApi` namespace. For more information on using Apex-defined variables in flows, see Considerations for the Apex-Defined Data Type on page 260.

Flow Core Action for Commerce Checkout Flow: Add Cart Item
Add an item to a cart.

Flow Core Action for Commerce Checkout Flow: Create Cart
Create a cart.

Flow Core Action for Commerce Checkout Flow: Delete Cart
Delete a cart.

Flow Core Action for Commerce Checkout Flow: Get Cart Items
Get items in a cart.

Flow Core Action for Commerce Checkout Flow: Get Cart Promotions
Get promotions associated with a cart.

Flow Core Action for Commerce Checkout Flow: Add Cart Item

Add an item to a cart.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

In Flow Builder, add an Action element to your flow. Select the **Commerce** category, and search for **Add Cart Item**. To access this action from REST API, use the name `addCartItem`.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cart State or ID`	ID of the cart or state of the cart to add an item to. Valid state values are: *`active`* and *`current`*. A current cart is not closed or pending deletion.
`effectiveAccountId`	(Optional) ID of the buyer account or guest buyer profile for which the request is made. If unspecified, the default value is determined from context.
`Web Store ID`	The ID of the web store.
`Cart Item Input`	This input is an Apex-defined variable of class ConnectApi.CartItemInput, which includes these fields: • `productId` — ID of the product. • `quantity` — Quantity of the cart item. Use a value that can be converted to BigDecimal. • `type` — Type of the cart item. The only valid value is *`Product`*.

Store Output Values

Use output values later in the flow. The values are assigned when the item is created.

Output Parameter	Description
`Added Cart Item`	This output is an Apex-defined variable of class ConnectApi.CartItem, which includes these fields: • `itemizedAdjustmentAmount` — Total itemized adjustment amount for the item, including promotions and excluding taxes. • `listPrice` — List price for the item. • `salesPrice` — Sales price for the item. • `totalAdjustmentAmount` — Adjustments made to the unit price for the item. This value is informational only and isn't used in pricing calculations. • `totalAmount` — Total amount for the item. • `totalListPrice` — Total list price for the item. • `totalPrice` — Total price for the item including adjustments but excluding taxes. • `totalTax` — Total tax for the item. • `unitAdjustedPrice` — Unit price, including adjustments, for the item. This value is informational only and isn't used in pricing calculations. • `unitAdjustmentAmount` — Total amount including discounts, but excluding shipping and tax, for product items in the cart.

Error Conditions

Error Condition	Description
The user doesn't have access to the cart.	Error Message: You don't have access to this cart. If possible, contact the admin for this web store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400

Flow Core Action for Commerce Checkout Flow: Create Cart

Create a cart.

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

In Flow Builder, add an Action element to your flow. Select the **Commerce** category, and search for **Create Cart**. To access this action from REST API, use the name `createCart`.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Web Store ID`	The ID of the web store.
`Cart Input`	This input is an Apex-defined variable of class ConnectApi.CartInput, which includes these fields: • `effectiveAccountId` — (Optional) ID of the buyer account or guest buyer profile for which the request is made. If unspecified, the default value is determined from context. • `isSecondary` — (Optional) Specifies whether the cart is secondary (*`true`*) or not (*`false`*). If unspecified, defaults to *`false`*. • `name` — (Optional) Name of the cart. The name can have up to 250 Unicode characters. If unspecified, defaults to the generated name. • `type` — (Optional) Type of cart. The only valid value is *`Cart`*. If unspecified, defaults to *`Cart`*.

Store Output Values

Use output values later in the flow. The values are assigned when the cart is created.

Output Parameter	Description
`Cart Summary`	This output is an Apex-defined variable of class ConnectApi.CartSummary, which includes these fields: • `accountId` — ID of the account for the cart. • `cartId` — ID of the cart. • `currencyIsoCode` — Three-letter ISO 4217 currency code associated with the cart. • `grandTotalAmount` — Grand total amount including shipping and tax for items in the cart, in the currency of the cart.

Output Parameter	Description
	• `isSecondary` — Specifies whether the cart is secondary (*`true`*) or not (*`false`*). • `name` — Name of the cart. • `purchaseOrderNumber` — Purchase order for the cart. • `status` — Status of the cart. Possible values are: – *`Active`*— Cart is active. – *`Checkout`*— Cart is in checkout. – *`Closed`*— Cart is closed. – *`PendingDelete`*— Cart is pending deletion; for example, a user deleted the cart but the job hasn't completed yet. – *`Processing`*— Cart is processing. • `totalChargeAmount` — Total amount for shipping and other charges in the currency of the cart. • `totalListPrice` — Total list price for the cart. • `totalProductAmount` — Total amount including discounts, but excluding shipping and tax, for product items in the cart. • `totalProductAmountAfterAdjustments` — Total product amount, including promotions. • `totalProductCount` — Total count of items in the cart. • `totalPromotionalAdjustmentAmount` — Total promotional adjustment amount for items in the cart. • `totalTaxAmount` — Total tax amount for the cart, including tax on shipping, if applicable. • `type` — Type of cart. Value is always *`Cart`*. • `uniqueProductCount` — Total count of unique items, or SKUs, in the cart. • `webstoreId` — ID of the web store of the cart.

Error Conditions

Error Condition	Description
The user doesn't have access to create a cart.	Error Message: You don't have access to this cart. If possible, contact the admin for this web store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400

Flow Core Action for Commerce Checkout Flow: Delete Cart

Delete a cart.

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

In Flow Builder, add an Action element to your flow. Select the **Commerce** category, and search for **Delete Cart**. To access this action from REST API, use the name `deleteCart`.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cart State or ID`	ID of the cart or state of the cart to delete. Valid state values are: *`active`* and *`current`*. A current cart is neither closed nor pending deletion.
`effectiveAccountId`	(Optional) ID of the buyer account or guest buyer profile for which the request is made. If unspecified, the default value is determined from context.
`Web Store ID`	ID of the web store associated with the cart.

Store Output Values

Output values aren't available for this action.

Error Conditions

Error Condition	Description
The user doesn't have access to the cart.	Error Message: You don't have access to this cart. If possible, contact the admin for this web store. Error Code: INSUFFICIENT_ACCESS_OR_READONLY HTTP Status Code: 400

Flow Core Action for Commerce Checkout Flow: Get Cart Items

Get items in a cart.

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

In Flow Builder, add an Action element to your flow. Select the **Commerce** category, and search for **Get Cart Items**. To access this action from REST API, use the name `getCartItems`.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cart ID`	The ID of the cart owned by the user.
`Effective Account ID`	(Optional) The ID of the buyer account or guest buyer profile for which the request is made. If unspecified, the default value is determined from context.
`User ID`	The ID of the buying user who owns the cart.
`Web Store ID`	The ID of the store associated with the cart.

Store Output Values

Use output values later in the flow. The values are assigned when the item is created.

Output Parameter	Description
`Cart Items`	An Apex ConnectApi.CartItemCollection record that includes a collection of line items in a cart.

Error Conditions

Error Condition	Description
A required parameter hasn't been specified.	Error Message: You must specify a value for the {0} parameter. <!--Where 0 is the API name of the parameter that requires input.--> Error Code: REQUIRED_FIELD_MISSING HTTP Status Code: 400
The specified ID is invalid.	Error Message: Something's not right with the ID "{0}" specified for the {1} parameter. Check it and try again. <!--Where 0 is the invalid ID, and 1 is the API name of the input parameter with the invalid ID.--> Error Code: INVALID_INPUT HTTP Status Code: 400
The specified effective account ID is invalid for the account or guest buyer profile.	Error Message: The ID "{0}" specified for the {1} parameter isn't a valid {2} record. <!--Where 0 is the specified ID, and 1 is the API name of the parameter with the specified ID, and 2 is the name of the valid record type.--> Error Code: INVALID_TYPE HTTP Status Code: 400
The specified store or cart doesn’t exist.	Error Message: We couldn't find a record with the ID "{0}" specified for the {1} parameter. Check the record and try again. <!--Where 0 is the ID of the record that doesn't exist, and 1 is the parameter that the ID was specified for–> Error Code: RECORD_NOT_FOUND HTTP Status Code: 400

Flow Core Action for Commerce Checkout Flow: Get Cart Promotions

Get promotions associated with a cart.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Note: Cart actions aren't available in flows for B2B stores built on an Aura template. To build a B2B Commerce Checkout flow for an Aura store, see B2B Commerce Checkout Flow (Aura).

In Flow Builder, add an Action element to your flow. Select the **Commerce** category, and search for **Get Cart Promotions**. To access this action from REST API, use the name `getCartPromotions`.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cart ID`	The ID of the cart owned by the user.
`Effective Account ID`	(Optional) The ID of the buyer account or guest buyer profile for which the request is made. If unspecified, the default value is determined from context.
`User ID`	The ID of the buying user who owns the cart.
`Web Store ID`	The ID of the store associated with the cart.

Store Output Values

Use output values later in the flow. The values are assigned when the item is created.

Output Parameter	Description
`Cart Promotions`	An Apex ConnectApi.CartPromotionCollection record that includes a collection of line items in a cart.

Error Conditions

Error Condition	Description
A required parameter hasn't been specified.	Error Message: You must specify a value for the {0} parameter. <!--Where 0 is the API name of the parameter that requires input.--> Error Code: REQUIRED_FIELD_MISSING HTTP Status Code: 400
The specified ID is invalid.	Error Message: Something's not right with the ID "{0}" specified for the {1} parameter. Check it and try again. <!--Where 0 is the invalid ID, and 1 is the API name of the input parameter with the invalid ID.--> Error Code: INVALID_INPUT HTTP Status Code: 400

Error Condition	Description
The specified effective account ID is invalid for the account or guest buyer profile.	Error Message: The ID "{0}" specified for the {1} parameter isn't a valid {2} record. <!--Where 0 is the specified ID, and 1 is the API name of the parameter with the specified ID, and 2 is the name of the valid record type.--> Error Code: INVALID_TYPE HTTP Status Code: 400
The specified store or cart doesn't exist.	Error Message: We couldn't find a record with the ID "{0}" specified for the {1} parameter. Check the record and try again. <!--Where 0 is the ID of the record that doesn't exist, and 1 is the parameter that the ID was specified for–> Error Code: RECORD_NOT_FOUND HTTP Status Code: 400

Salesforce Order Management Flow Core Actions

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Salesforce Order Management provides several core actions for implementing order management functionality in flows. To add one of these actions to your flow, add an Action element. Then select the **Order Management** category, and search for the appropriate action.

These actions use Apex-defined input and output variables that map to input and output classes in the Apex ConnectApi namespace. For more information on using Apex-defined variables in flows, see Considerations for the Apex-Defined Data Type on page 260.

Flow Core Action for Order Management: Add Order Item Summary
Add up to 100 order product summaries to an order summary. This action creates a change order record, an order product record, and an order product summary record. It also creates any supporting adjustment, tax, and summary records.

Flow Core Action for Order Management: Adjust Order Item Summaries Preview
Preview the expected results of adjusting the price of one or more order product summaries on an order summary, without executing the adjustment. You can only apply a discount, not an increase. The output of this action contains the values that would be set on the change orders created by submitting the proposed adjustment.

Flow Core Action for Order Management: Adjust Order Item Summaries Submit
Adjust the price of one or more order product summaries on an order summary. You can only apply a discount, not an increase. This action creates one or more change order records.

Flow Core Action for Order Management: Authorize Payment
Authorize a payment on a credit card. You can include details for a new credit card or reference an existing PaymentMethod.

Flow Core Action for Order Management: Cancel Fulfillment Order Item
Cancel fulfillment order products from a fulfillment order. You can cancel more than one product and specify a quantity to cancel for each of them. This action doesn't cancel the associated order product summaries, it only reduces their allocated quantities. Usually, you reallocate the canceled quantities to a new fulfillment order.

Flow Core Action for Order Management: Cancel Order Item Summaries Preview
Preview the expected results of canceling one or more order product summaries from an order summary without executing the cancel. The output of this action contains the values that would be set on the change order created by submitting the proposed cancel.

Flow Core Action for Order Management: Cancel Order Item Summaries Submit

Cancel one or more order product summaries from an order summary. This action creates a change order record.

Flow Core Action for Order Management: Cancel Order Summary Preview

Preview the expected results of canceling all order product summaries for an order summary without executing the cancel. The output of this action contains the values that would be set on the change order created by submitting the proposed cancel.

Flow Core Action for Order Management: Cancel Order Summary Submit

Cancel all order product summaries for an order summary. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation.

Flow Core Action for Order Management: Confirm Held Fulfillment Order Capacity

Confirm held fulfillment order capacity at one or more locations. This action decreases a location's held capacity and increases its assigned fulfillment order count. Confirm held capacity when you assign a fulfillment order to a location.

Flow Core Action for Order Management: Create Credit Memo

Create a credit memo to represent the refund for one or more change orders associated with an order summary.

Flow Core Action for Order Management: Create Fulfillment Order

Create one or more fulfillment orders and fulfillment order products for an order delivery group summary, which defines a recipient and delivery method. You specify the order product summaries to fulfill and the fulfillment locations to handle them. If you specify multiple fulfillment locations, a fulfillment order is created for each one.

Flow Core Action for Order Management: Create Fulfillment Orders

Create fulfillment orders and fulfillment order products for multiple order delivery group summaries, each of which defines a recipient and delivery method. You specify the order product summaries to fulfill and the fulfillment locations to handle them. If you specify multiple fulfillment locations for one order delivery group summary, a fulfillment order is created for each one.

Flow Core Action for Order Management: Create an Invoice from Change Orders

Create an invoice to represent the charges for one or more change orders. Create invoices for change orders that increase order amounts, such as return fees. When you ensure the refund for a return, include the invoices for the associated return fees in the input.

Flow Core Action for Order Management: Create an Invoice from Fulfillment Order

Create an invoice for a fulfillment order that doesn't have one.

Flow Core Action for Order Management: Create Order Payment Summary

Create an order payment summary for a payment authorization or payments that use the same payment method and are attached to the same order summary.

Flow Core Action for Order Management: Create Order Summary

Create an order summary based on an order. That order is considered the original order for the order summary. Subsequent change orders that apply to the order summary are also represented as order records.

Flow Core Action for Order Management: Create Return Order

Create a return order and return order items for order items belonging to an order summary. You can add return fees for any of the order items.

Flow Core Action for Order Management: Ensure Funds Async

Ensure funds for an invoice, and apply them to it. If needed, capture authorized funds by sending a request to a payment provider. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation so you can track its status. Payment gateway responses appear in the payment gateway log and don't affect the background operation status.

Flow Core Action for Order Management: Ensure Refunds Async

Ensure refunds for a credit memo or excess funds by sending a request to a payment provider. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation so you can track its status. Payment gateway responses appear in the payment gateway log and don't affect the background operation status.

Flow Core Action for Order Management: Find Routes with Fewest Splits

Evaluate ordered product quantities against available inventory to determine the smallest combination of locations that can fulfill the order. If multiple combinations of the minimum number of locations can fulfill the order, the action returns multiple options. Optionally, you can specify a maximum allowable number of locations. By default, the action executes up to 1,000,000 times, stopping when it hits 10,000 results.

Flow Core Action for Order Management: Use OCI to Find Routes with Fewest Splits

Evaluate ordered product quantities against available inventory to determine the smallest combination of locations that can fulfill the order. If multiple combinations of the minimum number of locations can fulfill the order, the action returns multiple options. Optionally, you can specify a maximum allowable number of locations and a list of locations to exclude from the calculation. This action combines the Omnichannel Inventory Get Availability action and the Order Management Find Routes with Fewest Splits actions. Instead of calling Get Availability and including the output in the Find Routes with Fewest Splits input, call this action and specify a location or location group to fulfill each ordered product. By default, this action executes up to 1,000,000 times, stopping when it hits 10,000 results. This action handles the inventory check.

Flow Core Action for Order Management: Get Fulfillment Order Capacity Values

Get information about the current fulfillment order capacity of one or more locations.

Flow Core Action for Order Management: Hold Fulfillment Order Capacity

Hold capacity to process fulfillment orders at one or more locations. This action increases a location's held capacity. Hold capacity when you plan to assign a fulfillment order to a location.

Flow Core Action for Order Management: Order Routing Rank by Average Distance

Calculate the average distance from sets of inventory locations to an order recipient, and return the sets sorted by that average distance. Use this action to compare the average shipping distances for different sets of locations that can fulfill an order.

Flow Core Action for Order Management: Release Held Fulfillment Order Capacity

Release held fulfillment order capacity at one or more locations. This action decreases a location's held capacity without increasing its assigned fulfillment order count. Release held capacity when you cancel assigning a fulfillment order to a location.

Flow Core Action for Order Management: Return Order Item Summaries Preview

Preview the expected results of a simple return of one or more order product summaries from an order summary without executing the return. The output of this action contains the values that would be set on the change order created by submitting the proposed return.

Flow Core Action for Order Management: Return Order Item Summaries Submit

Return one or more order product summaries from an order summary. This action is a simple return that creates a change order but not a return order.

Flow Core Action for Order Management: Return Return Order Items

Process one or more return order line items belonging to a return order. This action creates a change order record for the returned items and makes the processed return order line items read-only. You can include return order fees associated with the return order line items. If you do, a change order record is created for the return fees. If a processed return order line item has a remaining expected quantity, the action creates a separate return order line item representing that quantity.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Add Order Item Summary

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Add up to 100 order product summaries to an order summary. This action creates a change order record, an order product record, and an order product summary record. It also creates any supporting adjustment, tax, and summary records.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Add Order Item Summary**.

Important: Don't call this action via REST API. Use it only in flows.

Set Input Values

Create record variables to use in the input. Use values from earlier in the flow to set their values. The action generates records based on those values. Remember to include all required values for each object type. For example, the order item summary record variable must include an order delivery group summary ID.

Note: For this action's input values, use record variables, not existing records or record IDs.

Input Parameter	Description
`Order Item Summary Input`	This input is an Apex-defined variable of class runtime_commerce_oms.AddOrderItemSummaries. For information on setting up the input data, see the Usage section of this topic. The variable has one field: `newItems`. This field is a list of one or more Apex-defined variables of class runtime_commerce_oms.AddItem. Each of the variables includes these fields. • `orderItemSummary` — An order product summary record variable representing the order product to add. • `reasonCode` — Reason for the addition. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. • `orderItemTaxLineItemSummaries` — A list of zero or more order product tax line item summary record variables associated with the order product summary. • `orderItemAdjustmentLineSummaries` — A list of zero or more Apex-defined variables of class runtime_commerce_oms.AddItemAdjustment that has these fields. – `orderItemAdjustmentLineSummary` — An order product adjustment line summary record variable associated with the order product being added. – `orderItemTaxLineItemSummaries` — A list of zero or more order product tax line item summary record variables associated with the order product adjustment line summary.

Store Output Values

Output Parameter	Description
`Order Item Summary Output`	This output is an Apex-defined variable of class ConnectApi.AddOrderItemSummaryOutputRepresentation. It includes these fields. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record.

Output Parameter	Description
	• `changeBalances` — An Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation that has these fields. – `grandTotalAmount` — Change to the total with tax. – `totalAdjDeliveryAmtWithTax` — Change to the adjusted delivery subtotal, including tax. – `totalAdjDistAmountWithTax` — Change to the total order adjustments, including tax. – `totalAdjProductAmtWithTax` — Change to the adjusted product subtotal, including tax. – `totalAdjustedDeliveryAmount` — Change to the adjusted delivery subtotal. – `totalAdjustedDeliveryTaxAmount` — Change to the adjusted delivery subtotal tax. – `totalAdjustedProductAmount` — Change to the adjusted product subtotal. – `totalAdjustedProductTaxAmount` — Change to the adjusted product subtotal tax. – `totalAdjustmentDistributedAmount` — Change to the total order adjustments. – `totalAdjustmentDistributedTaxAmount` — Change to the total order adjustments tax. – `totalAmount` — Change to the pretax total. – `totalExcessFundsAmount` — The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment has been captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. – `totalRefundableAmount` — The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. – `totalRequiredFundsAmount` — The total amount associated with the order products added in the current action. This amount isn't necessarily the amount that must be captured. For example, in an even exchange flow, the order amount reduction from canceling the exchanged products offsets the required funds amount of the replacement products. – `totalTaxAmount` — Change to the total tax. • `changeOrderId` — ID of the change order generated by the action. • `newItems` — A list of one or more Apex-defined variables of class ConnectApi.AddItemOutputRepresentation, each of which represents an added order product, and has these fields. – `id` — ID of the order product summary. – `name` — Name of the order product summary. – `orderItemAdjustmentLineSummaries` — A list of zero or more Apex-defined variables of class ConnectApi.AddItemAdjustmentOutputRepresentation, each of which represents an order product adjustment line summary associated with the added order product summary, and has these fields. • `id` — ID of the order product adjustment line summary.

Output Parameter	Description
	• `name` — Name of the order product adjustment line summary. • `orderItemTaxLineItemSummaries` — A list of zero or more Apex-defined variables of class ConnectApi.AddItemTaxOutputRepresentation, each of which represents an order product tax line item summary associated with the order product adjustment line summary, and has these fields. – `id` — ID of the order product tax line item summary. – `name` — Name of the order product tax line item summary. – `orderItemTaxLineItemSummaries` — A list of zero or more Apex-defined variables of class ConnectApi.AddItemTaxOutputRepresentation, each of which represents an order product tax line item summary associated with the added order product summary, and has these fields. • `id` — ID of the order product tax line item summary. • `name` — Name of the order product tax line item summary. • `orderSummaryId` — ID of the order summary specified in the input.

To set up the Order Item Summary Input:

1. Use record variables to define the order product summaries, order product adjustment line summaries, and order product tax line item summaries. Sending an Id isn't required.
   - Required fields for an order product summary:
     - ListPrice (Only if Order Summary Pricebook2Id is NULL or empty)
     - Name
     - OrderDeliveryGroupSummaryId
     - OrderSummaryId
     - PricebookEntryId (Only if Order Summary Pricebook2Id is set)
     - Product2Id
     - Quantity
     - TotalLineAmount
     - UnitPrice
     - TypeCode
     - Type
   - Required fields for an order product adjustment line summary:
     - Amount
     - Name
     - OrderSummaryId
   - Required fields for an order product tax line item summary:
     - Amount
     - Name
     - OrderSummaryId

  - TaxEffectiveDate
  - Type

2. Use an assignment element to set the `orderItemSummary` field on a runtime_commerce_oms.AddItem variable to the order product summary record variable.
3. For each adjustment to the product being added, use an assignment element to set the `orderItemAdjustmentLineSummary` field on a runtime_commerce_oms.AddItemAdjustment variable to the corresponding order product adjustment line summary record variable. Use assignment elements to add the order product tax line summary record variables associated with it to the `orderItemTaxLineItemSummaries` field on the same runtime_commerce_oms.AddItemAdjustment variable.
4. Use an assignment element to add the runtime_commerce_oms.AddItemAdjustment variables to the `orderItemAdjustmentLineSummaries` field on the runtime_commerce_oms.AddItem variable.
5. For each tax on the product being added, use an assignment element to add the corresponding order product tax line summary record variable to the `orderItemTaxLineItemSummaries` field on the runtime_commerce_oms.AddItem variable.
6. Use an assignment element to set the `reasonCode` field on the runtime_commerce_oms.AddItem variable to a valid reason.
7. Use an assignment element to add the runtime_commerce_oms.AddItem variable to the `newItems` field on a runtime_commerce_oms.AddOrderItemSummaries variable.
8. Repeat steps 1 through 6 for each order product that you want to include in the action, adding the inputs to the same runtime_commerce_oms.AddOrderItemSummaries variable. You can add up to five order products at a time.
9. Use the runtime_commerce_oms.AddOrderItemSummaries variable in the action input.

Flow Core Action for Order Management: Adjust Order Item Summaries Preview

Preview the expected results of adjusting the price of one or more order product summaries on an order summary, without executing the adjustment. You can only apply a discount, not an increase. The output of this action contains the values that would be set on the change orders created by submitting the proposed adjustment.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Adjust Order Item Summaries Preview**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Order Summary Id`	ID of the order summary associated with the order product summaries that you want to preview adjusting the prices of.
`Adjust Order Product Summaries Input`	This input is an Apex-defined variable of class ConnectApi.AdjustOrderItemSummaryInputRepresentation, which includes these fields: • `adjustItems` —This field is a list of Apex-defined variables of class ConnectApi.AdjustItemInputRepresentation. Each of the variables includes these fields: – `orderItemSummaryId` —ID of an order product summary to preview a price adjustment for. – `description` —Optional description of the adjustment. – `adjustmentType` —Specifies how to calculate the adjustment amount from the `discountValue` field. It can have one of these values:

Input Parameter	Description
	• *`AmountWithTax`*—The value of `discountValue` is the adjustment, including tax. • *`AmountWithoutTax`*—The value of `discountValue` is the adjustment, not including tax. Tax is calculated on the value and added. • *`Percentage`*—The value of `discountValue` is a percentage discount. It’s divided by 100, and then multiplied by the total price and total tax amount of the order product summary to determine the adjustment amount. – `discountValue` —The value used to calculate the adjustment amount, as specified by the `adjustmentType`. It must be a negative value. – `reason` —Adjustment reason. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. • `allocatedItemsChangeOrderType` —Specifies how change orders would be created for order product summary quantities that are currently being fulfilled, defined as QuantityAllocated - QuantityFulfilled. It can have one of these values: – *`Disallowed`*—When distributing the adjustment, ignore any quantities being fulfilled. If an order product summary’s entire quantity is being fulfilled, return an error. This value is the default. – *`InFulfillment`*—When distributing the adjustment, include quantities being fulfilled. Submitting the adjustment would create a separate change order for the adjustments made to those quantities. – *`PreFulfillment`*—When distributing the adjustment, include quantities being fulfilled. Submitting the adjustment would include the adjustments made to those quantities in the change order for pre-fulfillment quantity adjustments.

Store Output Values

Use output values later in the flow.

Output Parameter	Description
`Adjust Order Product Summary Output`	This output is an Apex-defined variable of class ConnectApi.AdjustOrderSummaryOutputRepresentation, which contains the financial changes that would result from the proposed adjustment. Most of the values represent the deltas of the values on the associated order summary. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record. The `orderSummaryId` field is the ID of the order summary specified in the input. The `changeBalances` field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields. • `grandTotalAmount` —Change to the total with tax. • `totalAdjDeliveryAmtWithTax` —Change to the adjusted delivery subtotal, including tax. • `totalAdjDistAmountWithTax` —Change to the total order adjustments, including tax. • `totalAdjProductAmtWithTax` —Change to the adjusted product subtotal, including tax. • `totalAdjustedDeliveryAmount` —Change to the adjusted delivery subtotal.

Output Parameter	Description
	• `totalAdjustedDeliveryTaxAmount` —Change to the adjusted delivery subtotal tax. • `totalAdjustedProductAmount` —Change to the adjusted product subtotal. • `totalAdjustedProductTaxAmount` —Change to the adjusted product subtotal tax. • `totalAdjustmentDistributedAmount` —Change to the total order adjustments. • `totalAdjustmentDistributedTaxAmount` —Change to the total order adjustments tax. • `totalAmount` —Change to the pretax total. • `totalExcessFundsAmount` —The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment has been captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. • `totalRefundableAmount` —The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. • `totalTaxAmount` —Change to the total tax. The `postFulfillmentChangeOrderId` field is always null for a preview action. The `preFulfillmentChangeOrderId` field is always null for a preview action. The `inFulfillmentChangeOrderId` field is always null for a preview action.

Usage

When a price adjustment is applied to an order product summary, its quantities are considered in three groups:

- Pre-fulfillment—QuantityAvailableToFulfill, which is equal to QuantityOrdered - QuantityCanceled - QuantityAllocated
- In-fulfillment—QuantityAllocated - QuantityFulfilled
- Post-fulfillment—QuantityAvailableToReturn, which is equal to QuantityFulfilled - QuantityReturnInitiated

You can apply adjustments to these groups in three different ways, controlled by the `allocatedItemsChangeOrderType` input property:

- Distribute the adjustment evenly between pre-fulfillment and post-fulfillment quantities. Ignore in-fulfillment quantities. Submitting the adjustment would create one change order for the adjustments to pre-fulfillment quantities and one change order for the adjustments to post-fulfillment quantities.
- Distribute the adjustment evenly between pre-fulfillment, in-fulfillment, and post-fulfillment quantities. Submitting the adjustment would create one change order for the adjustments to both pre-fulfillment and in-fulfillment quantities, and one change order for the adjustments to post-fulfillment quantities.
- Distribute the adjustment evenly between pre-fulfillment, in-fulfillment, and post-fulfillment quantities. Submitting the adjustment would create one change order for the adjustments to pre-fulfillment quantities, one change order for the adjustments to in-fulfillment quantities, and one change order for the adjustments to post-fulfillment quantities.

To set up the Adjust Order Product Summaries Input:

- Use Assignment elements to set the `orderItemSummaryId`, `description`, `adjustmentType`, `discountValue`, and `reason` field values on one or more `ConnectApi.AdjustItemInputRepresentation` variables.

- Use an Assignment element to add the `ConnectApi.AdjustItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable.
- Use an Assignment element to set the `allocatedItemsChangeOrderType` field on the `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable.
- Use the `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable and the order summary ID in the action input.

In a flow for adjusting the prices of order product summaries, display the output of this action for the user to review before executing the adjustment. When the user verifies the expected results, pass the same input to an Adjust Order Item Summaries Submit action.

SEE ALSO:

Flow Core Action for Order Management: Adjust Order Item Summaries Submit

Add and Edit Elements

Flow Core Action for Order Management: Adjust Order Item Summaries Submit

Adjust the price of one or more order product summaries on an order summary. You can only apply a discount, not an increase. This action creates one or more change order records.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Adjust Order Item Summaries Submit**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Order Summary Id`	ID of the order summary associated with the order product summaries that you want to adjust the prices of.
`Adjust Order Product Summaries Input`	This input is an Apex-defined variable of class ConnectApi.AdjustOrderItemSummaryInputRepresentation, which includes these fields: • `adjustItems` —This field is a list of Apex-defined variables of class ConnectApi.AdjustItemInputRepresentation. Each of the variables includes these fields: – `orderItemSummaryId` —ID of an order product summary to adjust the price of. – `description` —Optional description of the adjustment. – `adjustmentType` —Specifies how to calculate the adjustment amount from the `discountValue` field. It can have one of these values: • *`AmountWithTax`*—The value of `discountValue` is the adjustment, including tax.

Input Parameter	Description
	• *`AmountWithoutTax`*—The value of `discountValue` is the adjustment, not including tax. Tax is calculated on the value and added. • *`Percentage`*—The value of `discountValue` is a percentage discount. It's divided by 100, and then multiplied by the total price and total tax amount of the order product summary to determine the adjustment amount. – `discountValue` —The value used to calculate the adjustment amount, as specified by the `adjustmentType`. It must be a negative value. – `reason` — Adjustment reason.The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. • `allocatedItemsChangeOrderType` —Specifies how to create change orders for order product summary quantities that are currently being fulfilled, defined as QuantityAllocated - QuantityFulfilled. It can have one of these values: – *`Disallowed`*—When distributing the adjustment, ignore any quantities being fulfilled. If an order product summary's entire quantity is being fulfilled, return an error. This value is the default. – *`InFulfillment`*—When distributing the adjustment, include quantities being fulfilled. Create a separate change order for the adjustments made to those quantities. – *`PreFulfillment`*—When distributing the adjustment, include quantities being fulfilled. Include the adjustments made to those quantities in the change order for pre-fulfillment quantity adjustments.

Store Output Values

Use output values later in the flow. The values are assigned when the change orders are created.

Output Parameter	Description
`Adjust Order Product Summary Output`	This output is an Apex-defined variable of class ConnectApi.AdjustOrderSummaryOutputRepresentation. Depending on the order product summaries included in the adjustment, one or more change orders are generated. If multiple change orders are generated, then the `changeBalances` values combine the values from both of them. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record. The `orderSummaryId` field is the ID of the order summary specified in the input. The `changeBalances` field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields. • `grandTotalAmount` —Change to the total with tax. • `totalAdjDeliveryAmtWithTax` —Change to the adjusted delivery subtotal, including tax. • `totalAdjDistAmountWithTax` —Change to the total order adjustments, including tax. • `totalAdjProductAmtWithTax` —Change to the adjusted product subtotal, including tax. • `totalAdjustedDeliveryAmount` —Change to the adjusted delivery subtotal. • `totalAdjustedDeliveryTaxAmount` —Change to the adjusted delivery subtotal tax.

Output Parameter	Description
	• `totalAdjustedProductAmount` —Change to the adjusted product subtotal. • `totalAdjustedProductTaxAmount` —Change to the adjusted product subtotal tax. • `totalAdjustmentDistributedAmount` —Change to the total order adjustments. • `totalAdjustmentDistributedTaxAmount` —Change to the total order adjustments tax. • `totalAmount` —Change to the pretax total. • `totalExcessFundsAmount` —The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment has been captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. • `totalRefundableAmount` —The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. • `totalTaxAmount` —Change to the total tax. The `postFulfillmentChangeOrderId` is the ID of the change order representing the portion of the adjustment that was applied to order product summary quantities that have been fulfilled. The `preFulfillmentChangeOrderId` is the ID of the change order representing the portion of the adjustment that was applied to order product summary quantities that haven't been fulfilled. If the input specified an `allocatedItemsChangeOrderType` of *`PreFulfillment`*, this change order also includes the changes applicable to order product summary quantities that are in the process of being fulfilled. The `inFulfillmentChangeOrderId` is the ID of the change order representing the portion of the adjustment that was applied to order product summary quantities that are in the process of being fulfilled. This change order is only created for an input that specified an `allocatedItemsChangeOrderType` of *`InFulfillment`*.

Usage

When a price adjustment is applied to an order product summary, its quantities are considered in three groups:

- Pre-fulfillment—QuantityAvailableToFulfill, which is equal to QuantityOrdered - QuantityCanceled - QuantityAllocated
- In-fulfillment—QuantityAllocated - QuantityFulfilled
- Post-fulfillment—QuantityAvailableToReturn, which is equal to QuantityFulfilled - QuantityReturnInitiated

You can apply adjustments to these groups in three different ways, controlled by the `allocatedItemsChangeOrderType` input property:

- Distribute the adjustment evenly between pre-fulfillment and post-fulfillment quantities. Ignore in-fulfillment quantities. Create one change order for the adjustments to pre-fulfillment quantities and one change order for the adjustments to post-fulfillment quantities.
- Distribute the adjustment evenly between pre-fulfillment, in-fulfillment, and post-fulfillment quantities. Create one change order for the adjustments to both pre-fulfillment and in-fulfillment quantities, and one change order for the adjustments to post-fulfillment quantities.

- Distribute the adjustment evenly between pre-fulfillment, in-fulfillment, and post-fulfillment quantities. Create one change order for the adjustments to pre-fulfillment quantities, one change order for the adjustments to in-fulfillment quantities, and one change order for the adjustments to post-fulfillment quantities.

To set up the Adjust Order Product Summaries Input:

- Use Assignment elements to set the `orderItemSummaryId`, `description`, `adjustmentType`, `discountValue`, and `reason` field values on one or more `ConnectApi.AdjustItemInputRepresentation` variables.
- Use an Assignment element to add the `ConnectApi.AdjustItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable.
- Use an Assignment element to set the `allocatedItemsChangeOrderType` field on the `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable.
- Use the `ConnectApi.AdjustOrderItemSummaryInputRepresentation` variable and the order summary ID in the action input.

In a flow for adjusting the prices of order product summaries, run an Adjust Order Item Summaries Preview action before running this action. Then display its output for the user to review. When the user verifies the expected results, pass the same input to this action.

After submitting a price adjustment, process refunds as appropriate:

- If the discount only applied to order product summaries for which payment hasn't been captured, it doesn't require a refund. This situation normally applies to order products in the US that haven't been fulfilled.
- If the discount applied to order product summaries that haven't been fulfilled and for which payment has been captured, process a refund. In this case, pass the `totalExcessFundsAmount` from `changeBalances` to the Ensure Refunds Async action.
- If the discount applied to order product summaries that have been fulfilled, process a refund. Pass the `postFulfillmentChangeOrderId` to the Create Credit Memo action, then pass the credit memo to the Ensure Refunds Async action.
- If the discount applied to both fulfilled and unfulfilled order product summaries for which payment has been captured, process both refunds. Pass the `postFulfillmentChangeOrderId` to the Create Credit Memo action, then pass the credit memo and the `totalExcessFundsAmount` from `changeBalances` to the Ensure Refunds Async action.

Important: Excess funds aren't reduced until the payment processor issues a refund. If you don't process refunds promptly, subsequent refunds can be inaccurate. Consider this example.

- An order with a total amount of $100 is placed, and the amount is captured immediately.
- A product is canceled from the order, resulting in $20 of excess funds.
- Before the excess funds are sent to the payment provider in an ensure refunds action, another product is canceled. This cancellation adds another $20 of excess funds. However, because the original $20 hasn't been refunded yet, the cancel action returns a total excess funds amount of $40.
- The first excess funds amount ($20) is sent to the payment provider in an ensure refunds request.
- The second excess funds amount ($40) is sent to the payment provider in an ensure refunds request.
- The payment provider receives requests for $60 of refunds, when the correct refund total is $40. Because the total refund amount is less than the total captured amount of $100, the payment provider issues $60 in refunds.

SEE ALSO:

Flow Core Action for Order Management: Adjust Order Item Summaries Preview

Add and Edit Elements

Flow Core Action for Order Management: Authorize Payment

Authorize a payment on a credit card. You can include details for a new credit card or reference an existing PaymentMethod.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Authorize Payment**. To access this action from REST API, use the name `authorizePayment`.

Note: This action is available with the PaymentsAPIUser user permission.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Payment Authorization Request`	This input is an Apex-defined variable of class ConnectApi.AuthorizationRequest, which includes these fields: • `accountId` —ID of the account that contains the payment transaction being authorized. • `amount` —Authorization amount. • `comments` —(Optional) Comments for the payment authorization. • `currencyIsoCode` —Three-letter ISO 4217 currency code associated with the payment group record. • `effectiveDate` —Date that the authorization is applied to the transaction. • `paymentGatewayId` —Payment gateway that processes the authorization. • `paymentGroup` —(Optional) Payment group for the authorization. The payload must reference either a paymentGroup or a paymentGroupId, but not both. This field is an Apex-defined variable of class ConnectApi.PaymentGroupRequest, which includes these fields: – `createPaymentGroup`—(Optional) Specifies whether to create a payment group (*`true`*) or not (*`false`*). – `currencyIsoCode` —(Optional) Three-letter ISO 4217 currency code associated with the payment group record. – `id`—(Optional) ID of the payment group record. – `sourceObjectId`—(Optional) Source object ID of the payment group record. Supports only OrderId. • `paymentMethod` —Payment method for the authorization. The payload must either reference an existing payment method or include details for a new payment method, but not both. This field is an Apex-defined variable of class ConnectApi.AuthApiPaymentMethodRequest, which includes these fields: This input includes the fields from the parent class, ConnectApi.BaseApiPaymentMethodRequest. – `address`—Address for the payment method. This field is an Apex-defined variable of class ConnectApi.AddressRequest. It includes these fields, all of which are optional: • `city` • `companyName` • `country` • `postalCode` • `state` • `street`

Input Parameter	Description
	– `cardPaymentMethod` —(Optional) When using a new payment method, the details of that method. This field is an Apex-defined variable of class ConnectApi.CardPaymentMethodRequest, which includes these fields: • `accountId`—Salesforce account to which this payment method is linked. • `cardCategory` —Valid values are *`CreditCard`* and *`DebitCard`*. • `cardHolderFirstName`—First name of the card holder. • `cardHolderLastName`—Last name of the card holder. • `cardHolderName`—Full name of the card holder. • `cardNumber`—Card number. • `cardType`—Valid values are: – *`AmericanExpress`* – *`DinersClub`* – *`JCB`* – *`Maestro`* – *`MasterCard`* – *`Visa`* • `comments` —(Optional) Comments for the payment method. • `cvv`—CVV. • `email`—Email of the card holder. • `expiryMonth`—Card expiration month. • `expiryYear`—Card expiration year. • `nickName`—(Optional) Nickname for the payment method. • `startMonth`—(Optional) Start month of the card. • `startYear`—(Optional) Start year of the card. – `id`—(Optional) When using an existing payment method, the ID of that method. – `saveForFuture`—Whether to save the payment method for future use.

Store Output Values

Use output values later in the flow. The values are assigned when a response is received from the payment gateway.

Output Parameter	Description
`Payment Authorization Response`	This output is an Apex-defined variable of class ConnectApi.AuthorizationResponse, which includes these fields: • `error` —If an error is returned, details about that error. This field is an Apex-defined variable of class ConnectApi.ErrorResponse, which includes these fields: – `errorCode`—Error code. – `message`—More detail, if available.

Output Parameter	Description
	• `gatewayResponse` —Response from the payment gateway. This field is an Apex-defined variable of class ConnectApi.AuthorizationGatewayResponse, which includes this field: – `gatewayAuthorizationCode`—Payment authorization code. • `paymentAuthorization` —Details about the payment authorization. This field is an Apex-defined variable of class ConnectApi.PaymentAuthorizationResponse, which includes these fields: – `accountId` —ID of the account that contains the payment transaction being authorized. – `amount` —Amount that the gateway authorized for the payment transaction. – `currencyIsoCode` —Three-letter ISO 4217 currency code associated with the payment group record. – `effectiveDate`—Date that the authorization becomes effective. – `expirationDate`—Date that the authorization expires. – `id`—ID of the payment authorization record. – `paymentAuthorizationNumber`—System-defined number for the payment authorization record. – `requestDate`—Date that the authorization occurred. – `status`—Status of the payment authorization as returned by the gateway. • `paymentGatewayLogs` —Payment gateway log information about the authorization transaction. This field is a list of Apex-defined variables of class ConnectApi.GatewayLogResponse, each of which includes these fields: – `createdDate`—Date when the gateway log was created. – `gatewayResultCode`—Result codes that show the status of a transaction as it is passed to the financial institution and then returned to the client. – `id`—ID of the gateway log record. – `interactionStatus`—Gateway interaction status. It can be `SUCCESS`, `FAILED`, or `TIMEOUT`. • `paymentGroup` —Details about the payment group. This field is an Apex-defined variable of class ConnectApi.PaymentGroupResponse, which includes these fields: – `currencyIsoCode` —Three-letter ISO 4217 currency code associated with the payment group record. – `id`—ID of the payment group record. – `sourceObjectId`—Source object ID of the payment group record. • `paymentMethod` —Details about the payment method. This field is an Apex-defined variable of class ConnectApi.PaymentMethodResponse, which includes these fields: – `accountId` —ID of the account for the payment method. – `id`—ID of the payment method. – `status`—Status of the payment method.

Usage

Use this action in custom flows that require payment authorization, such as adding an item to an order or an uneven exchange. Before using it, verify with your payment provider that it supports payment authorization calls from Salesforce Order Management.

Flow Core Action for Order Management: Cancel Fulfillment Order Item

Cancel fulfillment order products from a fulfillment order. You can cancel more than one product and specify a quantity to cancel for each of them. This action doesn't cancel the associated order product summaries, it only reduces their allocated quantities. Usually, you reallocate the canceled quantities to a new fulfillment order.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Cancel Fulfillment Order Item**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cancel Fulfillment Order Items Input`	This input is an Apex-defined variable of class ConnectApi.FulfillmentOrderLineItemsToCancelInputRepresentation. The variable has one field, `fulfillmentOrderLineItemsToCancel`, which is a list of Apex-defined variables of class ConnectApi.FulfillmentOrderLineItemInputRepresentation. Each of those variables includes these fields: • `fulfillmentOrderLineItemId` — Reference to the fulfillment order product to cancel. • `quantity` — Quantity to cancel.
`Fulfillment Order Id`	Reference to the fulfillment order that you want to cancel fulfillment order items from.

Store Output Values

Output Parameter	Description
`Cancel Fulfillment Order Items Output`	This value is an Apex-defined variable of class ConnectApi.FulfillmentOrderCancelLineItemsOutputRepresentation. This action doesn't return any data.

Usage

To set up the Cancel Fulfillment Order Items Input, first use Assignment elements to set the `fulfillmentOrderLineItemId` and `quantity` field values on one or more `ConnectApi.FulfillmentOrderLineItemInputRepresentation` variables. Then use an Assignment element to add those variables to the `FulfillmentOrderLineItemsToCancel` field on a ConnectApi.FulfillmentOrderLineItemsToCancelInputRepresentation variable. Use that variable in the action input.

SEE ALSO:

- Add and Edit Elements
- Add and Edit Elements

Flow Core Action for Order Management: Cancel Order Item Summaries Preview

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Preview the expected results of canceling one or more order product summaries from an order summary without executing the cancel. The output of this action contains the values that would be set on the change order created by submitting the proposed cancel.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Cancel Order Item Summaries Preview**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
Cancel Order Product Summary Items Input	This input is an Apex-defined variable of class ConnectApi.ChangeInputRepresentation. The variable has one field: changeItems. This field is a list of Apex-defined variables of class ConnectApi.ChangeItemInputRepresentation. Each variable includes these fields: • changeItemFees—A list of Apex-defined variables of class ConnectApi.ChangeItemFeeInputRepresentation. Each variable has these fields: – amount—Required. Value used to calculate the fee amount, as described by the amountType. It must be a positive value. – amountType—Required. Describes how the fee amount is calculated. It can have one of these values: • *AmountWithTax*—amount is the fee amount, including tax. • *AmountWithoutTax*—amount is the fee amount, not including tax. Tax is calculated on the value and added. • *Percentage*—amount is a percentage. The fee amount is amount divided by 100 and then multiplied by the TotalPrice and TotalTaxAmount of the associated order product summary, prorated for the quantity being returned. • *PercentageGross*—amount is a percentage. The fee amount is amount divided by 100 and then multiplied by the TotalLineAmountWithTax of the associated order product summary, prorated for the quantity being returned. – description—Description of the fee.

Input Parameter	Description
	– priceBookEntryId—Required unless price books are optional in the org. ID of the price book entry associated with the fee product. – product2Id—Required. ID of the product representing the fee. – reason—Required. Reason for the fee. The value must match an entry in the Order Product Summary Change object's Reason picklist. • orderItemSummaryId—Required. ID of an order product summary to cancel. It can't be a shipping charge product. • quantity—Required. Quantity to cancel. • reason—Required. Cancel reason. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. • shippingReductionFlag—Required. Boolean flag that specifies whether to prorate any related delivery charge based on the price change.
Order Summary Id	Reference to the order summary that you want to preview canceling order product summaries from.

Store Output Values

Output Parameter	Description
Cancel Order Product Summary Output	This output is an Apex-defined variable of class ConnectApi.PreviewCancelOutputRepresentation, which contains the values that would populate a change order record for the proposed cancel. The sign of a value in the changeBalances field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in changeBalances and a negative value on a change order record. The orderSummaryId field is the ID of the order summary specified in the input. The changeBalances field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields. • grandTotalAmount—Change to the total with tax. • totalAdjDeliveryAmtWithTax—Change to the adjusted delivery subtotal, including tax. • totalAdjDistAmountWithTax—Change to the total order adjustments, including tax. • totalAdjProductAmtWithTax—Change to the adjusted product subtotal, including tax. • totalAdjustedDeliveryAmount—Change to the adjusted delivery subtotal. • totalAdjustedDeliveryTaxAmount—Change to the adjusted delivery subtotal tax. • totalAdjustedProductAmount—Change to the adjusted product subtotal. • totalAdjustedProductTaxAmount—Change to the adjusted product subtotal tax. • totalAdjustmentDistributedAmount—Change to the total order adjustments. • totalAdjustmentDistributedTaxAmount—Change to the total order adjustments tax. • totalAmount—Change to the pretax total. • totalExcessFundsAmount—The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated

Output Parameter	Description
	with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment is captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. • `totalFeeAmount`—The total amount of the fees charged for the cancellation. • `totalFeeTaxAmount`—The total amount of tax on the fees charged for the cancellation. • `totalRefundableAmount`—The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. • `totalTaxAmount`—Change to the total tax.

Usage

To set up the Cancel Order Product Summary Items Input:

1. If you want to charge fees, use Assignment elements to set the `amount`, `amountType`, `description`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeInputRepresentation` variables.
2. Use Assignment elements to set the `orderItemSummaryId`, `quantity`, `reason`, and `shippingReductionFlag` field values on one or more `ConnectApi.ChangeItemInputRepresentation` variables.
3. If you're charging fees, use Assignment elements to add the `ConnectApi.ChangeItemFeeInputRepresentation` variables to the `changeItemFees` fields on the associated `ConnectApi.ChangeItemInputRepresentation` variables.
4. Use an Assignment element to add the `ConnectApi.ChangeItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.ChangeInputRepresentation` variable.
5. Use the `ConnectApi.ChangeInputRepresentation` variable and the order summary ID in the action input.

In a flow for canceling order product summaries, display the output of this action for the user to review before executing the cancel. When the user verifies the expected results, pass the same input to a Cancel Order Item Summaries Submit action.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Cancel Order Item Summaries Submit

Cancel one or more order product summaries from an order summary. This action creates a change order record.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Cancel Order Item Summaries Submit**.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Input Parameter	Description
`Cancel Order Product Summary Items Input`	This input is an Apex-defined variable of class ConnectApi.ChangeInputRepresentation. The variable has one field: `changeItems`. This field is a list of Apex-defined variables of class ConnectApi.ChangeItemInputRepresentation. Each variable includes these fields: • `changeItemFees`—A list of Apex-defined variables of class ConnectApi.ChangeItemFeeInputRepresentation. Each variable has these fields: – `amount`—Required. Value used to calculate the fee amount, as described by the amountType. It must be a positive value. – `amountType`—Required. Describes how the fee amount is calculated. It can have one of these values: • *`AmountWithTax`*—`amount` is the fee amount, including tax. • *`AmountWithoutTax`*—`amount` is the fee amount, not including tax. Tax is calculated on the value and added. • *`Percentage`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalPrice` and `TotalTaxAmount` of the associated order product summary, prorated for the quantity being returned. • *`PercentageGross`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalLineAmountWithTax` of the associated order product summary, prorated for the quantity being returned. – `description`—Description of the fee. – `priceBookEntryId`—Required unless price books are optional in the org. ID of the price book entry associated with the fee product. – `product2Id`—Required. ID of the product representing the fee. – `reason`—Required. Reason for the fee. The value must match an entry in the Order Product Summary Change object's `Reason` picklist. • `orderItemSummaryId`—Required. ID of an order product summary to cancel. It can't be a shipping charge product. • `quantity`—Required. Quantity to cancel. • `reason`—Required. Cancel reason. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. • `shippingReductionFlag`—Required. Boolean flag that specifies whether to prorate any related delivery charge based on the price change.
`Order Summary Id`	Reference to the order summary that you want to cancel order product summaries from.

Store Output Values

Output Parameter	Description
`Cancel Order Product Summary Output`	This output is an Apex-defined variable of class ConnectApi.SubmitCancelOutputRepresentation. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record. The `changeBalances` field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields. • `grandTotalAmount`—Change to the total with tax. • `totalAdjDeliveryAmtWithTax`—Change to the adjusted delivery subtotal, including tax. • `totalAdjDistAmountWithTax`—Change to the total order adjustments, including tax. • `totalAdjProductAmtWithTax`—Change to the adjusted product subtotal, including tax. • `totalAdjustedDeliveryAmount`—Change to the adjusted delivery subtotal. • `totalAdjustedDeliveryTaxAmount`—Change to the adjusted delivery subtotal tax. • `totalAdjustedProductAmount`—Change to the adjusted product subtotal. • `totalAdjustedProductTaxAmount`—Change to the adjusted product subtotal tax. • `totalAdjustmentDistributedAmount`—Change to the total order adjustments. • `totalAdjustmentDistributedTaxAmount`—Change to the total order adjustments tax. • `totalAmount`—Change to the pretax total. • `totalExcessFundsAmount`—The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment is captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. • `totalFeeAmount`—The total amount of the fees charged for the cancellation. • `totalFeeTaxAmount`—The total amount of tax on the fees charged for the cancellation. • `totalRefundableAmount`—The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. • `totalTaxAmount`—Change to the total tax. The `changeOrderId` field is the ID of the change order record created for the canceled items. Use this change order to create a credit memo. The `feeChangeOrderId` field is the ID of the change order record created for any cancel fees. Use this change order to create an invoice.

Usage

To set up the Cancel Order Product Summary Items Input:

1. If you want to charge fees, use Assignment elements to set the `amount`, `amountType`, `description`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeInputRepresentation` variables.

2. Use Assignment elements to set the `orderItemSummaryId`, `quantity`, `reason`, and `shippingReductionFlag` field values on one or more `ConnectApi.ChangeItemInputRepresentation` variables.
3. If you're charging fees, use Assignment elements to add the `ConnectApi.ChangeItemFeeInputRepresentation` variables to the `changeItemFees` fields on the associated `ConnectApi.ChangeItemInputRepresentation` variables.
4. Use an Assignment element to add the `ConnectApi.ChangeItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.ChangeInputRepresentation` variable.
5. Use the `ConnectApi.ChangeInputRepresentation` variable and the order summary ID in the action input.

In a flow for canceling order product summaries, run a Cancel Order Item Summaries Preview action before running the action. Then display its output for the user to review. When the user verifies the expected results, pass the same input to this action.

SEE ALSO:

Flow Core Action for Order Management: Cancel Order Item Summaries Preview

Add and Edit Elements

Flow Core Action for Order Management: Cancel Order Summary Preview

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Preview the expected results of canceling all order product summaries for an order summary without executing the cancel. The output of this action contains the values that would be set on the change order created by submitting the proposed cancel.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Cancel Order Summary Preview**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cancel All Order Items Input`	This input is an Apex-defined variable of class ConnectApi.CancelAllOrderItemsInputRepresentation, which contains details about the order summary to preview canceling all order products for. The `changeItemFees` field is a list of Apex-defined variables of class ConnectApi.ChangeItemFeeWithTaxInputRepresentation. Each of the variables includes these fields: • `amount`—Positive value used to calculate the fee amount. • `changeItemFees`—List of taxes associated with the change item fees. • `description`—Description of the fee. • `orderDeliveryGroupSummaryId`—ID of the order delivery group summary. • `priceBookEntryId`—ID of the price book entry associated with the fee product. • `product2Id`—ID of the product representing the fee.

Input Parameter	Description
	- `reason`—Reason for the cancellation. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. The `excludedItems` field is a list of items to exclude from the cancellation preview. The `orderSummaryId` field is the ID of the order summary to preview canceling all order products summaries for. The `reason` field is the reason for the cancellation. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. The `reasonText` field is the reason text used for the return insights. The value has a max of 255 characters.

Store Output Values

Output Parameter	Description
`Preview Cancel Output`	This output is an Apex-defined variable of class ConnectApi.PreviewCancelOutputRepresentation, which contains the values that would populate a change order record for the proposed cancel. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record. The `orderSummaryId` field is the ID of the order summary specified in the input. The `changeBalances` field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields. - `grandTotalAmount`—Change to the total with tax. - `totalAdjDeliveryAmtWithTax`—Change to the adjusted delivery subtotal, including tax. - `totalAdjDistAmountWithTax`—Change to the total order adjustments, including tax. - `totalAdjProductAmtWithTax`—Change to the adjusted product subtotal, including tax. - `totalAdjustedDeliveryAmount`—Change to the adjusted delivery subtotal. - `totalAdjustedDeliveryTaxAmount`—Change to the adjusted delivery subtotal tax. - `totalAdjustedProductAmount`—Change to the adjusted product subtotal. - `totalAdjustedProductTaxAmount`—Change to the adjusted product subtotal tax. - `totalAdjustmentDistributedAmount`—Change to the total order adjustments. - `totalAdjustmentDistributedTaxAmount`—Change to the total order adjustments tax. - `totalAmount`—Change to the pretax total. - `totalExcessFundsAmount`—The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund but isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment is captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. - `totalFeeAmount`—The total amount of the fees charged for the cancellation. - `totalFeeTaxAmount`—The total amount of tax on the fees charged for the cancellation.

Output Parameter	Description
	- `totalRefundableAmount`—The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. - `totalTaxAmount`—Change to the total tax.

Usage

To set up the Cancel All Order Items Input:

1. Use Assignment elements to set the `amount`, `amountType`, `changeItemFees`, `description`, `orderDeliveryGroupSummaryId`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeWithTaxInputRepresentation` variables.
2. Use an Assignment element to add the `ConnectApi.ChangeItemFeeWithTaxInputRepresentation` variables to the `changeItemFees` field on a `ConnectApi.CancelAllOrderItemsInputRepresentation` variable.
3. Use the `ConnectApi.CancelAllOrderItemsInputRepresentation` variable and the order summary ID in the action input.

In a flow for canceling all product summaries for an order, display the output of this action for the user to review before executing the cancel. When the user verifies the expected results, pass the same input to a Cancel Order Summary Submit action.

Flow Core Action for Order Management: Cancel Order Summary Submit

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Cancel all order product summaries for an order summary. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Cancel Order Summary Submit**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Cancel All Order Items Input`	This input is an Apex-defined variable of class ConnectApi.CancelAllOrderItemsInputRepresentation, which contains details about the order summary to preview canceling all order products for. The `changeItemFees` field is a list of Apex-defined variables of class ConnectApi.ChangeItemFeeWithTaxInputRepresentation. Each of the variables includes these fields: - `amount`—Positive value used to calculate the fee amount. - `changeItemFees`—List of taxes associated with the change item fees. - `description`—Description of the fee. - `orderDeliveryGroupSummaryId`—ID of the order delivery group summary.

Input Parameter	Description
	- `priceBookEntryId`—ID of the price book entry associated with the fee product. - `product2Id`—ID of the product representing the fee. - `reason`—Reason for the cancellation. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. The `excludedItems` field is a list of items to exclude from the cancellation preview. The `orderSummaryId` field is the ID of the order summary to preview canceling all order products summaries for. The `reason` field is the reason for the cancellation. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. The `reasonText` field is the reason text used for the return insights. The value has a max of 255 characters.

Store Output Values

Output Parameter	Description
`Cancel All Order Items Async Output`	This output is an Apex-defined variable of class ConnectApi.CancelAllOrderItemsAsyncOutputRepresentation, which contains the ID of the asynchronous background operation.

Usage

To set up the Cancel All Order Items Input:

1. Use Assignment elements to set the `amount`, `amountType`, `changeItemFees`, `description`, `orderDeliveryGroupSummaryId`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeWithTaxInputRepresentation` variables.
2. Use an Assignment element to add the `ConnectApi.ChangeItemFeeWithTaxInputRepresentation` variables to the `changeItemFees` field on a `ConnectApi.CancelAllOrderItemsInputRepresentation` variable.
3. Use the `ConnectApi.CancelAllOrderItemsInputRepresentation` variable and the order summary ID in the action input.

In a flow for canceling all product summaries for an order, run a Cancel Order Summary Preview action before running this action. Then display its output for the user to review. When the user verifies the expected results, pass the same input to this action. When the action completes, it generates OSAsyncChgCompletedEvent if successful and ProcessExceptionEvent if not.

Flow Core Action for Order Management: Confirm Held Fulfillment Order Capacity

Confirm held fulfillment order capacity at one or more locations. This action decreases a location's held capacity and increases its assigned fulfillment order count. Confirm held capacity when you assign a fulfillment order to a location.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Confirm Held Fulfillment Order Capacity**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Confirm Held Fulfillment Order Capacity Input`	This input is an Apex-defined variable of class ConnectApi.ConfirmHeldFOCapacityRequestInputRepresentation, which includes these fields: • `allOrNothing` —(Optional) Controls whether a single failed request cancels all other requests in the list (`true`) or some requests can succeed if others fail (`false`). The default value is `false`. • `capacityRequests` —This field is a list of Apex-defined variables of class ConnectApi.CapacityRequestInputRepresentation. Each of the variables represents a request to confirm one fulfillment order assigned to one location, and includes these fields: – `actionRequestId` —Unique string that identifies the request. Can be a UUID. To identify which requests succeeded or failed, use the action request IDs in response data. – `locationId` —ID of the location associated with the request.

Store Output Values

Use output values later in the flow. The values are assigned when the capacity properties are updated.

Output Parameter	Description
`Confirm Held Fulfillment Order Capacity Output`	This output is an Apex-defined variable of class ConnectApi.ConfirmHeldFOCapacityResponseOutputRepresentation, which includes this field: • `capacityResponses` —This field is a list of Apex-defined variables of class ConnectApi.CapacityResponseOutputRepresentation, each of which includes these fields: – `actionRequestId` —Unique string that identifies the original capacity request. – `error` —This field is an Apex-defined variable of class ConnectApi.ErrorResponse, which includes these fields: • `errorCode` —Error code, if the request returned an error. • `message` —More error detail, if available. – `success` —Indicates whether the request was successful (`true`) or not (`false`).

Flow Core Action for Order Management: Create Credit Memo

Create a credit memo to represent the refund for one or more change orders associated with an order summary.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Credit Memo**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Credit Memo Input`	This input is an Apex-defined variable of class ConnectApi.CreateCreditMemoInputRepresentation. The variable has one field, `changeOrderIds`, which is a list of IDs of the change orders to create a credit memo for.
`Order Summary Id`	Reference to the order summary associated with the change orders.

Store Output Values

Output Parameter	Description
`Credit Memo Output`	This value is an Apex-defined variable of class ConnectApi.CreateCreditMemoOutputRepresentation. The `creditMemoId` field contains the ID of the created credit memo.

Usage

To set up the Credit Memo Input, first use Assignment elements to add the change order IDs to a list of strings variable. Then use that variable in the action input.

SEE ALSO:

Flow Core Action for Order Management: Ensure Refunds Async

Add and Edit Elements

Flow Core Action for Order Management: Create Fulfillment Order

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Create one or more fulfillment orders and fulfillment order products for an order delivery group summary, which defines a recipient and delivery method. You specify the order product summaries to fulfill and the fulfillment locations to handle them. If you specify multiple fulfillment locations, a fulfillment order is created for each one.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Fulfillment Order**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Fulfillment Order Input`	This input is an Apex-defined variable of class ConnectApi.FulfillmentOrderInputRepresentation. The variable has three fields: • `fulfillmentGroups` — A list of Apex-defined variables of class ConnectApi.FulfillmentGroupInputRepresentation. A fulfillment order is created for each fulfillment group. A group represents a set of order product summaries to fulfill from a single location, using the same fulfillment type. Each fulfillment group variable has these fields: – `fulfilledFromLocationId` — Reference to the fulfillment location. – `fulfillmentType` — The fulfillment type. Specify one of the values that you defined for the `Type` field picklist on the Fulfillment Order object. – `orderItemSummaries` — A list of Apex-defined variables of class ConnectApi.OrderItemSummaryInputRepresentation. Each variable has these fields: • `orderItemSummaryId` — Reference to an order product summary. • `quantity` — The quantity of the order product summary to allocate to the fulfillment order. – `referenceId` — Reference to the fulfillment group input. This action doesn't use this value. • `orderDeliveryGroupSummaryId` — Reference to the order delivery group summary associated with the order product summaries. • `orderSummaryId` — Reference to the order summary associated with the order product summaries.

Store Output Values

Output Parameter	Description
`Fulfillment Order Output`	This value is an Apex-defined variable of class ConnectApi.FulfillmentOrderOutputRepresentation. The `fulfillmentOrderIds` field is a list of IDs of the created fulfillment orders.

Usage

To set up the Fulfillment Order Input:

1. Use Assignment elements to set the `orderItemSummaryId` and `quantity` field values on one or more `ConnectApi.OrderItemSummaryInputRepresentation` variables for each fulfillment group.
2. Use Assignment elements to add the `ConnectApi.OrderItemSummaryInputRepresentation` variables to the `orderItemSummaries` fields on one or more `ConnectApi.FulfillmentGroupInputRepresentation` variables, one for each fulfillment group.
3. Use Assignment elements to set the `fulfilledFromLocationId` and `fulfillmentType` field values on the fulfillment group variables.
4. Use Assignment elements to add the fulfillment group variables to the `fulfillmentGroups` field on a `ConnectApi.FulfillmentOrderInputRepresentation` variable.
5. Use Assignment elements to set the `orderDeliveryGroupSummaryId` and `orderSummaryId` field values on the `ConnectApi.FulfillmentOrderInputRepresentation` variable.
6. Use the `ConnectApi.FulfillmentOrderInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Create Fulfillment Orders

Create fulfillment orders and fulfillment order products for multiple order delivery group summaries, each of which defines a recipient and delivery method. You specify the order product summaries to fulfill and the fulfillment locations to handle them. If you specify multiple fulfillment locations for one order delivery group summary, a fulfillment order is created for each one.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Fulfillment Orders**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Fulfillment Orders Input`	This input is an Apex-defined variable of class ConnectApi.MultipleFulfillmentOrderInputRepresentation. The variable has one field: `fulfillmentOrders`. This field is a list of Apex-defined variables of class ConnectApi.FulfillmentOrderInputRepresentation. Each variable has three fields:

Input Parameter	Description
	• `fulfillmentGroups` — A list of Apex-defined variables of class ConnectApi.FulfillmentGroupInputRepresentation. A fulfillment order is created for each fulfillment group. A group represents a set of order product summaries to fulfill from a single location using the same fulfillment type. Each fulfillment group variable has these fields: – `fulfilledFromLocationId` — Reference to the fulfillment location. – `fulfillmentType` — The fulfillment type. Specify one of the values that you defined for the `Type` field picklist on the Fulfillment Order object. – `orderItemSummaries` — A list of Apex-defined variables of class ConnectApi.OrderItemSummaryInputRepresentation. Each variable has these fields: • `orderItemSummaryId` — Reference to an order product summary. • `quantity` — The quantity of the order product summary to allocate to the fulfillment order. – `referenceId` — Reference to the fulfillment group input. Use this value to troubleshoot a failure. • `orderDeliveryGroupSummaryId` — Reference to the order delivery group summary associated with the order product summaries. • `orderSummaryId` — Reference to the order summary associated with the order product summaries.

Store Output Values

Output Parameter	Description
`Fulfillment Orders Output`	This value is an Apex-defined variable of class ConnectApi.MultipleFulfillmentOrderOutputRepresentation. The variable has one field: `fulfillmentOrders`. This field is a list of Apex-defined variables of class ConnectApi.FulfillmentGroupOutputRepresentation. Each variable has these fields: • `fulfilledFromLocationId` — Reference to the fulfillment location. This value is included so that you can resubmit the creation if it fails. • `fulfillmentOrderId` — Reference to the created fulfillment order. • `fulfillmentType` — The fulfillment type. This value is included if the creation failed, so you can resubmit it. • `orderDeliveryGroupSummaryId` — Reference to the order delivery group summary associated with the order product summaries. This value is included if the creation failed, so you can resubmit it. • `orderItemSummaries` — A list of Apex-defined variables of class ConnectApi.OrderItemSummaryInputRepresentation. This value is included if the creation failed, so you can resubmit it. Each variable has these fields: – `orderItemSummaryId` — Reference to an order product summary. – `quantity` — The quantity of the order product summary to allocate to the fulfillment order. • `orderSummaryId` — Reference to the order summary associated with the order product summaries. This value is included if the creation failed, so you can resubmit it. • `referenceId` — Reference to the fulfillment group input. Use this value to troubleshoot a failure.

Usage

To set up the Fulfillment Orders Input:

1. For each order delivery group:
   a. Use Assignment elements to set the `orderItemSummaryId` and `quantity` field values on one or more `ConnectApi.OrderItemSummaryInputRepresentation` variables.
   b. Use Assignment elements to add the `ConnectApi.OrderItemSummaryInputRepresentation` variables to the `orderItemSummaries` fields on one or more `ConnectApi.FulfillmentGroupInputRepresentation` variables, one for each fulfillment group.
   c. Use Assignment elements to set the `fulfilledFromLocationId`, `fulfillmentType`, and `referenceId` field values on the `ConnectApi.FulfillmentGroupInputRepresentation` variables.
   d. Use Assignment elements to add the `ConnectApi.FulfillmentGroupInputRepresentation` variables to the `fulfillmentGroups` field on a `ConnectApi.FulfillmentOrderInputRepresentation` variable.
   e. Use Assignment elements to set the `orderDeliveryGroupSummaryId` and `orderSummaryId` field values on the `ConnectApi.FulfillmentOrderInputRepresentation` variable.
2. Use Assignment elements to add the `ConnectApi.FulfillmentOrderInputRepresentation` variables to the `fulfillmentOrders` field on a `ConnectApi.MultipleFulfillmentOrderInputRepresentation` variable.
3. Use the `ConnectApi.MultipleFulfillmentOrderInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Create an Invoice from Change Orders

Create an invoice to represent the charges for one or more change orders. Create invoices for change orders that increase order amounts, such as return fees. When you ensure the refund for a return, include the invoices for the associated return fees in the input.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create an Invoice from Change Orders**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Create Invoice From Change Order Input`	Required. This input is an Apex-defined variable of class ConnectApi.CreateInvoiceFromChangeOrdersInputRepresentation. It has two fields. The `changeOrderIds` field is a list of IDs of the change orders to create an invoice for. The `orderSummaryId` field is the ID of the order summary associated with the change orders.

Store Output Values

Output Parameter	Description
`Invoice Output`	This value is an Apex-defined variable of class ConnectApi.ChangeOrdersInvoiceOutputRepresentation. It has three fields. The `errors` field is a list of Apex-defined variables of class ConnectApi.ErrorResponse containing any errors that were returned. The `invoiceId` field contains the ID of the created invoice. The `success` field indicates whether the transaction was successful.

SEE ALSO:

- Flow Core Action for Order Management: Create Return Order
- Flow Core Action for Order Management: Return Return Order Items
- Flow Core Action for Order Management: Ensure Refunds Async
- Add and Edit Elements

Flow Core Action for Order Management: Create an Invoice from Fulfillment Order

Create an invoice for a fulfillment order that doesn't have one.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create an Invoice from Fulfillment Order**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Fulfillment Order Id`	Reference to the fulfillment order that needs an invoice.

Store Output Values

Output Parameter	Description
`Invoice creation output`	This value is an Apex-defined variable of class ConnectApi.FulfillmentOrderInvoiceOutputRepresentation. The `invoiceId` field contains the ID of the created invoice.

SEE ALSO:

- Flow Core Action for Order Management: Ensure Funds Async
- Add and Edit Elements

Flow Core Action for Order Management: Create Order Payment Summary

Create an order payment summary for a payment authorization or payments that use the same payment method and are attached to the same order summary.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Order Payment Summary**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs. Include at least one payment authorization or list of payments. You don't need both.

Input Parameter	Description
`Order Payment Summary Create Input`	This input is an Apex-defined variable of class ConnectApi.CreateOrderPaymentSummaryInputRepresentation. The variable includes these fields: • `orderSummaryId` — Reference to the order summary associated with the payments. In orgs with the multicurrency feature enabled, the order payment summary inherits the `ISO Currency` value from the order summary. • `paymentAuthorizationId` — Reference to the payment authorization to associate with the summary. • `paymentIds` — List of IDs of the payments to associate with the summary.

Store Output Values

Output Parameter	Description
`Order Payment Summary Output`	This value is an Apex-defined variable of class ConnectApi.CreateOrderPaymentSummaryOutputRepresentation. The `orderPaymentSummaryId` field contains the ID of the created order payment summary.

Usage

To set up the Order Payment Summary Create Input for payments, first use Assignment elements to add the payment IDs to a list of strings variable. Then use that variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Create Order Summary

Create an order summary based on an order. That order is considered the original order for the order summary. Subsequent change orders that apply to the order summary are also represented as order records.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Order Summary**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Order Summary Create Input`	This input is an Apex-defined variable of class ConnectApi.OrderSummaryInputRepresentation. The variable has these fields: • `businessModel`—The order's business model. It can have one of these values: – B2B – B2C • `externalReferenceIdentifier`—Used to prevent duplicate records. This value is case-sensitive. • `name`—Order summary number to assign to the order summary. • `orderId`—Required. The ID of the original order to create an order summary for. • `orderLifeCycleType`—Specifies whether the order is managed in Salesforce Order Management or by an external system. It can have one of these values: – *`MANAGED`*—The order is managed in Salesforce Order Management. If no value is specified, the default is *`MANAGED`*. – *`UNMANAGED`*—The order is managed by an external system. • `sourceProcess`—Describes the order process creating the order summary. It can have one of these values: – *`OrderOnBehalf`*—An Order on Behalf Of process. – *`Standard`*—Any process other than Order on Behalf Of. • `status`—Status to assign to the order summary. The value must match one of the picklist values on the `Status` field of the Order Summary object.

Store Output Values

Output Parameter	Description
`Order Summary Output`	This value is an Apex-defined variable of class OrderSummaryOutputRepresentation. The `orderSummaryId` field contains the ID of the created order summary.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Create Return Order

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management and Returns

Create a return order and return order items for order items belonging to an order summary. You can add return fees for any of the order items.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Create Return Order**.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Return Order Input`	This input is an Apex-defined variable of class ConnectApi.ReturnOrderInputRepresentation. The variable has four fields: • `orderSummaryId`—ID of the order summary containing the items to be returned. The order summary's order lifecycle type must be Managed. • `returnOrderLifeCycleType`—The LifeCycleType of the return order. Possible values are: – *`Managed`*—Process the return order using the actions and APIs. It can generate change orders and affects financial fields and rollup calculations. – *`Unmanaged`*—The return order is for tracking purposes only. It isn't involved in any financial calculations and doesn't generate any change orders. The system doesn't prevent the creation of duplicate return order line items in an unmanaged return order for the same order item. • `returnOrderLineItems`—A list of Apex-defined variables of class ConnectApi.ReturnOrderLineItemInputRepresentation. Each variable has these fields: – `canReduceShipping`—Whether the associated shipping charge can be refunded. – `orderItemSummaryId`—ID of the associated OrderItemSummary. If the OrderItemSummary already has an associated ReturnOrderLineItem, then you must specify a different `reasonForReturn`. Duplicating the reason breaks the financial calculations.

Input Parameter	Description
	– `quantityExpected`—Quantity expected to be returned. – `quantityReceived`—(Optional) Quantity already physically returned. This value isn't used by any standard features, but it's provided for use in customizations. – `reasonForReturn`—(Optional) Reason for the return. The value must match an entry in the ReturnOrderLineItem object's ReasonForReturn picklist. – `returnOrderLineItemFees`—(Optional) A list of Apex-defined variables of class ConnectApi.ReturnOrderLineItemFeeInputRepresentation. Each variable has these fields: • `amount`—Value used to calculate the fee amount, as described by the amountType. It must be a positive value. • `amountType`—Describes how the fee amount is calculated. It can have one of these values: – *`AmountWithTax`*—`amount` is the fee amount, including tax. – *`AmountWithoutTax`*—`amount` is the fee amount, not including tax. Tax is calculated on the value and added. – *`Percentage`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalPrice` and `TotalTaxAmount` of the associated OrderItemSummary, prorated for the quantity being returned. – *`PercentageGross`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalLineAmountWithTax` of the associated OrderItemSummary, prorated for the quantity being returned. • `description`—(Optional) Description of the fee. • `product2Id`—ID of the product representing the fee. • `reason`—Reason for the fee. The value must match an entry in the ReturnOrderLineItem object's `ReasonForReturn` picklist. • `status`—Status to assign to the return order. The value must match one of the picklist values on the Status field of the Return Order object.

Store Output Values

Output Parameter	Description
`Return Order Output`	This value is an Apex-defined variable of class ConnectApi.ReturnOrderOutputRepresentation. The `returnOrderId` field contains the ID of the created return order.

Usage

To set up the Create Return Order Input:

1. Use Assignment elements to set the `canReduceShipping`, `orderItemSummaryId`, `quantityExpected`, `quantityReceived`, and `reasonForReturn` field values on one or more `ConnectApi.ReturnOrderLineItemInputRepresentation` variables.

2. If you want to add any return fees, use Assignment elements to set the `amount`, `amountType`, `description`, `product2Id`, and `reason` field values on one or more `ConnectApi.ReturnOrderLineItemFeeInputRepresentation` variables. The `product2Id` points to a fee product that you created.
3. Use Assignment elements to add the `ConnectApi.ReturnOrderLineItemFeeInputRepresentation` variables to the `returnOrderLineItemFees` fields on the `ConnectApi.ReturnOrderLineItemInputRepresentation` variables representing the associated return order items.
4. Use an Assignment element to add the `ConnectApi.ReturnOrderLineItemInputRepresentation` variables to the `returnOrderLineItems` field on a `ConnectApi.ReturnOrderInputRepresentation` variable.
5. Use Assignment elements to set the `orderSummaryId`, `returnOrderLifeCycleType`, and `status` field values on the `ConnectApi.ReturnOrderInputRepresentation` variable.
6. Use the `ConnectApi.ReturnOrderInputRepresentation` variable in the action input.

SEE ALSO:

Flow Core Action for Order Management: Return Return Order Items

Add and Edit Elements

Flow Core Action for Order Management: Ensure Funds Async

Ensure funds for an invoice, and apply them to it. If needed, capture authorized funds by sending a request to a payment provider. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation so you can track its status. Payment gateway responses appear in the payment gateway log and don't affect the background operation status.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Ensure Funds Async**.

Note: If the action creates a payment, the payment record's ClientContext value isn't predictable. Don't use it in custom logic.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Ensure Funds Async Input`	This input is an Apex-defined variable of class ConnectApi.EnsureFundsAsyncInputRepresentation. The variable has one field: `invoiceId`, which is the ID of the invoice to ensure funds for and apply them to.
`Order Summary Id`	Reference to the order summary associated with the invoice.

Store Output Values

Output Parameter	Description
`Ensure Funds Async Output`	This value is an Apex-defined variable of class EnsureFundsAsyncOutputRepresentation. It only returns the ID of the asynchronous background operation, regardless of whether a call is made to an external payment gateway. It doesn't include any errors from the operation. The `backgroundOperationId` field contains the ID of the background operation.

Usage

This action applies funds to the invoice balance from order payment summaries associated with the specified order summary following this logic:

Note: If multiple order payment summaries have equal `BalanceAmount` values, their order of selection is random.

1. Verify that the invoice balance doesn't exceed the total `BalanceAmount` of all the order payment summaries associated with the order summary.
2. If an order payment summary has a `BalanceAmount` equal to the invoice balance, apply the funds from that order payment summary.
3. If no exact match was found, apply funds from the order payment summary with the largest `BalanceAmount`.
4. If the invoice still has a balance to ensure, repeat steps 2 and 3 until the full balance is ensured or no captured funds remain.
5. If the invoice still has a balance, look for an order payment summary with an authorized amount equal to the remaining invoice balance. If one exists, capture and apply the funds from that order payment summary.
6. If no exact match was found, capture and apply funds from the order payment summary with the largest authorized amount.
7. If the invoice still has a balance to ensure, repeat steps 5 and 6 until the full balance is ensured.

SEE ALSO:

Flow Core Action for Order Management: Create an Invoice from Fulfillment Order

Flow Core Action for Order Management: Ensure Refunds Async

Add and Edit Elements

Flow Core Action for Order Management: Ensure Refunds Async

Ensure refunds for a credit memo or excess funds by sending a request to a payment provider. This action inserts a background operation into an asynchronous job queue and returns the ID of that operation so you can track its status. Payment gateway responses appear in the payment gateway log and don't affect the background operation status.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Ensure Refunds Async**.

Note: If the action creates a refund, the refund record's ClientContext value isn't predictable. Don't use it in custom logic.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Ensure Refunds Async Input`	This input is an Apex-defined variable of class ConnectApi.EnsureReundsAsyncInputRepresentation. The variable has these fields. You must specify `creditMemoId` or `excessFundsAmount`. You can specify both. • `creditMemoId`—The ID of the credit memo to ensure refunds for. • `excessFundsAmount`—The amount of excess funds to apply the refunds against. • `invoicesToPay`—List of invoices for fees that reduce the refund, such as return fees.

Input Parameter	Description
	- `isAllowPartial`—This value controls the behavior when the amounts included in the `sequences` list don't cover the entire refund amount. If this value is false, the default refund logic is applied to ensure the remaining refund amount. If this value is true, the unrefunded balance remains on the credit memo. If you don't specify a `sequences` list, this value is ignored, and the default refund logic is applied. The default is false. - `sequences`—This input is an ordered list of refund amounts and the OrderPaymentSummaries to apply them to. The process traverses this list in order and stops when it has refunded the full amount. It's a list of Apex-defined variables of class SequenceOrderPaymentSummaryInputRepresentation. It contains these fields: – `amount`—Amount of the refund to apply to the OrderPaymentSummary. – `orderPaymentSummaryId`—ID of the OrderPaymentSummary to apply the Amount to.
`Order Summary Id`	Reference to the order summary associated with the credit memo.

Store Output Values

Output Parameter	Description
`Ensure Refunds Async Output`	This value is an Apex-defined variable of class EnsureRefundsAsyncOutputRepresentation. It only returns the ID of the asynchronous background operation, regardless of whether a call is made to an external payment gateway. It doesn't include any errors from the operation. The `backgroundOperationId` field contains the ID of the background operation.

Usage

This action applies the refund to order payment summaries associated with the specified order summary following this logic.

Note: If multiple order payment summaries have equal `AvailableToRefund` amounts, their order of selection is random.

1. Verify that the credit memo balance and excess funds amount don't exceed the total `AvailableToRefund` amount of all the order payment summaries associated with the order summary.
2. If `sequences` is specified, follow these steps.
   a. Traverse the `sequences` list in order and apply the specified refund amounts to the specified order payment summaries.
   b. If the specified credit memo and excess funds are fully refunded, or if `isAllowPartial` is true, then the action stops here.
3. If a credit memo is specified, follow these steps.
   a. If an order payment summary has an `AvailableToRefund` amount matching the credit memo's remaining balance, apply the refund to that payment.
   b. If no exact match was found, apply the refund to the order payment summary with the largest `AvailableToRefund` amount.
   c. If the credit memo has any remaining balance, repeat steps a and b until that balance is fully refunded.
4. If only one OrderPaymentSummary is specified but has multiple payments, follow these steps.

a. If a payment has an amount matching the CreditMemo's remaining balance, apply the refund to that payment.

b. If no exact match was found but one or more payment has a large enough amount to cover the balance, use the payment with the smallest amount.

c. If no single payment has a large enough amount, use multiple payments in descending order of amount. This ensures the fewest payments are used.

5. If an excess funds amount is specified, follow these steps.

   a. Examine those order payment summaries. If one has an `AvailableToRefund` amount matching the excess funds amount, apply the refund to that payment.

   b. If no exact match was found, apply the refund to the order payment summary with the largest `AvailableToRefund` amount.

   c. If any excess funds amount remains, repeat steps a and b until it's fully refunded.

SEE ALSO:

Flow Core Action for Order Management: Create Credit Memo

Flow Core Action for Order Management: Create an Invoice from Change Orders

Flow Core Action for Order Management: Return Return Order Items

Flow Core Action for Order Management: Ensure Funds Async

Add and Edit Elements

Flow Core Action for Order Management: Find Routes with Fewest Splits

Evaluate ordered product quantities against available inventory to determine the smallest combination of locations that can fulfill the order. If multiple combinations of the minimum number of locations can fulfill the order, the action returns multiple options. Optionally, you can specify a maximum allowable number of locations. By default, the action executes up to 1,000,000 times, stopping when it hits 10,000 results.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Find Routes With Fewest Splits**.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Input Parameter	Description
`Order Routing Minimize Shipments Input`	This input is an Apex-defined variable of class ConnectApi.FindRoutesWithFewestSplitsInputRepresentation. The `locationAvailableInventory` field is a list of Apex-defined variables of class ConnectApi.LocationAvailabilityInputRepresentation. Each of the variables represents a fulfillment location to consider and includes these fields: • `externalReferenceId` — External reference ID of the inventory location. • `quantity` — Available quantity of the product. • `stockKeepingUnit` — Stock Keeping Unit (SKU) of the product. The `maximumNumberOfSplits` field is the maximum allowable number of shipment splits. The action doesn't return routing options that involve more than this number of splits.

Input Parameter	Description
	Each split represents an additional shipment. Specifying a maximum of 0 returns only locations that can fulfill the entire order in a single shipment. A maximum of 1 returns combinations of locations that can fulfill the order in one or two shipments, and so on. The `orderedQuantities` field is a list of Apex-defined variables of class ConnectApi.QuantityWithSkuInputRepresentation. Each of the variables represents an ordered product quantity to fulfill, and includes these fields: • `quantity` — Ordered quantity of the product. • `stockKeepingUnit` — SKU of the product.

Store Output Values

Output Parameter	Description
`Order Routing Minimize Shipments Output`	This output is an Apex-defined variable of class ConnectApi.FindRoutesWithFewestSplitsOutputRepresentation, which contains the sets of fulfillment locations that meet the requirements. The variable has one field: `targetLocations`. This field is a list of Apex-defined variables of class ConnectApi.AvailableLocationOutputRepresentation, each of which represents a set of fulfillment locations that can combine to fulfill the ordered products. Each of the variables includes one field: `locations`. This field is a list of the locations in the set.

Usage

To set up the Order Routing Minimize Shipments Input:

1. Use Assignment elements to set the `externalReferenceId`, `quantity`, and `stockKeepingUnit` field values on one or more `ConnectApi.LocationAvailabilityInputRepresentation` variables.
2. Use Assignment elements to set the `quantity` and `stockKeepingUnit` field values on one or more `ConnectApi.QuantityWithSkuInputRepresentation` variables.
3. Use an Assignment element to add the `ConnectApi.LocationAvailabilityInputRepresentation` variables to the `locationAvailableInventory` field on a `ConnectApi.FindRoutesWithFewestSplitsInputRepresentation` variable.
4. Optionally, use an Assignment element to set the `maximumNumberOfSplits` field on the `ConnectApi.FindRoutesWithFewestSplitsInputRepresentation` variable.
5. Use an Assignment element to add the `ConnectApi.QuantityWithSkuInputRepresentation` variables to the `orderedQuantities` field on the `ConnectApi.FindRoutesWithFewestSplitsInputRepresentation` variable.
6. Use the `ConnectApi.FindRoutesWithFewestSplitsInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Use OCI to Find Routes with Fewest Splits

Evaluate ordered product quantities against available inventory to determine the smallest combination of locations that can fulfill the order. If multiple combinations of the minimum number of locations can fulfill the order, the action returns multiple options. Optionally, you can specify a maximum allowable number of locations and a list of locations to exclude from the calculation. This action combines the Omnichannel Inventory Get Availability action and the Order Management Find Routes with Fewest Splits actions. Instead of calling Get Availability and including the output in the Find Routes with Fewest Splits input, call this action and specify a location or location group to fulfill each ordered product. By default, this action executes up to 1,000,000 times, stopping when it hits 10,000 results. This action handles the inventory check.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Find Routes With Fewest Splits Using OCI**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Note: Set the flow's runtime API version to 54.0 or later.

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Find Routes With Fewest Splits Using OCI Input`	This input is an Apex-defined variable of class ConnectApi.FindRoutesWithFewestSplitsUsingOCIInputRepresentation. The `findRoutesWithFewestSplitsUsingOCIInputs` field is a list of Apex-defined variables of class ConnectApi.FindRoutesWithFewestSplitsGroupUsingOCIInputRepresentation. Each of the variables represents one order and includes these fields: • `excludeLocations` —List of locations to exclude from the routing calculations. • `maximumNumberOfSplits` —Maximum allowable number of shipment splits. The action doesn't return routing options that involve more than this number of splits. Each split represents an additional shipment. Specifying a maximum of 0 returns only locations that can fulfill the entire order in a single shipment. A maximum of 1 returns combinations of locations that can fulfill the order in one or two shipments, and so on. • `orderedItems` —A list of Apex-defined variables of class ConnectApi.FindRoutesWithFewestSplitsUsingOCIItemInputRepresentation. Each of the variables represents an ordered product quantity to fulfill and a location or location group, and includes these fields: – `locationGroupIdentifier` —External reference ID of the inventory location or location group. – `quantity` —Ordered quantity of the product. – `stockKeepingUnit` —Stock Keeping Unit (SKU) of the product.

Store Output Values

Use output values later in the flow.

Output Parameter	Description
`Find Routes With Fewest Splits Using OCI Output`	This output is an Apex-defined variable of class ConnectApi.FindRoutesWithFewestSplitsUsingOCIOutputRepresentation, which contains inventory availability data and the sets of fulfillment locations that meet the requirements. The variable has one field: `results`. This field is a list of Apex-defined variables of class ConnectApi.FindRoutesWithFewestSplitsWithInventoryOutputRepresentation, each of which represents the output for one order, and includes these fields: • `inventory` —Inventory availability data for the location groups and locations specified in the input. • `targetLocations` —A list of Apex-defined variables of class ConnectApi.AvailableLocationOutputRepresentation, each of which represents a set of fulfillment locations that can combine to fulfill the ordered products. Each of the variables includes one field `locations`. This field is a list of the locations in the set.

Usage

To set up the Find Routes With Fewest Splits Using OCI Input:

1. Use assignment elements to set the values for the `locationGroupIdentifier`, `quantity`, and `stockKeepingUnit` field values on one or more ConnectApi.FindRoutesWithFewestSplitsUsingOCIItemInputRepresentation variables.
2. Use assignment elements to add the ConnectApi.FindRoutesWithFewestSplitsUsingOCIItemInputRepresentation variables to the `orderedItems` field on a ConnectApi.FindRoutesWithFewestSplitsGroupUsingOCIInputRepresentation variable.
3. Optionally, use an assignment element to set the value for the `maximumNumberOfSplits` field on the ConnectApi.FindRoutesWithFewestSplitsGroupUsingOCIInputRepresentation variable.
4. Use an assignment element to add the ConnectApi.FindRoutesWithFewestSplitsGroupUsingOCIInputRepresentation variable to the `findRoutesWithFewestSplitsUsingOCIInputs` field on a ConnectApi.FindRoutesWithFewestSplitsUsingOCIInputRepresentation variable.
5. Repeat steps 1–4 for each order that you want to include in the action, adding the inputs to the same ConnectApi.FindRoutesWithFewestSplitsUsingOCIInputRepresentation variable.
6. Use the ConnectApi.FindRoutesWithFewestSplitsUsingOCIInputRepresentation variable in the action input.

Flow Core Action for Order Management: Get Fulfillment Order Capacity Values

Get information about the current fulfillment order capacity of one or more locations.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Get Fulfillment Order Capacity Values**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Get Fulfillment Order Capacity Values Input`	This input is an Apex-defined variable of class ConnectApi.GetFOCapacityValuesRequestInputRepresentation, which includes this field: • `locationIds` —List of IDs of the locations to get fulfillment order capacity information for.

Store Output Values

Use output values later in the flow.

Output Parameter	Description
`Get Fulfillment Order Capacity Values Output`	This output is an Apex-defined variable of class ConnectApi.GetFOCapacityValuesOutputRepresentation, which includes this field: • `locations` —This field is a list of Apex-defined variables of class ConnectApi.LocationCapacityOutputRepresentation, each of which includes these fields: – `assigned` —Value of the location's Assigned Fulfillment Order Count. – `capacity` —Value of the location's Fulfillment Order Capacity. This property represents the location's maximum capacity. – `error` —This field is an Apex-defined variable of class ConnectApi.ErrorResponse, which includes these fields: • `errorCode` —Error code, if the request returned an error. • `message` —More error detail, if available. – `heldCapacity` —Number of fulfillment orders that the location is holding capacity for. – `locationId` —ID of the location.

Flow Core Action for Order Management: Hold Fulfillment Order Capacity

Hold capacity to process fulfillment orders at one or more locations. This action increases a location's held capacity. Hold capacity when you plan to assign a fulfillment order to a location.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Hold Fulfillment Order Capacity**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Hold Fulfillment Order Capacity Input`	This input is an Apex-defined variable of class ConnectApi.HoldFOCapacityRequestInputRepresentation, which includes these fields: - `allOrNothing` —(Optional) Controls whether a single failed request cancels all other requests in the list (`true`) or whether some requests can succeed if others fail (`false`). The default value is `false`. - `capacityRequests` —This field is a list of Apex-defined variables of class ConnectApi.CapacityRequestInputRepresentation. Each of the variables represents a request to hold capacity for one fulfillment order at one location, and includes these fields: – `actionRequestId` —Unique string that identifies the request. Can be a UUID. Use the action request IDs in response data to identify which requests succeeded or failed. – `locationId` —ID of the location associated with the request.

Store Output Values

Use output values later in the flow. The values are assigned when the capacity properties are updated.

Output Parameter	Description
`Hold Fulfillment Order Capacity Output`	This output is an Apex-defined variable of class ConnectApi.HoldFOCapacityResponseOutputRepresentation, which includes this field: - `capacityResponses` —This field is a list of Apex-defined variables of class ConnectApi.CapacityResponseOutputRepresentation, each of which includes these fields: – `actionRequestId` —Unique string that identifies the original capacity request. – `error` —This field is an Apex-defined variable of class ConnectApi.ErrorResponse, which includes these fields: - `errorCode` —Error code, if the request returned an error. - `message` —More error detail, if available. – `success` —Indicates whether the request was successful (`true`) or not (`false`).

Flow Core Action for Order Management: Order Routing Rank by Average Distance

Calculate the average distance from sets of inventory locations to an order recipient, and return the sets sorted by that average distance. Use this action to compare the average shipping distances for different sets of locations that can fulfill an order.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Order Routing Rank By Average Distance**.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Input Parameter	Description
`Order Routing Rank By Average Distance Input`	This input is an Apex-defined variable of class ConnectApi.RankAverageDistanceInputRepresentation. The `deliveryCountryCode` field is the country code of the order recipient. The `deliveryPostalCode` field is the postal code of the order recipient. The `distanceUnit` field specifies whether to return average distances in miles or kilometers, respectively. The value can be *mi* or *km*. The `sortResult` field specifies whether to sort the location sets in ascending or descending order by average distance. The value can be *ASC* or *DESC*. The `targetLocations` field is a list of Apex-defined variables of class ConnectApi.TargetLocationInputRepresentation. Each of the variables represents a set of fulfillment locations that can fulfill an order together, and includes one field: `locations`. This field is a list of Apex-defined variables of class ConnectApi.LocationInputRepresentation, each of which represents one location in the list and contains these fields: • `countryCode` — Country code of the location. • `locationIdentifier` — ID of the location. • `postalCode` — Postal code of the location.

Store Output Values

Output Parameter	Description
`Order Routing Rank By Average Distance Output`	This output is an Apex-defined variable of class ConnectApi.RankAverageDistanceOutputRepresentation, which contains the list of fulfillment location sets, sorted by average distance to the order recipient. The `distanceUnit` field is the specified unit of distance. It can be *miles* or *kilometers*. The `results` field is a list of Apex-defined variables of class ConnectApi.AverageDistanceResultOutputRepresentation, each of which includes one field: `distanceCalculation`. It's an Apex-defined variable of class ConnectApi.DistanceCalculationOutputRepresentation, which includes these fields: • `averageDistance` — Average distance from the locations to the order recipient. • `locations` — A list of Apex-defined variables of class ConnectApi.LocationOutputRepresentation, each of which represents a location in the set and includes two fields: – `distance` — Distance from the location to the order recipient. – `locationIdentifier` — ID of the location. • `rank` — This result's rank among all results by average distance to the order recipient.

Usage

To set up the Order Routing Rank By Average Distance Input:

1. Use Assignment elements to set the `countryCode`, `locationIdentifier`, and `postalCode` field values on one or more `ConnectApi.LocationInputRepresentation` variables to represent the locations in a set.

2. Use an Assignment element to add the `ConnectApi.LocationInputRepresentation` variables to the `locations` field on a `ConnectApi.TargetLocationInputRepresentation` variable.
3. Repeat the previous two steps for each set of fulfillment locations.
4. Use an Assignment element to add the `ConnectApi.TargetLocationInputRepresentation` variables to the `targetLocations` field on a `ConnectApi.RankAverageDistanceInputRepresentation` variable.
5. Use Assignment elements to set the `deliveryCountryCode`, `deliveryPostalCode`, `distanceUnit`, and `sortResult` field values on the `ConnectApi.RankAverageDistanceInputRepresentation` variable.
6. Use the `ConnectApi.RankAverageDistanceInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Order Management: Release Held Fulfillment Order Capacity

Release held fulfillment order capacity at one or more locations. This action decreases a location's held capacity without increasing its assigned fulfillment order count. Release held capacity when you cancel assigning a fulfillment order to a location.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Release Held Fulfillment Order Capacity**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Fulfillment Order Location Release Held Capacity Input`	This input is an Apex-defined variable of class ConnectApi.ReleaseHeldFOCapacityRequestInputRepresentation, which includes these fields: • `allOrNothing` —(Optional) Controls whether a single failed request cancels all other requests in the list (*`true`*) or whether some requests can succeed if others fail (*`false`*). The default value is *`false`*. • `capacityRequests` —This field is a list of Apex-defined variables of class ConnectApi.CapacityRequestInputRepresentation. Each of the variables represents a request to release capacity for one fulfillment order at one location, and includes these fields: – `actionRequestId` —Unique string that identifies the request. Can be a UUID. Use the action request IDs in response data to identify which requests succeeded or failed. – `locationId` —ID of the location associated with the request.

Store Output Values

Use output values later in the flow. The values are assigned when the capacity properties are updated.

Output Parameter	Description
Fulfillment Order Location Release Held Capacity Output	This output is an Apex-defined variable of class ConnectApi.ReleaseHeldFOCapacityResponseOutputRepresentation, which includes this field: • capacityResponses —This field is a list of Apex-defined variables of class ConnectApi.CapacityResponseOutputRepresentation, each of which includes these fields: – actionRequestId —Unique string that identifies the original capacity request. – error —This field is an Apex-defined variable of class ConnectApi.ErrorResponse, which includes these fields: • errorCode —Error code, if the request returned an error. • message —More error detail, if available. – success —Indicates whether the request was successful (*true*) or not (*false*).

Flow Core Action for Order Management: Return Order Item Summaries Preview

Preview the expected results of a simple return of one or more order product summaries from an order summary without executing the return. The output of this action contains the values that would be set on the change order created by submitting the proposed return.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Return Order Item Summaries Preview**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
Order Summary Id	Reference to the order summary that you want to preview returning order product summaries from.
Return Order Product Summary Items Input	This input is an Apex-defined variable of class ConnectApi.ChangeInputRepresentation. The variable has one field: changeItems. This field is a list of Apex-defined variables of class ConnectApi.ChangeItemInputRepresentation. Each variable includes these fields: • changeItemFees—A list of Apex-defined variables of class ConnectApi.ChangeItemFeeInputRepresentation. Each variable has these fields: – amount—Required. Value used to calculate the fee amount, as described by the amountType. It must be a positive value. – amountType—Required. Describes how the fee amount is calculated. It can have one of these values: • *AmountWithTax*—amount is the fee amount, including tax.

Input Parameter	Description
	- *AmountWithoutTax*—amount is the fee amount, not including tax. Tax is calculated on the value and added. - *Percentage*—amount is a percentage. The fee amount is amount divided by 100 and then multiplied by the TotalPrice and TotalTaxAmount of the associated order product summary, prorated for the quantity being returned. - *PercentageGross*—amount is a percentage. The fee amount is amount divided by 100 and then multiplied by the TotalLineAmountWithTax of the associated order product summary, prorated for the quantity being returned. – description—Description of the fee. – priceBookEntryId—Required unless price books are optional in the org. ID of the price book entry associated with the fee product. – product2Id—Required. ID of the product representing the fee. – reason—Required. Reason for the fee. The value must match an entry in the Order Product Summary Change object's Reason picklist. - orderItemSummaryId—Required. ID of an order product summary to return. It can't be a shipping charge product. - quantity—Required. Quantity to return. - reason—Required. Return reason. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. - shippingReductionFlag—Required. Boolean flag that specifies whether to prorate any related delivery charge based on the price change.

Store Output Values

Output Parameter	Description
Return Order Product Summary Items Output	This output is an Apex-defined variable of class ConnectApi.PreviewCancelOutputRepresentation, which contains the values that would populate a change order record for the proposed return. The sign of a value in the changeBalances field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in changeBalances and a negative value on a change order record. The orderSummaryId field is the ID of the order summary specified in the input. The changeBalances field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields: - grandTotalAmount—Change to the total with tax. - totalAdjDeliveryAmtWithTax—Change to the adjusted delivery subtotal, including tax. - totalAdjDistAmountWithTax—Change to the total order adjustments, including tax. - totalAdjProductAmtWithTax—Change to the adjusted product subtotal, including tax. - totalAdjustedDeliveryAmount—Change to the adjusted delivery subtotal. - totalAdjustedDeliveryTaxAmount—Change to the adjusted delivery subtotal tax.

Output Parameter	Description
	• `totalAdjustedProductAmount`—Change to the adjusted product subtotal. • `totalAdjustedProductTaxAmount`—Change to the adjusted product subtotal tax. • `totalAdjustmentDistributedAmount`—Change to the total order adjustments. • `totalAdjustmentDistributedTaxAmount`—Change to the total order adjustments tax. • `totalAmount`—Change to the pretax total. • `totalExcessFundsAmount`—The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that is owed as a refund, but it isn't associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment is captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. • `totalFeeAmount`—The total amount of the fees charged for the return. • `totalFeeTaxAmount`—The total amount of tax on the fees charged for the return. • `totalRefundableAmount`—The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action. • `totalTaxAmount`—Change to the total tax.

Usage

To set up the Return Order Product Summary Items Input:

1. If you want to charge fees, use Assignment elements to set the `amount`, `amountType`, `description`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeInputRepresentation` variables.
2. Use Assignment elements to set the `orderItemSummaryId`, `quantity`, `reason`, and `shippingReductionFlag` field values on one or more `ConnectApi.ChangeItemInputRepresentation` variables.
3. If you're charging fees, use Assignment elements to add the `ConnectApi.ChangeItemFeeInputRepresentation` variables to the `changeItemFees` fields on the associated `ConnectApi.ChangeItemInputRepresentation` variables.
4. Use an Assignment element to add the `ConnectApi.ChangeItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.ChangeInputRepresentation` variable.
5. Use the `ConnectApi.ChangeInputRepresentation` variable and the order summary ID in the action input.

In a flow for returning order product summaries, display the output of this action for the user to review before executing the return. When the user verifies the expected results, pass the same input to a Return Order Item Summaries Submit action.

SEE ALSO:

Flow Core Action for Order Management: Return Order Item Summaries Submit

Add and Edit Elements

Flow Core Action for Order Management: Return Order Item Summaries Submit

Return one or more order product summaries from an order summary. This action is a simple return that creates a change order but not a return order.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Return Order Item Summaries Submit**.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management

Input Parameter	Description
`Order Summary Id`	Reference to the order summary that you want to return order product summaries from.
`Return Order Product Summary Items Input`	This input is an Apex-defined variable of class ConnectApi.ChangeInputRepresentation. The variable has one field: `changeItems`. This field is a list of Apex-defined variables of class ConnectApi.ChangeItemInputRepresentation. Each variable includes these fields: • `changeItemFees`—A list of Apex-defined variables of class ConnectApi.ChangeItemFeeInputRepresentation. Each variable has these fields: – `amount`—Required. Value used to calculate the fee amount, as described by the amountType. It must be a positive value. – `amountType`—Required. Describes how the fee amount is calculated. It can have one of these values: • *`AmountWithTax`*—`amount` is the fee amount, including tax. • *`AmountWithoutTax`*—`amount` is the fee amount, not including tax. Tax is calculated on the value and added. • *`Percentage`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalPrice` and `TotalTaxAmount` of the associated order product summary, prorated for the quantity being returned. • *`PercentageGross`*—`amount` is a percentage. The fee amount is `amount` divided by 100 and then multiplied by the `TotalLineAmountWithTax` of the associated order product summary, prorated for the quantity being returned. – `description`—Description of the fee. – `priceBookEntryId`—Required unless price books are optional in the org. ID of the price book entry associated with the fee product. – `product2Id`—Required. ID of the product representing the fee. – `reason`—Required. Reason for the fee. The value must match an entry in the Order Product Summary Change object's `Reason` picklist.

Input Parameter	Description
	- `orderItemSummaryId`—Required. ID of an order product summary to return. It can't be a shipping charge product. - `quantity`—Required. Quantity to return. - `reason`—Required. Return reason. The value must match one of the picklist values on the Reason field of the Order Product Summary Change object. - `shippingReductionFlag`—Required. Boolean flag that specifies whether to prorate any related delivery charge based on the price change.

Store Output Values

Output Parameter	Description
`Return Order Product Summary Items Output`	This output is an Apex-defined variable of class ConnectApi.SubmitReturnOutputRepresentation. The sign of a value in the `changeBalances` field is the opposite of the corresponding value on a change order record. For example, a discount is a positive value in `changeBalances` and a negative value on a change order record. The `changeBalances` field is an Apex-defined variable of class ConnectApi.ChangeItemOutputRepresentation, which includes these fields: - `grandTotalAmount`—Change to the total with tax. - `totalAdjDeliveryAmtWithTax`—Change to the adjusted delivery subtotal, including tax. - `totalAdjDistAmountWithTax`—Change to the total order adjustments, including tax. - `totalAdjProductAmtWithTax`—Change to the adjusted product subtotal, including tax. - `totalAdjustedDeliveryAmount`—Change to the adjusted delivery subtotal. - `totalAdjustedDeliveryTaxAmount`—Change to the adjusted delivery subtotal tax. - `totalAdjustedProductAmount`—Change to the adjusted product subtotal. - `totalAdjustedProductTaxAmount`—Change to the adjusted product subtotal tax. - `totalAdjustmentDistributedAmount`—Change to the total order adjustments. - `totalAdjustmentDistributedTaxAmount`—Change to the total order adjustments tax. - `totalAmount`—Change to the pretax total. - `totalExcessFundsAmount`—The amount of excess funds available on the order payment summaries related to the order summary. It's equal to the captured amount that's owed as a refund, but it's not associated with an invoice or credit memo. Excess funds normally occur when order products are canceled before fulfillment but after payment is captured. This situation isn't common in the US, where funds are normally authorized but not captured until the fulfillment process begins. This value includes all excess funds related to the order summary, not only the funds related to the current action. - `totalFeeAmount`—The total amount of the fees charged for the return. - `totalFeeTaxAmount`—The total amount of tax on the fees charged for the return. - `totalRefundableAmount`—The total amount available to be refunded. It's the sum of the excess funds and any outstanding change order grand total amounts that apply to post-fulfillment changes. This value includes all refundable amounts related to the order summary, not only the amount related to the current action.

Output Parameter	Description
	• `totalTaxAmount`—Change to the total tax. The `changeOrderId` field is the ID of the change order record created for the returned items. Use this change order to create a credit memo. The `feeChangeOrderId` field is the ID of the change order record created for any return fees. Use this change order to create an invoice.

Usage

To set up the Return Order Product Summary Items Input:

1. If you want to charge fees, use Assignment elements to set the `amount`, `amountType`, `description`, `priceBookEntryId`, `product2Id`, and `reason` field values on one or more `ConnectApi.ChangeItemFeeInputRepresentation` variables.
2. Use Assignment elements to set the `orderItemSummaryId`, `quantity`, `reason`, and `shippingReductionFlag` field values on one or more `ConnectApi.ChangeItemInputRepresentation` variables.
3. If you're charging fees, use Assignment elements to add the `ConnectApi.ChangeItemFeeInputRepresentation` variables to the `changeItemFees` fields on the associated `ConnectApi.ChangeItemInputRepresentation` variables.
4. Use an Assignment element to add the `ConnectApi.ChangeItemInputRepresentation` variables to the `changeItems` field on a `ConnectApi.ChangeInputRepresentation` variable.
5. Use the `ConnectApi.ChangeInputRepresentation` variable and the order summary ID in the action input.

In a flow for returning order product summaries, run a Return Order Item Summaries Preview action before running this action. Then display its output for the user to review. When the user verifies the expected results, pass the same input to this action.

SEE ALSO:

Flow Core Action for Order Management: Return Order Item Summaries Preview

Add and Edit Elements

Flow Core Action for Order Management: Return Return Order Items

Process one or more return order line items belonging to a return order. This action creates a change order record for the returned items and makes the processed return order line items read-only. You can include return order fees associated with the return order line items. If you do, a change order record is created for the return fees. If a processed return order line item has a remaining expected quantity, the action creates a separate return order line item representing that quantity.

In Flow Builder, add an Action element to your flow. Select the **Order Management** category, and search for **Return Return Order Items**.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Order Management and Returns

Input Parameter	Description
`Return Order Id`	Reference to the return order that you want to process return order line items from.
`Return Items Input`	This input is an Apex-defined variable of class ConnectApi.ReturnItemsInputRepresentation. It has three fields. The `returnOrderItemDeliveryCharges` field is an optional list of Apex-defined variables of class ConnectApi.ReturnOrderItemDeliveryChargeInputRepresentation. Each variable includes one field: • `returnOrderLineItemId`—ID of a return order line item representing a shipping charge to return. The `returnOrderItemFees` field is an optional list of Apex-defined variables of class ConnectApi.ReturnOrderItemFeeInputRepresentation. Each variable includes these fields: • `quantityReturned`—The quantity of the ReturnOrderLineItem to process. The amount of the fee to charge is determined by multiplying the total fee amount by this value, divided by the quantityExpected. For example, if the fee amount is $10 and the quantityExpected is 2, if the quantityReturned is 1, $5 is charged. This value normally equals the quantity returned of the ReturnOrderLineItem for the returned item that the fee applies to. The value must be greater than 0. If this value plus quantityToCancel is less than the expected return quantity, the remaining quantity to be returned is added to a new ReturnOrderLineItem. • `quantityToCancel`—The quantity of the ReturnOrderLineItem to remove. This value normally equals the quantity canceled of the ReturnOrderLineItem for the returned item that the fee applies to. This value can also be used to cancel a portion of the fee. The value must be 0 or greater. If this value plus quantityReturned is less than the expected return quantity, the remaining quantity to be returned is added to a new ReturnOrderLineItem. • `returnOrderLineItemId`—ID of a return order line item representing a return fee to charge. The `returnOrderItems` field is a list of Apex-defined variables of class ConnectApi.ReturnOrderItemInputRepresentation. Each of the variables includes these fields: • `quantityReceived`—(Optional) The quantity of the return order line item that has been received. The value must be zero or greater. This value isn't used by any standard features, but is provided for use in customizations. • `quantityRejected`—(Optional) The quantity of the return order line item that has been rejected for return. The value must be zero or greater. This value isn't used by any standard features, but is provided for use in customizations. • `quantityReturned`—The quantity of the return order line item that has been returned. The value must be greater than zero. If this value plus quantityToCancel is less than the expected return quantity, then the remaining quantity to be returned is added to a new return order line item. • `quantityToCancel`—(Optional) The quantity of the return order line item to remove because it's not being returned. The value must be zero or greater. If this value plus quantityReturned is less than the expected return quantity, then the remaining quantity to be returned is added to a new return order line item. • `reasonForRejection`—(Optional) The reason why the rejected quantity, if any, was rejected. This value isn't used by any standard features, but is provided for use in customizations. • `returnOrderLineItemId`—The return order line item ID.

Store Output Values

Output Parameter	Description
`Return Items Output`	This output is an Apex-defined variable of class ConnectApi.ReturnItemsOutputRepresentation. It has three fields. The `changeOrderId` field is the ID of the change order record created for the returned item and delivery charges. Use this change order to create a credit memo. The `feeChangeOrderId` field is the ID of the change order record created for the return fees. Use this change order to create an invoice. The `returnLineItemSplits` field is a list of Apex-defined variables of class ConnectApi.ReturnOrderItemSplitLineOutputRepresentation, which includes these fields. After a change order is created for a return order line item, the return order line item is read-only. If this action is used to return a partial quantity, it creates a new "split" return order line item to hold the remaining quantity to be returned. In that case, it returns the IDs of the original and split return order line items in an element of the `returnLineItemSplits` output list property. • `newReturnOrderItemId`—ID of the new return order line item that holds the remaining return quantity. • `originalReturnOrderItemId`—ID of the original return order line item.

Usage

To set up the Return Return Order Items Input:

1. Use Assignment elements to set the `quantityReceived`, `quantityRejected`, `quantityReturned`, `quantityToCancel`, `reasonForRejection`, and `returnOrderLineItemId` field values on one or more `ConnectApi.ReturnOrderItemInputRepresentation` variables.
2. If you want to include a delivery charge, use Assignment elements to set the `returnOrderLineItemId` field value on one or more `ConnectApi.ReturnOrderItemDeliveryChargeInputRepresentation` variables.
3. If you want to include a return fee, use Assignment elements to set the `quantityReturned`, `quantityToCancel`, and `returnOrderLineItemId` field values on one or more `ConnectApi.ReturnOrderItemFeeInputRepresentation` variables.
4. Use an Assignment element to add the `ConnectApi.ReturnOrderItemInputRepresentation` variables to the `returnOrderItems` field on a `ConnectApi.ReturnItemsInputRepresentation` variable.
5. Use an Assignment element to add the `ConnectApi.ReturnOrderItemDeliveryChargeInputRepresentation` variables to the `returnOrderItemDeliveryCharges` field on a `ConnectApi.ReturnItemsInputRepresentation` variable.
6. Use an Assignment element to add the `ConnectApi.ReturnOrderItemFeeInputRepresentation` variables to the `returnOrderItemFees` field on a `ConnectApi.ReturnItemsInputRepresentation` variable.

7. Use the `ConnectApi.ReturnItemsInputRepresentation` variable and the return order ID in the action input.

SEE ALSO:

Flow Core Action for Order Management: Create Return Order

Flow Core Action for Order Management: Create Credit Memo

Flow Core Action for Order Management: Create an Invoice from Change Orders

Flow Core Action for Order Management: Ensure Refunds Async

Add and Edit Elements

Salesforce Omnichannel Inventory Flow Core Actions

Salesforce Omnichannel Inventory provides several core actions for implementing inventory functionality in flows. To add one of these actions to your flow, add an Action element. Then select the **Omnichannel Inventory Service** category, and search for the appropriate action.

These actions use Apex-defined input and output variables that map to input and output classes in the Apex ConnectApi namespace. For more information on using Apex-defined variables in flows, see Considerations for the Apex-Defined Data Type on page 260.

Important: A flow that uses Omnichannel Inventory actions must have a runtime API version of 52.0 or later. If possible, always use the latest API version in your flows.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Flow Core Action for Omnichannel Inventory: Create Reservation

Create one or more inventory reservations at a location or location group.

Flow Core Action for Omnichannel Inventory: Fulfill Reservation

Fulfill one or more inventory reservations at a location.

Flow Core Action for Omnichannel Inventory: Get Availability

Get inventory availability data for one or more products at one or more inventory locations or location groups.

Flow Core Action for Omnichannel Inventory: Release Reservation

Release one or more inventory reservations.

Flow Core Action for Omnichannel Inventory: Transfer Reservation

Transfer one or more inventory reservations between locations or location groups. This action reduces the reserved quantity at the source and increases it at the destination. It doesn't change physical quantities.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Omnichannel Inventory: Create Reservation

Create one or more inventory reservations at a location or location group.

In Flow Builder, add an Action element to your flow. Select the **Omnichannel Inventory Service** category, and search for **Omnichannel Inventory Service Create Reservation**.

Note: Set the flow's runtime API version to 52.0 or later.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Omnichannel Inventory Create Service Reservation Input`	This input is an Apex-defined variable of class ConnectApi.OCICreateReservationInputRepresentation. The variable has these fields. • `actionRequestId`—A UUID that identifies the request. To identify which actions succeeded or failed, use the action request IDs in the output variables. • `allowPartialReservations`—Optional. When *true*, if the system can't create the entire reservation, then it attempts to create a partial reservation. • `createRecords`—A list of up to 100 Apex-defined variables of class ConnectApi.OCICreateReservationSingleInputRepresentation. Each variable has these fields. – `locationGroupIdentifier`—Identifier of the location group at which to reserve inventory. Either `locationGroupIdentifier` or `locationIdentifier` is required, but not both. – `locationIdentifier`—Identifier of the location at which to reserve inventory. Either `locationIdentifier` or `locationGroupIdentifier` is required, but not both. – `quantity`—The quantity of the product to reserve. – `stockKeepingUnit`—The Stock Keeping Unit (SKU) of the product to reserve. • `expirationSeconds`—Optional. A length of time in seconds. If the reservation isn't fulfilled within this amount of time after the reservationTime, then it expires. The maximum value is 14400. • `externalRefId`—Optional The external reference ID. • `reservationTime`—Optional The time at which to record the reservation. Example: 2020-07-24T21:13:00Z

Store Output Values

Output Parameter	Description
`Omnichannel Inventory Service Create Reservation Output`	This value is an Apex-defined variable of class ConnectApi.OCICreateReservationOutputRepresentation. The variable has these fields. • `details`—A list of Apex-defined variables of class ConnectApi.OCICreateReservationSingleOutputRepresentation. Each variable represents one product being reserved and has these fields. – `errorCode`—The error code, if any. – `locationGroupIdentifier`—Identifier of the location group where the inventory is reserved. – `locationIdentifier`—Identifier of the location where the inventory is reserved – `quantity`—The reserved quantity of the product. – `stockKeepingUnit`—The SKU of the reserved product. • `errors`—A list of Apex-defined variables of class ConnectApi.OCICreateReservationErrorOutputRepresentation. Each variable represents a returned error and has these fields. – `errorCode`—The error code. – `message`—Details of the error, if available. • `expirationTime`—The time at which the reservation would expire. • `reservationTime`—The time when the reservation was recorded. • `success`—Indicates whether the reservation succeeded.

To set up the Omnichannel Inventory Create Service Reservation Input:

1. For each product to reserve, use Assignment elements to set the `locationGroupIdentifier` or `locationIdentifier` field, `quantity` field, and `stockKeepingUnit` field values on a `ConnectApi.OCICreateReservationSingleInputRepresentation` variable.
2. Use Assignment elements to add the `ConnectApi.OCICreateReservationSingleInputRepresentation` variables to the `createRecords` field on a `ConnectApi.OCICreateReservationInputRepresentation` variable.
3. Use Assignment elements to set the `actionRequestId`, `allowPartialReservations`, `expirationSeconds`, `externalRefId`, and `reservationTime` field values on the `ConnectApi.OCICreateReservationInputRepresentation` variable.
4. Use the `ConnectApi.OCICreateReservationInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Omnichannel Inventory: Fulfill Reservation

Fulfill one or more inventory reservations at a location.

In Flow Builder, add an Action element to your flow. Select the **Omnichannel Inventory Service** category, and search for **Omnichannel Inventory Service Fulfill Reservation**.

Note: Set the flow's runtime API version to 52.0 or later.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Omnichannel Inventory Service Fulfill Reservation Input`	This input is an Apex-defined variable of class ConnectApi.OCIFulfillReservationInputRepresentation. The variable has one field: `fulfillmentRecords`. This field is a list of up to 100 Apex-defined variables of class ConnectApi.OCIFulfillReservationSingleInputRepresentation. Each variable has these fields. • `actionRequestId`—A UUID that identifies the request. To identify which actions succeeded or failed, use the action request IDs in the output variables. • `externalRefId`—Optional. The external reference ID. • `locationIdentifier`—Identifier of the location at which to fulfill the reserved inventory. • `quantity`—The quantity of the product to fulfill. • `stockKeepingUnit`—The Stock Keeping Unit of the product to fulfill.

Store Output Values

Output Parameter	Description
`Omnichannel Inventory Service Fulfill Reservation Output`	This value is an Apex-defined variable of class ConnectApi.OCIFulfillReservationOutputRepresentation. The variable has these fields. • `errors`—A list of Apex-defined variables of class ConnectApi.OCIFulfillReservationErrorOutputRepresentation. Each variable represents a returned error and has these fields. – `details`—An Apex-defined variable of class ConnectApi.OCIFulfillReservationSingleOutputRepresentation. Each variable represents a returned error and includes the values from the input so you can resubmit them: • `actionRequestId`—A UUID that identifies the failed request. • `externalRefId`—The external reference ID. • `locationIdentifier`—Identifier of the location at which to fulfill the reserved inventory. • `quantity`—The quantity of the product to fulfill.

Output Parameter	Description
	- `stockKeepingUnit`—The Stock Keeping Unit of the product to fulfill. – `errorCode`—The error code. – `message`—Details of the error, if available. - `success`—Indicates whether the fulfillment succeeded.

To set up the Omnichannel Inventory Service Fulfill Reservation Input:

1. For each reservation to fulfill, use Assignment elements to set the `actionRequestId`, `externalRefId`, `locationIdentifier`, `quantity`, and `stockKeepingUnit` field values on a `ConnectApi.OCIFulfillReservationSingleInputRepresentation` variable.
2. Use Assignment elements to add the `ConnectApi.OCIFulfillReservationSingleInputRepresentation` variables to the `fulfillmentRecords` field on a `ConnectApi.OCIFulfillReservationInputRepresentation` variable.
3. Use the `ConnectApi.OCIFulfillReservationInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Omnichannel Inventory: Get Availability

Get inventory availability data for one or more products at one or more inventory locations or location groups.

In Flow Builder, add an Action element to your flow. Select the **Omnichannel Inventory Service** category, and search for **Omnichannel Inventory Service Get Availability**.

Note: Set the flow's runtime API version to 52.0 or later.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Set Input Values

Use values from earlier in the flow to set the inputs.

Input Parameter	Description
`Omnichannel Inventory Service Get Availability Input`	This input is an Apex-defined variable of class ConnectApi.OCIGetInventoryAvailabilityInputRepresentation. The variable has these fields. - `locationGroupIdentifier`—Optional. Can't combine with `locationGroupIdentifiers` or `locationIdentifiers`. The identifier of a location group to retrieve inventory availability data for. Specifying this value retrieves inventory data for all locations belonging to this group. - `locationGroupIdentifiers`—Optional; can't combine with `locationGroupIdentifier` or `locationIdentifiers`. A list

Input Parameter	Description
	of up to 100 identifiers of location groups to retrieve inventory availability data for. • `locationIdentifiers`—Optional; can't combine with `locationGroupIdentifier` or `locationGroupIdentifiers`. A list of up to 100 identifiers of locations to retrieve inventory availability data for. • `stockKeepingUnit`—Optional; can't combine with `stockKeepingUnits`. The SKU of a product to retrieve inventory availability data for. Specifying a SKU with no locations or location groups returns availability data for that SKU at all inventory locations that aren't assigned to location groups. • `stockKeepingUnits`—Optional; can't combine with `stockKeepingUnit`. A list of up to 100 SKUs of products to retrieve inventory availability data for. • `useCache`—Optional. Fetch the inventory data from the cache. The default value is `true`.

Store Output Values

Output Parameter	Description
`Omnichannel Inventory Service Get Availability Output`	This input is an Apex-defined variable of class ConnectApi.OCIGetInventoryAvailabilityOutputRepresentation. The variable has these fields. • `locationGroups`—A list of Apex-defined variables of class ConnectApi.OCILocationGroupAvailabilityOutputRepresentation. Each variable represents availability data for one location group and has these fields. – `inventoryRecords`—A list of Apex-defined variables of class ConnectApi.OCIInventoryRecordOutputRepresentation. Each variable represents the availability of one product and has these fields. • `availableToFulfill`—The Available To Fulfill quantity. • `availableToOrder`—The Available To Order quantity. • `effectiveDate`—The effective date of the inventory. • `futures`—A list of Apex-defined variables of class ConnectApi.OCIFutureInventoryOutputRepresentation. Each variable represents one future restock and has these fields. – `expectedDate`—Date when the future inventory is expected. – `quantity`—Quantity of the future inventory. • `onHand`—The On Hand quantity. • `reserved`—The Reserved quantity. • `safetyStockCount`—The Safety Stock Count. • `stockKeepingUnit`—The SKU of the product. – `locationGroupIdentifier`—The identifier of the location group. • `locations`—A list of Apex-defined variables of class ConnectApi.OCILocationAvailabilityOutputRepresentation. Each variable represents availability data for one location and has these fields.

Output Parameter	Description
	– inventoryRecords—A list of Apex-defined variables of class ConnectApi.OCIInventoryRecordOutputRepresentation. Each variable represents the availability of one product and has these fields. • availableToFulfill—The Available To Fulfill quantity. • availableToOrder—The Available To Order quantity. • effectiveDate—The effective date of the inventory. • futures—A list of Apex-defined variables of class ConnectApi.OCIFutureInventoryOutputRepresentation. Each variable represents one future restock and has these fields. – expectedDate—Date when the future inventory is expected. – quantity—Quantity of the future inventory. • onHand—The On Hand quantity. • reserved—The Reserved quantity. • safetyStockCount—The Safety Stock Count. • stockKeepingUnit—The SKU of the product. – locationIdentifier—The identifier of the location.

To set up the Omnichannel Inventory Service Get Availability Input:

1. Use Assignment elements to set the `locationGroupIdentifier`, `locationGroupIdentifiers`, or `locationIdentifiers` field value, `stockKeepingUnit` or `stockKeepingUnits` field value, and `useCache` field value on a `ConnectApi.OCIGetInventoryAvailabilityInputRepresentation` variable.
2. Use the `ConnectApi.OCIGetInventoryAvailabilityInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Omnichannel Inventory: Release Reservation

Release one or more inventory reservations.

In Flow Builder, add an Action element to your flow. Select the **Omnichannel Inventory Service** category, and search for **Omnichannel Inventory Service Release Reservation**.

Note: Set the flow's runtime API version to 52.0 or later.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Input Parameter	Description
`Omnichannel Inventory Service Release Reservation Input`	This input is an Apex-defined variable of class ConnectApi.OCIReleaseReservationInputRepresentation. The variable has one field: `releaseRecords`. This field is a list of up to 100 Apex-defined variables of class ConnectApi.OCIReleaseReservationSingleInputRepresentation. Each variable has these fields. - `actionRequestId`—A UUID that identifies the request. To identify which actions succeeded or failed, use the action request IDs in the output variables. - `externalRefId`—Optional. The external reference ID. - `locationGroupIdentifier`—Identifier of the location group at which to release the reserved inventory. Either `locationGroupIdentifier` or `locationIdentifier` is required, but not both. - `locationIdentifier`—Identifier of the location at which to release the reserved inventory. Either `locationIdentifier` or `locationGroupIdentifier` is required, but not both. - `quantity`—The quantity of the product to release. - `stockKeepingUnit`—The Stock Keeping Unit of the product to release.

Store Output Values

Output Parameter	Description
`Omnichannel Inventory Service Release Reservation Output`	This value is an Apex-defined variable of class ConnectApi.OCIReleaseReservationOutputRepresentation. The variable has these fields. - `errors`—A list of Apex-defined variables of class ConnectApi.OCIReleaseReservationErrorOutputRepresentation. Each variable represents a returned error and has these fields. – `details`—An Apex-defined variable of class ConnectApi.OCIReleaseReservationSingleOutputRepresentation. Each variable represents a returned error and includes the values from the input so you can resubmit them: - `actionRequestId`—A UUID that identifies the failed request. - `externalRefId`—The external reference ID. - `locationGroupIdentifier`—Identifier of the location group at which to release the reserved inventory. - `locationIdentifier`—Identifier of the location at which to release the reserved inventory. - `quantity`—The quantity of the product to release. - `stockKeepingUnit`—The Stock Keeping Unit of the product to release. – `errorCode`—The error code. – `message`—Details of the error, if available. - `success`—Indicates whether the release succeeded.

To set up the Omnichannel Inventory Service Release Reservation Input:

1. For each reservation to release, use Assignment elements to set the `actionRequestId`, `externalRefId`, `locationGroupIdentifier` or `locationIdentifier`, `quantity`, and `stockKeepingUnit` field values on a `ConnectApi.OCIReleaseReservationSingleInputRepresentation` variable.
2. Use Assignment elements to add the `ConnectApi.OCIReleaseReservationSingleInputRepresentation` variables to the `releaseRecords` field on a `ConnectApi.OCIReleaseReservationInputRepresentation` variable.
3. Use the `ConnectApi.OCIReleaseReservationInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Omnichannel Inventory: Transfer Reservation

Transfer one or more inventory reservations between locations or location groups. This action reduces the reserved quantity at the source and increases it at the destination. It doesn't change physical quantities.

In Flow Builder, add an Action element to your flow. Select the **Omnichannel Inventory Service** category, and search for **Omnichannel Inventory Service Transfer Reservation**.

Note: Set the flow's runtime API version to 52.0 or later.

Set Input Values

Use values from earlier in the flow to set the inputs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Salesforce Omnichannel Inventory

Input Parameter	Description
`Omnichannel Inventory Service Transfer Reservation Input`	This input is an Apex-defined variable of class ConnectApi.OCITransferReservationInputRepresentation. The variable has these fields. • `allOrNothingTransferId`—Optional. Controls whether a single failed transfer cancels all other transfers in the transferRecords list. – To allow some transfers in the transferRecords list to succeed when others fail, don't set this value. – To cancel all the transfers in the transferRecords list when any of them fail, set this value to a UUID. The ID must be unique, but isn't otherwise used. • `transferRecords`—A list of up to 100 Apex-defined variables of class ConnectApi.OCITransferReservationSingleInputRepresentation. Each variable represents an inventory transfer and has these fields. – `actionRequestId`—A UUID that identifies the request. To identify which actions succeeded or failed, use the action request IDs in the output variables. – `externalRefId`—Optional. The external reference ID. – `fromLocationGroupIdentifier`—The identifier of the location group transferring the reservation. Either `fromLocationGroupIdentifier` or `fromLocationIdentifier` is required, but not both. – `fromLocationIdentifier`—The identifier of the location transferring the reservation. Either `fromLocationIdentifier` or `fromLocationGroupIdentifier` is required, but not both.

Input Parameter	Description
	– `ignoreAvailabilityCheck`—If true, force the transfer even if the receiving location doesn't have sufficient available inventory. The default value is false. – `quantity`—The quantity of the product reservation to transfer. – `stockKeepingUnit`—The Stock Keeping Unit (SKU) of the product reservation to transfer. – `toLocationGroupIdentifier`—The identifier of the location group receiving the reservation. Either `toLocationGroupIdentifier` or `toLocationIdentifier` is required, but not both. – `toLocationIdentifier`—The identifier of the location receiving the reservation. Either `toLocationIdentifier` or `toLocationGroupIdentifier` is required, but not both.

Store Output Values

Output Parameter	Description
`Omnichannel Inventory Service Transfer Reservation Output`	This value is an Apex-defined variable of class ConnectApi.OCITransferReservationOutputRepresentation. The variable has these fields. • `errors`—A list of Apex-defined variables of class ConnectApi.OCITransferReservationErrorOutputRepresentation. Each variable represents a returned error and has these fields. – `details`—An Apex-defined variable of class ConnectApi.OCITransferReservationSingleOutputRepresentation. Each variable represents a returned error and includes the fields from the input: • `actionRequestId`—A UUID that identifies the failed request. • `externalRefId`—The external reference ID. • `fromLocationGroupIdentifier`—The identifier of the location group transferring the reservation. • `fromLocationIdentifier`—The identifier of the location transferring the reservation. • `ignoreAvailabilityCheck`—Whether this call ignored availability data at the location that received the reservation. • `quantity`—The quantity of the product reservation to transfer. • `stockKeepingUnit`—The SKU of the product reservation to transfer. • `toLocationGroupIdentifier`—The identifier of the location group intended to receive the reservation. • `toLocationIdentifier`—The identifier of the location intended to receive the reservation. – `errorCode`—The error code. – `message`—Details of the error, if available. • `success`—Indicates whether the transfer succeeded.

To set up the Omnichannel Inventory Service Transfer Reservation Input:

1. For each reservation to transfer, use Assignment elements to set the `actionRequestId`, `externalRefId`, `fromLocationGroupIdentifier` or `fromLocationIdentifier`, `quantity`, `stockKeepingUnit`, and `toLocationGroupIdentifier` or `toLocationIdentifier` field values on a `ConnectApi.OCITransferReservationSingleInputRepresentation` variable.
2. Use Assignment elements to add the `ConnectApi.OCITransferReservationSingleInputRepresentation` variables to the `transferRecords` field on a `ConnectApi.OCITransferReservationInputRepresentation` variable.
3. Use an Assignment element to set the `allOrNothingTransferId` field on the `ConnectApi.OCITransferReservationInputRepresentation` variable.
4. Use the `ConnectApi.OCITransferReservationInputRepresentation` variable in the action input.

SEE ALSO:

Add and Edit Elements

Flow Core Actions: Send Conversation Messages

Send a messaging component to one or more messaging users in enhanced WhatsApp, enhanced Apple Messages for Business, enhanced SMS, or Messaging for In-App.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Messages`*, and select **Send Conversation Messages**.

EDITIONS

Available in: Lightning Experience

Available in: Enterprise, Unlimited, and Developer Editions

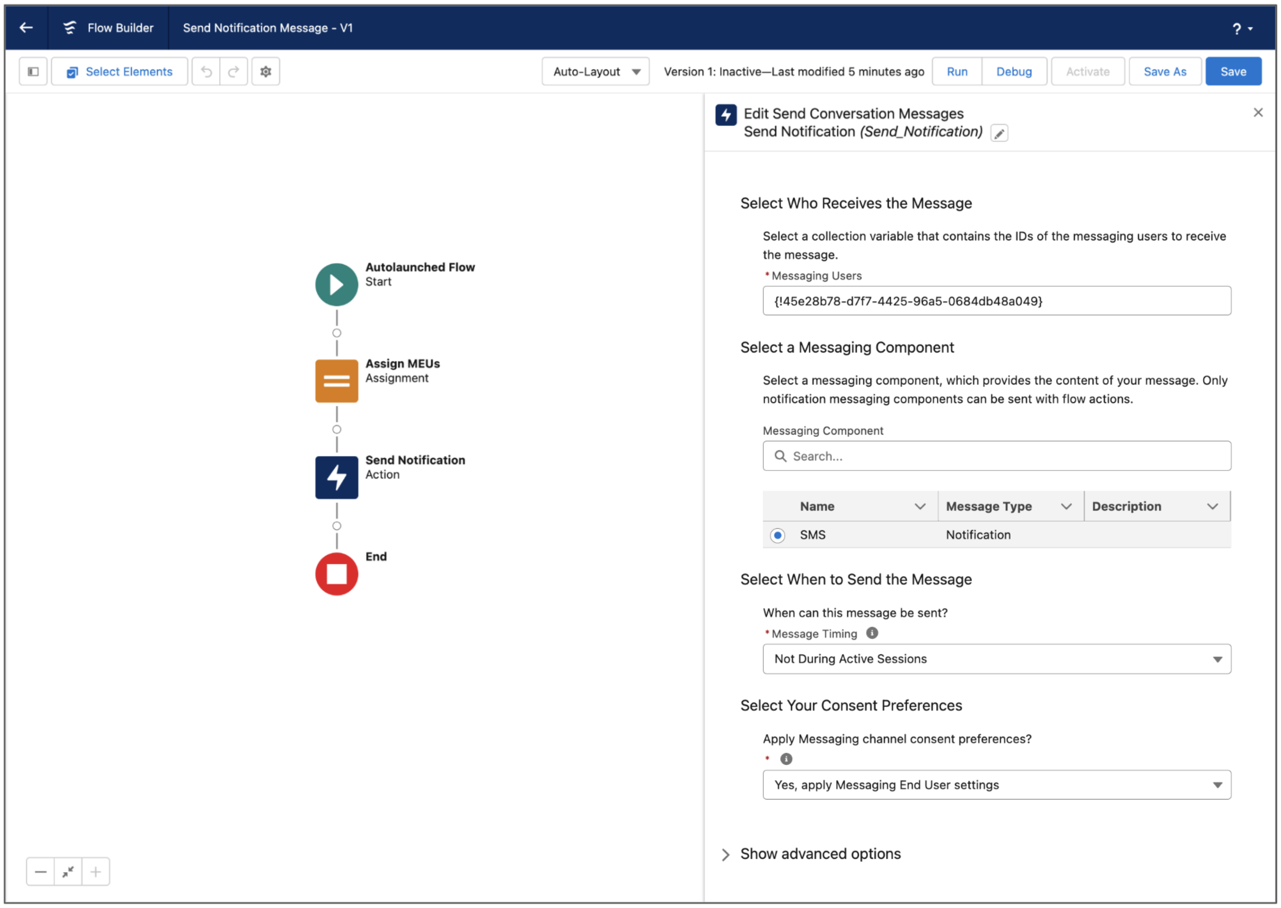
Set Input Values

Field	Description
Messaging Users	Select a collection variable that contains the IDs of up to 100 messaging users to receive the message. To send more than 100 messages, divide your end user recipients into batches of 100 or fewer and repeat the action for each batch.
Messaging Component	Select a messaging component, which provides the content of your message. Only notification messaging components can be sent with flow actions. You can also reference up to 5 custom parameters on the selected messaging component.
Messaging Timing	Optionally, select an option to limit when the message is sent. • Any Time: Send the message regardless of whether the messaging user is engaged in an active messaging session with your company. • Not During Active Sessions: Send the message unless the messaging user is

Field	Description
	engaged in a session with a status of Active. The message is sent when the session's status changes. - Not During Open Sessions: Send the message unless the messaging user is engaged in a session with a status other than Error or Ended. The message is sent when the session's status changes to Error or Ended.
Apply Messaging channel consent preferences	Determine who receives the message. - To respect messaging users' consent settings for a channel, select **Yes, apply Messaging End User settings**. This is the most common approach. - To add custom consent logic, select **No, I'll apply custom consent logic**. For example, if a channel requires users to explicitly opt in, you can select **No, I'll apply custom consent logic** and customize your flow to send the message to users who both implicitly and explicitly opt in. - To send the message only to messaging users who opted into a particular subscription, select **Yes, choose a communication subscription**. In the subsequent field, enter a flow variable that provides the subscription ID. The subscription must be tied to the channel where you're sending the message. This option is visible only if you have a unified channel that supports marketing interactions.

Usage

Here's an example of the Send Conversation Messages action in a simple flow.



Edit Send Conversation Messages

Send Notification *(Send_Notification)*

Select Who Receives the Message

Select a collection variable that contains the IDs of the messaging users to receive the message.

* Messaging Users: {!45e28b78-d7f7-4425-96a5-0684db48a049}

Select a Messaging Component

Select a messaging component, which provides the content of your message. Only notification messaging components can be sent with flow actions.

Messaging Component: Search...

	Name	Message Type	Description
●	SMS	Notification	

Select When to Send the Message

When can this message be sent?

* Message Timing: Not During Active Sessions

Select Your Consent Preferences

Apply Messaging channel consent preferences?

* Yes, apply Messaging End User settings

Show advanced options

To track messages sent by this action, query the ConvMessageSendRequest object.

Flow Core Action: Send Custom Notification

Add the Send Custom Notification action to a flow, then add recipients and content.

Important: The Send Custom Notifications user permission is enforced in orgs created in Winter '21 or later.

The Send Custom Notifications user permission isn't required to trigger the Send Custom Notification action in processes or flows that run in system context.

Tip:

- Before you begin, make sure that the custom notification type you want to call from your process exists. If not, create the notification type.
- To query for the Notification Type ID directly from a flow, add the Get Record element to your flow and filter by API name. If you've installed a notification type via a managed package, filter by the namespace prefix as well as the API name.
- To add recipients, define Recipient ID as a resource. Then add values to your Recipient ID collection by adding the Assignment element to your flow.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Notifications`*, and select **Send Custom Notification**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To trigger the Send Custom Notification action in flows that run in user context, REST API calls, and Apex callouts:

- Send Custom Notifications

Set Input Values

Use values from earlier in the flow to set the inputs for the email. Specify at least one recipient for the email.

Field	Description
`Custom Notification Type ID`	The ID of the Custom Notification Type being used for the notification. This parameter accepts single-value resources of any type. That value is treated as text.
`Notification Body`	The body of the notification that recipients see. The content of mobile push notifications depends on the Display full content push notifications setting. This parameter accepts single-value resources of any type. That value is treated as text and is limited to 750 characters.
`Notification Title`	The title of the notification as seen by recipients. This parameter accepts single-value resources of any type. That value is treated as text and is limited to 250 characters.
`Recipient IDs`	The ID of the recipient or recipient type of the notification. Valid values are: • *`User ID`*—The notification is sent to this user, if this user is active. • *`Account ID`*—The notification is sent to all active users who are members of this account's Account Team. Valid only if account teams are enabled for your org.

Field	Description
	• *`Opportunity ID`*—The notification is sent to all active users who are members of this opportunity's Opportunity Team. Valid only if team selling is enabled for your org. • *`Group ID`*—The notification is sent to all active users who are members of this group. • *`Queue ID`*—The notification is sent to all active users who are members of this queue. This parameter accepts collection variables of type Text and is limited to 500 values. The values that you enter for an individual Send Custom Notification action can represent a total of up to 10,000 users as recipients.
`Target ID`	Optional. The Record ID for the target record of the notification. Specify either a Target ID or a Target Page Reference. This parameter accepts single-value resources of any type. That value is treated as text.
`Target Page Reference`	Optional. The Page Reference for the navigation target of the notification. Specify either a Target ID or a Target Page Reference. This parameter accepts single-value resources of any type. That value is treated as text. To see how to specify the target using JSON, see pageReference.
`Sender ID`	Optional. The User ID of the sender of the notification. This parameter accepts single-value resources of any type. That value is treated as text.

Usage

- Each notification can have up to 10,000 users as recipients. However, you can add an action to the same process within Process Builder or to the same flow in Flow Builder to have more recipients.
- Your org saves your most recent 1 million custom notifications for view in notification trays. Your org can save up to 1.2 million custom notifications, but it trims the amount to the most recent 1 million notifications when you reach the 1.2 million limit.
- An org can execute up to 10,000 notification actions per hour. When you exceed this limit, no more notifications are sent in that hour, and all unsent notifications are lost. Notification actions resume in the next hour.

  For example, your notification action processes are triggered 10,250 times between 4:00 and 4:59. Salesforce executes the first 10,000 of those actions. The remaining 250 notifications aren't sent and are lost. Salesforce begins executing notification actions again at 5:00.

SEE ALSO:

Create and Send Custom Desktop or Mobile Notifications

Flow Run Context

Flow Elements

Add and Edit Elements

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Flow Core Action: Send Email

Send and optionally log an email by specifying the email content and recipients in a flow. If you're using Marketing Cloud Growth, use the Send Email Message on page 377 element to send an email to your audience segment.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

> Note: If you're using Marketing Cloud Growth, use the Send Email Message action instead of the Send Email action. The Send Email action doesn't work with audience segments.

Before you begin:

- Use a Get Records element to get the email template to use, using the Email Template object and filtering by the **Name** (Email Template Name) field.
- Then, in `Email Template ID`, select the ID of the record found by the Get Records. For example, if you labeled your Get Records element *`Get Email Template`*, select **Email Template from Get_Email_Template**.
- Then, select **Id** (Email Template ID).

In Flow Builder, search for *`Send Email`* in the element menu, and select **Send Email**.

> Important: If the Sender Type is OrgWideEmailAddress, ensure that the user running the flow has the proper profile configurations required by the specific org-wide email address being used. Proceeding without the proper configuration results in an error.

Set Input Values

To set the inputs for the email, use values from earlier in the flow. Specify at least one recipient for the email.

> Example: You want to send and log an email to a contact record, and also log to its related account record. For the email content, you want to use an email template with Contact and Account merge fields. Set `Email Template ID` to the ID of the email template to use. Next, set `Log Email on Send` to **{!$GlobalConstant.True}**. Then, set `Recipient ID` to the contact record's ID and `Related Record ID` to the account record's ID.

Input Parameter	Description
`Add Threading Token to Body`	Optional. Indicates whether to create a unique token for the related record and add it to the email body. When the related record is a case record, Email-to-Case uses the token to link future email responses to that case. To link future email responses to other records, create an Apex Email Service and use the `EmailMessages.getRecordIdFromEmail` function to find the record that matches the token.
`Add Threading Token to Subject`	Optional. Indicates whether to create a unique token for the related record and add it to the email subject. When the related record is a case record, Email-to-Case uses the token to link future email responses to that case. To link future email responses to other records, create an Apex Email Service and use the `EmailMessages.getRecordIdFromEmail` function to find the record that matches the token.

Input Parameter	Description
BCC Recipient Address List	Optional. A comma-delimited list of recipient email addresses to send a copy of the email to. Email addresses in the BCC list are hidden from all recipients. This parameter accepts single-value resources of any type. The value is treated as text. The maximum size for this field is 4,000 bytes. You can enter values for BCC Recipient Address List, CC Recipient Address List, Recipient ID, Recipient Address List, and Recipient Address Collection as long as the combined number of recipients is 150 or fewer.
Body	The body of the email. Optional if you're using an email template. The email template overrides the entry in this field. Required if you're not using an email template. Enter text or select a single-value resource of any type that contains your content, for example, a Text Template resource. If entering text, the value is treated as plain text. If you're using a resource, the value can be treated as plain text or rich text, depending on your selection in Rich-Text-Formatted Body.
CC Recipient Address List	Optional. A comma-delimited list of recipient email addresses to send a copy of the email to. This parameter accepts single-value resources of any type. The value is treated as text. The maximum size for this field is 4,000 bytes. You can enter values for BCC Recipient Address List, CC Recipient Address List, Recipient ID, Recipient Address List, and Recipient Address Collection as long as the combined number of recipients is 150 or fewer.
Email Template ID	Optional. The ID of the Classic or Lightning email template to use for the email subject and body. If the email template has merge fields from an object other than the one associated with Recipient ID, specify the record used to supply those merge fields in Related Record ID. If you're using this parameter, Recipient ID is required. This parameter can be used with Log Email on Send. Using email templates in the Send Email action changes the API called by the action, which changes the daily email send limit to the General Email Limit instead of the Daily Workflow Email Limit.
Log Email on Send	Optional. Indicates whether to log the email on the specified records' activity timelines and activity history. Valid values are: • **{!$GlobalConstant.True}**—Log the email to the record associated with Recipient ID, Related Record ID, or both. • **{!$GlobalConstant.False}**—Don't log the email to a record. This value is the default. To log an email, you must specify a value for Recipient ID, Related Record ID, or both. This parameter can be used with Email Template ID. Logging emails with the Send Email action changes the API called by the action, which changes the daily email send limit to the General Email Limit instead of the Daily Workflow Email Limit.

Input Parameter	Description
`Recipient Address Collection`	Optional. A collection of the recipients' email addresses. This parameter accepts collection variables of type Text. If `Log Email on Send` is set to **{!$GlobalConstant.True}**, the email is logged to the ID specified for `Recipient ID`, not the records associated with the email addresses in `Recipient Address Collection`. The maximum size for this field is 4,000 bytes. You can enter values for `BCC Recipient Address List`, `CC Recipient Address List`, `Recipient ID`, `Recipient Address List`, and `Recipient Address Collection` as long as the combined number of recipients is 150 or fewer.
`Recipient Address List`	Optional. A comma-delimited list of the recipients' email addresses. This parameter accepts single-value resources of any type. The value is treated as text. If `Log Email on Send` is set to **{!$GlobalConstant.True}**, the email is logged to the ID specified for `Recipient ID`, not the records associated with the email addresses in `Recipient Address List` The maximum size for this field is 4,000 bytes. You can enter values for `BCC Recipient Address List`, `CC Recipient Address List`, `Recipient ID`, `Recipient Address List`, and `Recipient Address Collection` as long as the combined number of recipients is 150 or fewer.
`Recipient ID`	Optional. The ID of a lead or a contact record. Required if `Email Template ID` is specified. If `Log Email on Send` is included, this parameter is the ID of the person to send and log the email to. If `Email Template ID` is included, this parameter is the ID of the person to send an email to and populate recipient merge fields with. If the ID entered in this parameter is a lead record, you can’t use `Related Record ID`. The maximum size for this field is 4,000 bytes. You can enter values for `BCC Recipient Address List`, `CC Recipient Address List`, `Recipient ID`, `Recipient Address List`, and `Recipient Address Collection` as long as the combined number of recipients is 150 or fewer.
`Related Record ID`	Optional. The ID of a non-recipient record. For example, the ID of a case record. If `Log Email on Send` is included, this parameter is the ID of a secondary record to log the email to. If `Email Template ID` is included, this parameter is the ID of the non-recipient record used to populate email template merge fields. You can’t use this parameter if the ID entered in `Recipient ID` is a lead record.

Input Parameter	Description
`Rich-Text-Formatted Body`	Optional. Indicates whether you want the resource specified for the `Body` parameter to use rich text. Valid values are: • **{!$GlobalConstant.True}**—Use rich text for the email body. • **{!$GlobalConstant.False}**—Use plain text for the email body. This value is the default.
`Sender Email Address`	Optional. The organization-wide email address that's used to send the email. Required when `Sender Type` is set to *`OrgWideEmailAddress`*. Required when the running flow user is the guest user. This parameter accepts a single-value resource of any type. The value is treated as text.
`Sender Type`	Optional. The type of sender that the email is sent from. Valid values are: • *`CurrentUser`*—The email address of the user running the flow. This value is the default. • *`DefaultWorkflowUser`*—The email address of the default workflow user. • *`OrgWideEmailAddress`*—The organization-wide email address that is specified in `Sender Email Address`. When the running flow user is the guest user, the `Sender Email Address` must be set to a verified organization-wide email. Emails sent from the guest user and not using a verified organization-wide email are blocked.
`Subject`	The subject of the email. Optional if you're using an email template. The email template overrides the entry in this field. Required if you're not using an email template. Enter text or select a single-value resource of any type that contains your content, for example, a Text Template resource. The value is treated as plain text.
`Use Line Breaks`	Optional. Indicates whether to render the line breaks in the rich-text-formatted body text template. Valid values are true and false. The default value is false.

Usage

At run time, the email isn't sent until the interview's transaction completes. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element. Before activating your flow, confirm that your org can send email in **Setup** > **Deliverability** > **Access to Send Email (All Email Services)** > **All email**.

If you set Email Deliverability to No Access and:

- If you don't set `Email Template ID` or `Log Email on Send` fields, the flow runs but doesn't send the email.
- If you do set `Email Template ID` or `Log Email on Send` fields, the flow returns an error when it sends the email.

Setup Configurations for Scheduled Flows

If you use the Send Email action element in a Scheduled-Triggered flow, you must configure the organization-wide email address in Setup.

- Set the organization-wide email address in **Setup** > **Email** > **Organization-Wide Email Addresses**
- Add the organization-wide email address in **Setup** > **Process Automation Settings** > **Automated Process User Email Address**

Email Sending Limits

- If you're using Log Email on Send or Email Template ID, the daily email send limit is based on the single email limit. For details, see General Email Limits.
- If you're not using Log Email on Send or Email Template ID, the daily email send limit is based on the daily workflow email limit. For details, see Proactive Alert Monitoring: Daily Workflow Email Limit.

Considerations

- Emails sent using the Send Email action don't include email signatures from My Email Settings. To include a signature, add one to the email template, flow text template, or other resource used in the Send Email action.
- If the Related Record ID is set as a Case ID by the flow, Customer Community users can't create an EmailMessage record. For details, see Experience Cloud User Licenses.

SEE ALSO:

Add and Edit Elements

Options for Sending Emails from Flows

Flow Resource: Text Template

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Options for Sending Emails from Flows

Flow Core Action: Send Notification Actions

Call a notification type to send. Each Send Notification action corresponds to a supported notification type. Send Notification actions are available only for Slack-enabled custom notification types and certain Slack-enabled standard notification types.

Note: To send notifications for Slack, enable Salesforce for Slack Integrations.

To create a custom Slack notification type supported by a Send Notification action, see Create and Send Custom Slack Notifications.

Add an Action element to the flow. In the Action field, select the Send Notification-supported notification type that you want to configure. Each Send Notification action corresponds to a supported notification type. For example, if you created a custom notification type named My Opportunity Notification, look for the My Opportunity Notification action in the Notifications category.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To trigger a Send Notification action in flows that run in user context and REST API calls:

- Send Custom Notifications

Set Input Values

Use values from earlier in the flow to set the inputs for the notification type.

Field	Description
`Recipient IDs`	Required. The IDs of the notification recipients or recipient types. The value must be a collection variable that represents one or more Salesforce User IDs or Collaboration Room IDs.

Field	Description
	Some Salesforce features link standard objects to Collaboration Room through the Swarm object. For these features, you can find an existing Collaboration Room ID from the Swarm object. The collection variable’s Data Type must be Text. The collection can have up to 500 values.
`Record ID`	Required. The ID of the record that the notifications are about. The record ID must be an ID from the Salesforce object related to the notification type. For example, enter the record ID for an opportunity when configuring a notification type associated with the Opportunity object. For custom notification types, you can find the related object by viewing the notification type's settings from Custom Notifications in Setup. For supported standard notification types, refer to the Standard Notification Types and Related Objects table. Enter a record ID or select a variable that identifies the record. This parameter accepts single-value resources of any type. That value is treated as text.

Standard Notification Types and Related Objects

Use this table to identify which object applies to each standard notification type that’s supported by a Send Notification action. The object determines the value that you enter for `Record Id`.

Standard Notification Type	Related Salesforce Object
`Amount Updated`	Opportunity
`Close Date Reminder`	Opportunity
`Close Date Updated`	Opportunity
`Deal Won`	Opportunity
`Deals to Watch`	Opportunity
`High Priority Case`	Case
`New Allergy Intolerance`	Allergy Intolerance
`New Child Opportunity`	Opportunity
`New Care Determinant`	Care Determinant
`New Health Condition`	Health Condition
`New or Updated Care Plan Task`	Task

Standard Notification Type	Related Salesforce Object
`Next Step Reminder`	Opportunity
`Stage Reminder`	Opportunity
`Stage Updated`	Opportunity
`Updated Care Plan`	Case

Usage

- Each notification can have up to 10,000 users as recipients. However, you can add another action to the same flow in Flow Builder to have more recipients.
- You can save up to 1.2 million custom notifications, but notification trays show only your most recent 1 million custom notifications.
- You can execute up to 10,000 notification actions per hour. When you exceed this limit, no more notifications are sent in that hour, and all unsent notifications are lost. Notification actions resume in the next hour.
- The sending rates of Slack notifications are also subject to the limits of the Slack service.

SEE ALSO:

Object Reference for the Salesforce Platform: CollaborationRoom

Object Reference for the Salesforce Platform: Swarm

Flow Core Action: Send Surveys

Create an action to send an active survey by specifying the name, subject, recipients, and invitation link options in the flow.

In Flow Builder, add an Action element to your flow. In the Action field, enter the name of an active survey. Or, in the left navigation, click **Survey**, and then in the Action field, select an active survey. Define the name of the action and the survey recipients.

Note: If you deactivate a survey after it's added to a flow and then activate it, the Flow Builder renders an incorrect Action layout for that survey.

Example: You want to collect feedback from all the participants when a case is closed. First, create a flow and get all records where the status of the case object is closed. Then, create an action that selects the survey to send to the participants for feedback.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Specify at least one recipient for the survey.

Field	Description
`Label`	Name for the action.

Field	Description
`API Name`	Associate an API name for the action. This parameter auto-generates the API name based on the label, which you can edit, if necessary.
`Description`	Optional. Description about the purpose of the action.
`Survey Subject`	Optional. Select a survey subject that you want to perform the action on. For example, to get all case records, select the survey subject as Case, or create a required resource for the subject. This parameter accepts flow variables of type Text.
`Recipient Type`	Select the type of recipient of the survey. Choose the Lead or Contact recipient type only when there's a default Experience Cloud site selected for sending public surveys. This parameter accepts flow variable of type Text.
`Unique link`	Optional. Each participant receives a unique survey invitation. The responses are mapped to the participant name.
`Anonymize responses`	Optional. The responses received aren't mapped to any participant.
`Don't require authentication`	Optional. By default, surveys sent to lead or contact require authentication. However, you can enable this option to allow access to the survey without any authentication.
`Invitation expires in days`	Optional. Define the number of days after which the access to the survey is restricted.

SEE ALSO:

Add and Edit Elements

Flow Core Action: Perform Survey Sentiment Analysis

Get insights into the sentiments that underlie survey responses.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Sentiment`*, and select **Perform Survey Sentiment Analysis**. Or, in the left navigation, click **Survey**, enter *`Sentiment`* in the Action field, and select **Perform Survey Sentiment Analysis**. Define the name of the action and the survey recipients.

To access this action from the API, use the name `performSurveySentimentAnalysis`.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Unlimited**, and **Developer** Editions

Available with Survey Response Pack, Feedback Management - Starter, and Feedback Management - Growth licenses

Set Input Values

Field	Description
`End Date`	Required. The date until when participant responses are processed to get sentiment insights.
`Operation`	Required. The action performed on the AI Sentiment Result records. • **Create**: Use the create operation when sentiment analysis is yet to be done on survey responses and there are no associated AI Sentiment

Field	Description
	Result records, or to analyze the sentiment again. After the processing is completed, AI Sentiment Result records are created with the sentiment of the survey responses and with the Submitted status. • **Update**: Use the update operation to bulk process survey responses that have associated AI Sentiment Result records in Draft status. After the processing is completed, the AI Sentiment Result records are updated with the sentiment of the survey responses and their status is changed to Submitted.
`Question IDs`	Required. The IDs of the questions for whose responses you want to get sentiment insights.
`Start Date`	Required. The date from when participant responses are processed to get sentiment insights.
`Survey ID`	Required. The ID of the survey containing the questions for whose responses you want to get sentiment insights.

Usage

At run time, the AI Sentiment Result record isn't created until the interview's transaction is completed. After the transactions are completed, AI Sentiment Result records are created with Completed status.

Flow Core Action: Get Assessment Response Summary

Create a printable summary view of assessments taken. This action enables you to extract responses saved in an assessment and create a flow to generate a document.

In Flow Builder, add an Action element to your flow. In the Action field, search for Get Assessment Response Summary invocable action to configure.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

Field	Description
`assessmentId`	Required. The ID of the assessment record for which to summarize responses.

Set Output Values

Set Field	Description
`assessmentResponseSummary`	A JSON string containing the summary assessment question texts and responses for the specified assessment record.

Usage

Get Assessment Response Summary makes it easy to use a flow to trigger server-side document generation using Docgen. You can use this invocable action to pass assessment summary data to the downstream processes. This invocable action provides a summary JSON that can be consumed in Docgen workflows to generate documents.

The Get Assessment Response Summary invocable action takes assessment ID as the input to get the OmniProcess ID, which is used to retrieve the OmniProcess elements. The assessment ID also retrieves the assessment response and merges the response with the OmniProcess elements to create an assessment summary response in JSON.

DocGen Limitations

OmniScript doesn't provide a modification history of the same OmniScript form, such as the addition or removal of questions. It's recommended that you trigger the document generation when you submit an assessment. The summary API fetches the layout data from the active version of the OmniScript.

DocGen has the following limits:

- Token data is limited to 131,072 characters.
- Server-side document generation - Maximum supported document size is 1 MB.
- Client-side document generation - Maximum supported document size is 10 MB.
- There's no image-type support for server-side document generation. Image-type support is only available on the client-side.

Slack Flow Core Actions

Manage Slack channels, channel members, and messages from flows. As your Salesforce records change, a flow can trigger changes in Slack.

Important: Slack core actions execute in user context. The flow has access to whatever the running user of the flow has access to.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. Select the **Slack** category, and search for an action.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Core Actions for Slack: Archive Slack Channel
Archive a Slack channel in a Slack workspace.

Flow Core Actions for Slack: Check If Users Are Connected to Slack
Determine whether a collection of Salesforce users is connected to a given Slack workspace.

Flow Core Actions for Slack: Create Slack Channel
Create a Slack channel in a Slack workspace.

Flow Core Actions for Slack: Edit Slack Message
Edit a message that was previously sent to Slack.

Flow Core Actions for Slack: Get Information About Slack Conversation
Retrieve the name of a Slack channel and find out whether it's archived. Archived channels are closed to new activity, but users can still view and search an archived channel's message history.

Flow Core Actions for Slack: Invite Users to Slack Channel
Add users who are connected to a given Slack app to a Slack channel or direct message.

Flow Core Actions for Slack: Pin or Unpin Slack Message
Pin or unpin a message in a Slack channel or direct message. Pin messages so that they're readily available from the conversation header.

Flow Core Actions for Slack: Send Slack Message
Send a message to a Slack channel, direct message, or the Messages tab of a Slack app.

Flow Core Actions for Slack: Send Message to Launch Flow

Send a message to a Slack channel, direct message, or the Messages tab of a Slack app that includes a button that a recipient can use to launch a screen flow.

Flow Core Actions for Slack: Archive Slack Channel

Archive a Slack channel in a Slack workspace.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Archive Slack Channel**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

The flow sends the connection values that you provide to Slack to retrieve an access token.

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value. The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Channel

Input Parameter	Description
`Slack Channel ID`	Required. The ID of the channel to archive. Get the Slack channel ID by logging in to Slack.com and launching Slack in your browser. The channel ID is the last parameter in the URL. For example, in this URL, the channel ID is `C56789FGHIJ`:

Input Parameter	Description
	`https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

- Enable Salesforce for Slack Integrations
- *Salesforce Admins*: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Check If Users Are Connected to Slack

Determine whether a collection of Salesforce users is connected to a given Slack workspace.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Check If Users Are Connected to Slack**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID. You can obtain the Slack workspace ID by logging in to Slack.com and launching Slack in your browser. The workspace ID is the penultimate parameter in the URL. For example, in this URL, the workspace ID is `T01234ABCDE`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Salesforce User ID Collection Resource`	Required. The collection resource that contains the Salesforce user IDs to check. The maximum number of user IDs is 1,000.

Store Output Values

Output Parameter	Description
`Collection of Salesforce User IDs Connected to Slack`	A collection resource that contains the Salesforce user IDs connected to Slack.
`Collection of Salesforce User IDs Not Connected to Slack`	A collection resource that contains the Salesforce user IDs not connected to Slack.

Usage

This action is available only if you enable the connection to Slack in Setup. Otherwise, the action fails. Additionally, the user that initiates the flow and any users impacted by the action must have logged in to a Salesforce Slack app at least once.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Create Slack Channel

Create a Slack channel in a Slack workspace.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Create Slack Channel**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

The flow sends the connection values that you provide to Slack to retrieve an access token.

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value. The Slack app must be a member of the conversation to execute the action on.

Input Parameter	Description
	- User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Channel Details

Input Parameter	Description
`Slack Channel Name`	Required. The name of the new channel. Specify a value or select a resource.
`Channel Type`	Select a value or Boolean resource. Valid values are: - Public - Private - Resource If you select a Boolean resource that evaluates to true, the channel type is private. If you select a Boolean resource that evaluates to false, the channel type is public. The default channel type is public.
`Slack Workspace ID for Channel`	Indicates whether to associate the new channel with a different workspace ID than the workspace ID of the Slack app. If you turn on this option, select a value or resource.

Store Output Values

OUTPUT Parameter	Description
`Slack Channel ID`	The ID of the new channel.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Edit Slack Message

Edit a message that was previously sent to Slack.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Edit Slack Message**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Input Values

The flow sends the connection values that you provide to Slack to retrieve an access token.

Input Parameter	Description
`Slack App ID for Token`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID. The Slack app must be a member of the conversation that contains the message to edit.
`Slack Conversation ID`	Required. The ID of the channel or the direct message to send the message to. Alternatively, specify a Slack user ID if the message was sent to the user via the Messages tab of the Slack app. Enter a value or select a resource. You can obtain the Slack conversation ID by logging in to Slack.com and launching Slack in your browser. The conversation ID is the last parameter in the URL. For example, in this URL, the conversation ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Slack Message`	Required. The message to send. Use alongside Post Message action. For best results, include no more than 4,000 characters. Slack truncates messages containing more than 40,000 characters. Enter a value or select a resource. This action only supports editing messages with standard markdown formatting. Slack supports text formatting with Slack `mrkdown`. To disable formatting for a plain text message that contains Slack `mrkdown`, use an escape sequence. Slack doesn’t support text formatting with HTML and renders rich text messages as plain text.
`Slack Message Timestamp`	Required. The timestamp of the message sent. Enter a value or select a resource. For example, enter *`1234567890.123456`*. The numerals before the period character (.) specify a Unix timestamp. The numerals after the period character specify microseconds.

Input Parameter	Description
	You can store the Slack Message Timestamp output parameter of the Send Slack Message, Edit Slack Message, or Send Message To Launch Flow action as a resource to use later.
`Slack Workspace ID for Token`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID. You can obtain the Slack workspace ID by logging in to Slack.com and launching Slack in your browser. The workspace ID is the penultimate parameter in the URL. For example, in this URL, the workspace ID is `T01234ABCDE`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Get Information About Slack Conversation

Retrieve the name of a Slack channel and find out whether it's archived. Archived channels are closed to new activity, but users can still view and search an archived channel's message history.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Get Information About Slack Conversation**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

The flow sends the connection values that you provide to Slack to retrieve an access token.

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value.

Input Parameter	Description
	The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Conversation

Input Parameter	Description
`Slack Conversation ID`	Required. The ID of the channel to retrieve information about. You can obtain the Slack conversation ID by logging in to Slack.com and launching Slack in your browser. The conversation ID is the last parameter in the URL. For example, in this URL, the conversation ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`

Store Output Values

Output Parameter	Description
`Conversation Is Archived`	Indicates whether the conversation is archived.
`Conversation Is Shared Externally`	Indicates whether the conversation is shared with people outside of your org that aren't part of your Enterprise Grid in Slack.
`Slack Conversation ID`	The ID of the Slack conversation that you retrieved information about.
`Slack Conversation Name`	The name of the Slack conversation that you retrieved information about.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Invite Users to Slack Channel

Add users who are connected to a given Slack app to a Slack channel or direct message.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Invite Users to Slack Channel**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value. The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Channel Details

Use values from earlier in the flow to set the inputs for the action.

Input Parameter	Description
`Slack Channel ID`	Required. The ID of the channel or direct message to invite users to. You can obtain the Slack channel ID by logging in to Slack.com and launching Slack in your browser. The channel ID is the last parameter in the URL. For example, in this URL, the channel ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`

Input Parameter	Description
`Slack Workspace ID for Channel`	Required. The Slack workspace that contains the channel. Select a value or resource. The input value evaluates to the Slack workspace ID. You can obtain the Slack workspace ID by logging in to Slack.com and launching Slack in your browser. The workspace ID is the penultimate parameter in the URL. For example, in this URL, the workspace ID is `T01234ABCDE`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Salesforce User ID Collection Resource`	The collection resource that contains the Salesforce user IDs to invite to the channel. The maximum number of user IDs is 1,000.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails. Additionally, the user that initiates the flow and any users impacted by the action must have logged in to a Salesforce Slack app at least one time.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Pin or Unpin Slack Message

Pin or unpin a message in a Slack channel or direct message. Pin messages so that they're readily available from the conversation header.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Pin or Unpin Slack Message**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value.

Input Parameter	Description
	The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Message Details

Input Parameter	Description
`Slack Conversation ID`	Required. The ID of the channel or group direct message to send the message to. Enter a value or select a resource. You can obtain the Slack conversation ID by logging in to Slack.com and launching Slack in your browser. The conversation ID is the last parameter in the URL. For example, in this URL, the conversation ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Slack Message Timestamp`	Required. The timestamp of the sent message. Enter a value or select a resource. For example, enter *`1234567890.123456`*. The numerals before the period character (.) specify a Unix timestamp. The numerals after the period character specify microseconds. You can store the Slack Message Timestamp output parameter of the Send Slack Message, Edit Slack Message, or Send Message To Launch Flow action as a resource to use later.
`Pin or Unpin Message`	Select a value or Boolean resource. Valid values are: **Pin** Pins the message to the conversation header. **Unpin** Unpins the message from the conversation header. If you select a Boolean value that evaluates to true, the action pins the message. If you select a Boolean value that evaluates to false, the action unpins the message. The default is Pin.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Send Slack Message

Send a message to a Slack channel, direct message, or the Messages tab of a Slack app.

Before using a core action for Slack, enable Salesforce for Slack integrations.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select **Send Slack Message**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID.
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value. The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Message Details

Input Parameter	Description
`Slack Conversation ID`	Required. The ID of the channel or direct message to send the message to. Alternatively, specify a Slack user ID to send the message to the user via the Messages tab of the Slack app. Enter a value or select a resource. You can obtain the Slack conversation ID by logging in to Slack.com and launching Slack in your browser. The conversation ID is the last parameter in the URL. For example, in this URL, the conversation ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`

Input Parameter	Description
`Slack Message`	Required. The message to send. For best results, include no more than 4,000 characters. Slack truncates messages containing more than 40,000 characters. Enter a value or select a resource. Slack supports text formatting with Slack `mrkdown`. To disable formatting for a plain text message that contains Slack `mrkdown`, use an escape sequence. Slack doesn't support text formatting with HTML and renders rich text messages as plain text.
`Salesforce Record ID`	The record ID to send to the view. Defining a view is a pilot feature. For more information, see Define a View in the *Apex SDK for Slack (Pilot) Guide*.
`Slack Message Timestamp`	The timestamp of the Slack message. Specify a timestamp to start a Slack thread. Enter a value or select a resource. For example, enter *`1234567890.123456`*. The numerals before the period character (.) specify a Unix timestamp. The numerals after the period character specify microseconds. You can store the Slack Message Timestamp output parameter of the Send Slack Message, Edit Slack Message, or Send Message To Launch Flow action as a resource to use later.
`View API Name`	The API name of the view that the Slack message is sent with. Defining a view is a pilot feature. For more information, see Define a View in the *Apex SDK for Slack (Pilot) Guide*.

Store Output Values

OUTPUT Parameter	Description
`Slack Message Timestamp`	The timestamp of the sent message.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

- Enable Salesforce for Slack Integrations
- *Salesforce Admins*: How Admins Can Connect Salesforce and Slack

Flow Core Actions for Slack: Send Message to Launch Flow

Send a message to a Slack channel, direct message, or the Messages tab of a Slack app that includes a button that a recipient can use to launch a screen flow.

In Flow Builder, add an Action element to your flow. In the New Action window, select **Slack**, and then select the name of the flow to send.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Set Connection Values for Slack

Input Parameter	Description
`Slack App`	Required. The Slack app that executes the action. Only Slack apps that are installed for the org are available. The input value evaluates to the Slack app ID.
`Slack Workspace`	Required. The Slack workspace where the Slack app is installed. Select a value or resource. The input value evaluates to the Slack workspace ID. You can obtain the Slack workspace ID by logging in to Slack.com and launching Slack in your browser. The workspace ID is the penultimate parameter in the URL. For example, in this URL, the workspace ID is `T01234ABCDE`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Execute Action As`	The entity that executes the action. Valid values are: • Slack App—Execute the action as the Slack app that you selected in the Slack App field. It's the default value. The Slack app must be a member of the conversation to execute the action on. • User Who Runs the Flow—Execute the action as the user who runs the flow. The user can execute the action only when the flow runs in the user context. If the flow runs in the system context, the Slack app executes it. The user must be a member of the conversation to execute the action on.

Set Slack Message Details

Input Parameter	Description
`Slack Conversation ID`	Required. The ID of the channel or the direct message to send the message to. Alternatively, specify a Slack user ID to send the message to the user via the Messages tab of the Slack app. Enter a value or select a resource. You can obtain the Slack conversation ID by logging in to Slack.com and launching Slack in your browser. The conversation ID is the last parameter in the URL. For example, in this URL, the conversation ID is `C56789FGHIJ`: `https://app.slack.com/client/T01234ABCDE/C56789FGHIJ`
`Slack Message`	Required. The message to send. For best results, include no more than 4,000 characters. Slack truncates messages containing more than 40,000 characters. Enter a value or select a resource. The message to send can't be edited. Using the Edit Message action or manual editing results in process failures. Slack supports text formatting with Slack `mrkdown`. To disable formatting for a plain text message that contains Slack `mrkdown`, use an escape sequence. Slack doesn't support text formatting with HTML and renders rich text messages as plain text.
`Button Label`	Required. The label for the button that appears below the message. The user clicks the button to launch the flow from Slack. Slack supports text formatting with Slack `mrkdown`. To disable formatting for a plain text message that contains Slack `mrkdown`, use an escape sequence. Slack doesn't support text formatting with HTML and renders rich text messages as plain text.
`Slack Bot Name`	The name of the bot that sends the message in Slack. Enter a value or select a resource.
`Slack Message Timestamp`	The timestamp of the Slack message. Specify a timestamp to start a Slack thread. Enter a value or select a resource. For example, enter *`1234567890.123456`*. The numerals before the period character (.) specify a Unix timestamp. The numerals after the period character specify microseconds. You can store the Slack Message Timestamp output parameter of the Send Slack Message, Edit Slack Message, or Send Message To Launch Flow action as a resource to use later.

Store Output Values

Input Parameter	Description
`Slack Message Timestamp`	The timestamp of the message sent.

Usage

This action is available only if you enable the connection to Slack in Setup and the user who runs the flow is connected to Slack. Otherwise, the action fails.

SEE ALSO:

Enable Salesforce for Slack Integrations

Salesforce Admins: How Admins Can Connect Salesforce and Slack

Flow Core Action: Submit for Approval

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Submit one Salesforce record for approval.

Tip: Before you begin, store the ID for the record that you want to submit for approval in a variable.

In Flow Builder, add an Action element to your flow. In the Action field, enter *`Submit`*, and select **Submit for Approval**.

Set Input Values

Use values from earlier in the flow to set the inputs for the approval request.

Input Parameter	Description
`Record ID`	The ID of the record that you want to submit for approval. This parameter accepts single-value resources of any type. That value is treated as text.
`Approval Process Name or ID`	The unique name or ID of the approval process that you want to submit the record to. The process must have the same object type as the record you specified in `Record ID`. Required if `Skip Entry Criteria` is set to *`true`*. If this parameter and `Submitter ID` aren't set, the flow succeeds only when: Make sure that: • The approver on submit is determined automatically, and • The user who launched the flow is an allowed initial submitter • The approver on submit is determined automatically. • The initial submitters for the approval processes related to this object include all users who could launch this flow. This parameter accepts single-value resources of any type. That value is treated as text.
`Next Approver IDs`	The ID of the user to be assigned the approval request when the approval process doesn't assign the approver. This parameter accepts collection variables of type Text that include exactly one item.
`Skip Entry Criteria`	If set to *`true`*, the record isn't evaluated against the entry criteria set on the process that is defined in `Approval Process Name or ID`. This parameter accepts any single-value resource of type Boolean.

Input Parameter	Description
`Submission Comments`	Text that you want to accompany the submission. Don't reference merge fields or formula expressions. Submission comments appear in the approval history for the specified record. This text also appears in the initial approval request email if the template uses the `{!ApprovalRequest.Comments}` merge field. This parameter accepts single-value resources of any type. That value is treated as text.
`Submitter ID`	The ID for the user who submitted the record for approval. The user receives notifications about responses to the approval request. The user must be one of the allowed submitters for the process. If you don't set this field, the user who launched the flow is the submitter. If a workflow rule triggers a flow that includes this element, the submitter is the user who triggered the workflow rule. Workflow rules can be triggered when a user creates or edits a record. When the record is approved or rejected, the user who launched the flow or triggered the workflow rule is notified. This parameter accepts single-value resources of any type. That value is treated as text.

Store Output Values

To use the approval request's outputs later in the flow, store them in variables. The values are assigned when the approval request is created.

Optional Output Parameter	Description
`Instance ID`	The ID of the approval request that was submitted. This parameter accepts single-value variables of type Text, Picklist, or Multi-Select Picklist.
`Instance Status`	The status of the current approval request. Valid values are Approved, Rejected, Removed, or Pending. This parameter accepts single-value variables of type Text, Picklist, or Multi-Select Picklist.
`New Work Item IDs`	The IDs of the new items submitted to the approval request. There can be 0 or 1 approval processes. This parameter accepts collection variables of type Text.
`Next Approver IDs`	The IDs of the users who are assigned as the next approvers. This parameter accepts collection variables of type Text.
`Record ID`	The ID of the record that the flow submitted for approval. This parameter accepts single-value variables of type Text, Picklist, or Multi-Select Picklist.

Usage

At run time, the approval request isn't created until the interview's transaction is completed. Transactions are complete when the interview either finishes or executes a Screen, Local Action, or Wait element.

SEE ALSO:

Flow Elements

Add and Edit Elements

Customize What Happens When a Flow Fails

Move and Connect Elements to Change a Flow Route

Salesforce Anywhere Core Flow Actions (Beta)

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Salesforce Anywhere provides several core actions for implementing Salesforce Anywhere functionality in flows. To add one of these actions to your flow, add an Action element. Then select the Salesforce Anywhere category, and search for the appropriate action.

Note: Salesforce Anywhere Beta is a Non-GA Service and not a "Service" or part of the "Services", as defined in the Main Services Agreement ("MSA") with Salesforce. Such Non-GA Service is subject to the terms and conditions of the Universal Pilot Research Agreement ("UPRA"), including the Data Processing Addendum to the UPRA. Use of this Non-GA Service is at your sole discretion, and any purchase decisions are made only on the basis of Salesforce generally available products and features.

These actions are available when you enable Salesforce Anywhere.

Flow Core Action for Salesforce Anywhere: Create a Salesforce Anywhere Chat (Beta)
Create a Salesforce Anywhere chat by specifying participants, and optionally, an initial message and chat title.

Flow Core Action for Salesforce Anywhere: Add a Message to a Salesforce Anywhere Chat (Beta)
Add a message to an existing Salesforce Anywhere chat by specifying the chat URL and message content.

Flow Core Action for Salesforce Anywhere: Add Users to a Salesforce Anywhere Chat (Beta)
Add users to an existing Salesforce Anywhere chat by specifying the chat URL and the users to be added.

Flow Core Action for Salesforce Anywhere: Send Salesforce Anywhere Alerts to Users (Beta)
Notify users about Salesforce Anywhere chat by specifying the chat URL and the users to be added.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Salesforce Anywhere: Create a Salesforce Anywhere Chat (Beta)

Create a Salesforce Anywhere chat by specifying participants, and optionally, an initial message and chat title.

Note: Salesforce Anywhere Beta is a Non-GA Service and not a "Service" or part of the "Services", as defined in the Main Services Agreement ("MSA") with Salesforce. Such Non-GA Service is subject to the terms and conditions of the Universal Pilot Research Agreement ("UPRA"), including the Data Processing Addendum to the UPRA. Use of this Non-GA Service is at your sole discretion, and any purchase decisions are made only on the basis of Salesforce generally available products and features.

In Flow Builder, add an Action element to your flow. Select the Salesforce Anywhere category, and search for `chat`. Select **Create Chat**.

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the chat.

Input Parameter	Description
`chatMessage`	The first message sent to the chat. This parameter accepts single-value resources of any type. That value is treated as text.
`userEmails`	A comma-separated list of email addresses belonging to one or more users getting added to the chat. Must list at least two email addresses and no more than 50 email addresses. Email addresses must be part of your Salesforce Anywhere organization. If an email address isn't included in your Salesforce Anywhere organization, the user isn't included in the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Store Output Values

Output Parameter	Description
`chatId`	The chat's ID. This parameter accepts single-value resources of any type. That value is treated as text.
`chatTitle`	The name users see at the top of the chat. This parameter accepts single-value resources of any type. That value is treated as text.
`chatUrl`	The chat's URL. This parameter accepts single-value resources of any type. That value is treated as text.

Usage

In Flow Builder, this action doesn't check the number of email addresses or the validity of the email addresses. When either criteria is invalid, the flow fails at run time.

The API used for this action has a rate limit of 50 requests per minute and 750 requests per hour.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Salesforce Anywhere: Add a Message to a Salesforce Anywhere Chat (Beta)

Add a message to an existing Salesforce Anywhere chat by specifying the chat URL and message content.

Note: Salesforce Anywhere Beta is a Non-GA Service and not a "Service" or part of the "Services", as defined in the Main Services Agreement ("MSA") with Salesforce. Such Non-GA Service is subject to the terms and conditions of the Universal Pilot Research Agreement ("UPRA"), including the Data Processing Addendum to the UPRA. Use of this Non-GA Service is at your sole discretion, and any purchase decisions are made only on the basis of Salesforce generally available products and features.

In Flow Builder, add an Action element to your flow. Select the Salesforce Anywhere category, and search for *message*. Select **Add Message to Chat**.

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the message.

Input Parameter	Description
chatUrl	The chat's URL. This parameter accepts single-value resources of any type. That value is treated as text.
chatMessage	The message to send to the chat. This parameter accepts single-value resources of any type. That value is treated as text.
recordId	The ID of the Salesforce record to send to the chat. The record's compact layout is displayed in the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Store Output Values

Output Parameter	Description
chatId	The chat's ID. This parameter accepts single-value resources of any type. That value is treated as text.
chatMessage	The message sent to the chat. This parameter accepts single-value resources of any type. That value is treated as text.
chatUrl	The chat's URL. This parameter accepts single-value resources of any type. That value is treated as text.

Output Parameter	Description
`recordId`	The ID of the record sent to the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Usage

Only existing chat members can trigger this action. For example, only an existing chat member can successfully run a flow that sends a message to a chat about a service case when that case record is updated.

A flow can't create a record and then reference that new record ID as an input for this action.

The API used for this action has a rate limit of 50 requests per minute and 750 requests per hour.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Salesforce Anywhere: Add Users to a Salesforce Anywhere Chat (Beta)

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Add users to an existing Salesforce Anywhere chat by specifying the chat URL and the users to be added.

Note: Salesforce Anywhere Beta is a Non-GA Service and not a "Service" or part of the "Services", as defined in the Main Services Agreement ("MSA") with Salesforce. Such Non-GA Service is subject to the terms and conditions of the Universal Pilot Research Agreement ("UPRA"), including the Data Processing Addendum to the UPRA. Use of this Non-GA Service is at your sole discretion, and any purchase decisions are made only on the basis of Salesforce generally available products and features.

In Flow Builder, add an Action element to your flow. Select the Salesforce Anywhere category, and search for `users`. Select **Add Users to Chat**.

Set Input Values

Use values from earlier in the flow to set the inputs for the new users.

Input Parameter	Description
`chatUrl`	The chat's URL. This parameter accepts single-value resources of any type. That value is treated as text.
`userEmails`	Required. A comma-separated list of email addresses belonging to up to 50 users getting added to the chat. Email addresses must be part of your Salesforce Anywhere organization. If an email address isn't included in your Salesforce Anywhere organization, the user isn't be included in the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Store Output Values

Output Parameter	Description
`chatId`	The chat's ID. This parameter accepts single-value resources of any type. That value is treated as text.
`chatUrl`	The chat's URL. This parameter accepts single-value resources of any type. That value is treated as text.
`chatTitle`	The name users see at the top of the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Usage

In Flow Builder, this action doesn't check the number of email addresses or the validity of the email addresses. When either criteria is invalid, the flow fails at run time.

Only existing chat members can trigger this action. For example, only an existing chat member can successfully run a flow that adds new users to a chat about a service case when that case record is updated.

The API used for this action has a rate limit of 50 requests per minute and 750 requests per hour.

SEE ALSO:

Add and Edit Elements

Flow Core Action for Salesforce Anywhere: Send Salesforce Anywhere Alerts to Users (Beta)

Notify users about Salesforce Anywhere chat by specifying the chat URL and the users to be added.

> Note: Salesforce Anywhere Beta is a Non-GA Service and not a "Service" or part of the "Services", as defined in the Main Services Agreement ("MSA") with Salesforce. Such Non-GA Service is subject to the terms and conditions of the Universal Pilot Research Agreement ("UPRA"), including the Data Processing Addendum to the UPRA. Use of this Non-GA Service is at your sole discretion, and any purchase decisions are made only on the basis of Salesforce generally available products and features.

In Flow Builder, add an Action element to your flow. Select the Salesforce Anywhere category, and search for `alert`. Select **Send Alert**.

EDITIONS

Available in: **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Professional**, **Developer**, and **Unlimited** Editions

Set Input Values

Use values from earlier in the flow to set the inputs for the alert.

Input Parameter	Description
`alertMessage`	The message sent in the alert. This parameter accepts single-value resources of any type. That value is treated as text.

Input Parameter	Description
userEmails	A comma-separated list of the users' email addresses. This parameter accepts single-value resources of any type. That value is treated as text.
recordId	The ID of the Salesforce record to send to the chat. The record's compact layout is displayed in the chat. This parameter accepts single-value resources of any type. That value is treated as text.

Store Output Values

Output Parameter	Description
eventOperationId	The unique ID generated for the alert. This parameter accepts single-value resources of any type. That value is treated as text.

SEE ALSO:

Platform Events Developer Guide : Platform Events Considerations

Add and Edit Elements

Standard Flow Screen Components

Salesforce provides several standard screen components that extend the types of input fields available in screens.

If you need more functionality, for example, to install a custom screen component from an external library, have a developer build one for you.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Screen Input Component: Action Button

Use the Action Button component so the running user can trigger a screen action with the click of a button on a screen. The screen action runs an active autolaunched flow, and the results of the autolaunched flow can be shown on the same screen as the button. Using this component means that you need fewer screens so users can complete screen flows more quickly.

Flow Screen Input Component: Address

Simplify gathering address information by adding the Address component to a flow screen. The Address screen component displays a complete address form that's customized to your settings. It can also use state and country/territory picklists.

Flow Screen Input Component: Checkbox

Offer flow users a yes-or-no choice with a checkbox.

Flow Screen Input Component: Checkbox Group

Let users choose multiple options in a checkbox format.

Flow Screen Input Component: Choice Lookup

Let users search for and select one option from a set of choices on a flow screen. The component supports only Text values.

Flow Screen Input Component: Currency

Let users enter currency values from a flow screen.

Flow Screen Input Component: Data Table

Let users select records from a table in a flow.

Flow Screen Input Component: Date

Let users enter date values from a flow screen.

Flow Screen Input Component: Date & Time

Let users enter date and time values from a flow screen, such as to request an appointment.

Flow Screen Input Component: Dependent Picklists

Display picklists in a flow screen in which the options for one picklist depend on the selected value of another picklist. The Dependent Picklists screen component determines which options to display in each picklist by using an existing field dependency in your org. A *field dependency* connects two picklist fields on the same object.

Flow Screen Input Component: Display Image

Easily insert images in flow screens. Upload images to Salesforce as static resources and then you can reference them while configuring the component.

Flow Screen Input Component: Email

Let users enter email address values from a flow screen.

Flow Screen Input Component: Enhanced Message

Let users send a messaging component in an enhanced Messaging session.

Flow Screen Input Component: File Upload

Let users upload files from a flow screen.

Flow Screen Input Component: Long Text Area

Let users enter a paragraph or two of text from a flow screen.

Flow Screen Input Component: Lookup

Let users search for and select one or more records in a flow.

Flow Screen Input Component: Multi-Select Picklist

Let users choose multiple options in a picklist format.

Flow Screen Input Component: Name

Let users enter multiple name values with one screen component. Instead of the Name screen component, you can use Text input fields to capture name information, but it takes a lot more configuration.

Flow Screen Input Component: Number

Let users enter number values from a flow screen.

Flow Screen Input Component: Order Management Product Selector

Let users select which fields show in columns during product selector for various transaction types, such as returns or exchanges.

Flow Screen Input Component: Password

Let users enter sensitive information in a flow screen, such as a social security number. Text entered by the user is masked.

Flow Screen Input Component: Phone

Let users enter phone values from a flow screen.

Flow Screen Input Component: Picklist

Let users choose from a list of options in a picklist format.

Flow Screen Input Component: Radio Buttons

Let users choose from a list of options in a radio button format.

Flow Screen Input Component: Slack Channel Selector

Let users select a Slack channel to send a Slack message from a flow screen.

Flow Screen Input Component: Slack Workspace Selector

Let users select a Slack workspace to send a Slack message to from a flow screen.

Flow Screen Input Component: Slider

Let users visually specify number values from a flow screen.

Flow Screen Input Component: Text

Let users enter text from a flow screen, such as the name of the user's company.

Flow Screen Input Component: Toggle

Let users flip a toggle in a flow screen.

Flow Screen Input Component: URL

Let users enter URL values in a flow screen.

Flow Screen Output Component: Display Text

Display information in a flow screen.

Flow Screen Display Component: Repeater

Collect information about multiple items of the same type on a screen with the Repeater component. To use the output of the component elsewhere in the flow, loop over the output and save the relevant data in a variable. Use the variable to build a list of records.

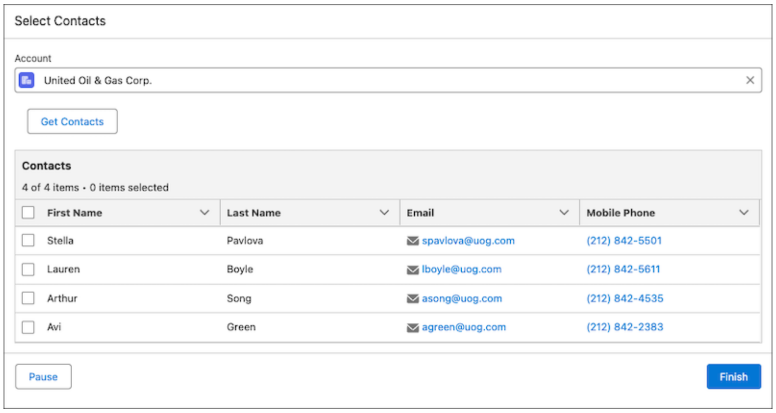
Flow Screen Output Component: Section

Organize screen components and record fields to give your users a better experience.

Flow Screen Input Component: Action Button

Use the Action Button component so the running user can trigger a screen action with the click of a button on a screen. The screen action runs an active autolaunched flow, and the results of the autolaunched flow can be shown on the same screen as the button. Using this component means that you need fewer screens so users can complete screen flows more quickly.

For example, you can make it possible for users to select an account record in a Lookup component, click a button to retrieve the contact records associated with the account record, and then display the contact records in a Data Table component on the same screen.



Select Contacts

Account

United Oil & Gas Corp. ×

Contacts

4 of 4 items • 0 items selected

□ First Name	Last Name	Email	Mobile Phone
□ Stella	Pavlova	✉ spavlova@uog.com	(212) 842-5501
□ Lauren	Boyle	✉ lboyle@uog.com	(212) 842-5611
□ Arthur	Song	✉ asong@uog.com	(212) 842-4535
□ Avi	Green	✉ agreen@uog.com	(212) 842-2383

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Action Button Name

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Label`	If you select Use Label as the table title, the user-friendly text that appears above the component.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.

Configure the Action

Attribute	Description
`Action`	The screen action that launches the autolaunched flow. This is the flow that runs when the user clicks the button rendered by the Action Button component. The autolaunched flow must be active.
`Label`	The user-friendly name for the action associated with the component. This value can be different than the label of the flow that you select as the action.
`API Name`	The API name for the action associated with the component. This value can be different than the API name of the flow that you select as the action. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Set Input Values`	Specify the value of each input field required by the action associated with the component. For example, if you select an autolaunched flow that requires an Account ID as an input, provide the Account ID. Variables that are available for input in the autolaunched flow appear in this area.
`View Output Values`	View the outputs created by the action. To reference an output elsewhere in the flow, first reference the Results field, for example, `actionButtonApiName. Results.output`. Variables that are available for output in the autolaunched flow appear in this area. Output values include: • ErrorMessage—Description of an error that occurred while executing the invocable action • IsSuccess—If true, indicates that the invocable action ran without errors • Action.Results.Flow__InterviewGuid—Unique identifier of the flow interview • Action.Results.Flow__InterviewStatus—The status of the flow interview • InProgress—If true, indicates that the screen action is running.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Specify Another Component's Behavior with the In Progress Output Attribute

When a user clicks an action button, the In Progress attribute for the associated screen action is set to `true`. When the action completes, the In Progress attribute is set back to `false`.

Use the In Progress attribute to specify another component's behavior. For example, use it to disable a screen component while the action is running. Set the value of the Disabled field on the component to the In Progress attribute. When In Progress is `true` the Disabled field is also set to `true`. When the action completes and In Progress is set to `false`, the disabled field is also set to false.

Considerations

- If a user runs a flow with an Action Button component in a web browser, the outputs of the action associated with the component are available to the browser. Don't share sensitive information as the output of an Action Button component.
- Autolaunched flows that include Wait elements or subflows with Wait elements aren't supported as Action Button actions because the flow won't resume after a Wait element.

- Action Buttons aren't supported in Repeater components.
- Launching a flow with an asynchronous path isn't supported.
- If a flow launched from the action button doesn't have fault paths, and an error occurs, a generic error message shows under the action button. To show a helpful error message to users instead, add fault paths to the launched flow. On each fault path, set an output variable to `{!$Flow.FaultMessage}`. Then, on the flow screen with the action button, add a Display Text component that's conditionally hidden and contains a helpful error message along with the fault message variable.

  Note: Even if a Display Text component content contains an error message, screen readers don't announce the content as an error message.

- If an input or output variable in the screen action's autolaunched flow is a record variable, and you change a field name on the object, the new field name isn't reflected when you refresh the inputs and outputs.
- If an input or output variable in the screen action's autolaunched flow is an Apex variable, and you change the structure of the Apex type, those changes aren't reflected when you refresh the inputs and outputs.

SEE ALSO:

Data Safety When Running Screen and Autolaunched Flows in System Context

Video: Action Button in Salesforce Flow

Flow Screen Actions

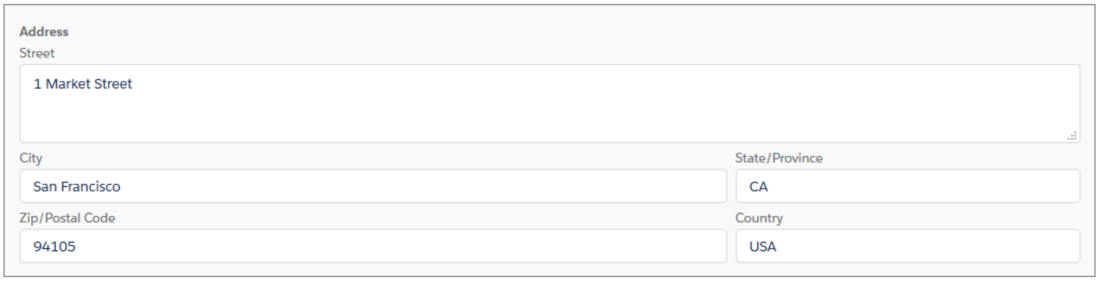
Flow Screen Input Component: Address

Simplify gathering address information by adding the Address component to a flow screen. The Address screen component displays a complete address form that's customized to your settings. It can also use state and country/territory picklists.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Address

Street: 1 Market Street

City: San Francisco

State/Province: CA

Zip/Postal Code: 94105

Country: USA

For information about adding screen components to your flow screen, see Flow Element: Screen.

Note: This screen component requires Lightning runtime.

Configure the Address Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`City Value`	To give City a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`Country Code`	The code for the country in the address. To give Country a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`Country Options`	The active countries and territories configured in state and country/territory picklists. To override the options, set this attribute to a comma-delimited set of countries and territories. This field populates a dropdown menu of options. This attribute accepts single-value resources. The value is treated as text.
`Country Value`	The value for the country in the address. To give Country a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The label for the heading that appears above the group of address fields. This attribute accepts single-value resources. The value is treated as text.
`Postal Code Value`	To give Postal Code a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.
`Show Google Maps Search Field`	Indicates whether to include a search field powered by Google Maps in the component. To include a search field, enter `true` as a boolean value. When a user selects an address in the search field, the flow populates the other fields in the component. The default value is `false`.
`Google Maps Search Field Label`	The label that appears above the Google Maps search field.

Attribute	Description
`State or Province Code`	The code for the state or province in the address. If `State/Province Options` is configured, this value is selected by default. To give State a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`State or Province Options`	The active states configured in state and country/territory picklists. To override the options, set this attribute to a comma-delimited set of states. This field populates a dropdown menu of options. This attribute accepts single-value resources. The value is treated as text.
`State or Province Value`	The value of the state or province in the address. If `State/Province Options` is configured, this value is selected by default. To give State a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.
`Street Value`	To give Street a default value, set this attribute's value. This attribute accepts single-value resources. The value is treated as text.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.

Option	Description
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Store the Address Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables. Most likely, you must store one of these attributes.

Attribute	Description
`City Value`	What the user entered in the City Value field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Country Code`	What the user entered in the Country Code field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Country Value`	What the user entered in the Country Value field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Postal Code Value`	What the user entered in the Postal Code Value field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`State or Province Code`	What the user entered in the State or Province Code field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`State or Province Value`	What the user entered in the State of Province Value field. To update records in orgs with the State and Country/Territory Picklists setting enabled, use State or Province Code instead. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Street Value`	What the user entered in the Street Value field. This value can be stored in a single-value Text variable or a Text field on a record variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.

- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- To update records in orgs with the State and Country/Territory Picklists setting enabled, use the Country Code and State or Province Code outputs instead of the Country Value and State or Province Value outputs.
- The Google Maps search fields isn't supported in Playground, Experience Builder sites, Lightning Out, Lightning Components for Visualforce, and standalone apps.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Checkbox

Offer flow users a yes-or-no choice with a checkbox.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Checkbox Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.

Attribute	Description
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Usage

When the user selects the checkbox, the screen component evaluates to `true`. If the user doesn't select the checkbox, the screen component evaluates to `false`. If the associated screen isn't executed, the screen component evaluates to `null`.

Example: Let users opt into a marketing campaign, agree to a follow-up call after a purchase, or confirm that they understand an important policy.

SEE ALSO:

Flow Resource: Global Constant

Standard Flow Screen Components

Flow Screen Input Component: Checkbox Group

Let users choose multiple options in a checkbox format.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Checkbox Group Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Choice`	Add at least one choice, record choice set, or picklist choice set to this component. Available only when you add a choice component to the screen component. If you select a dynamic Choice resource such as a collection choice set or record choice set, ensure that each value in the Choice resource is unique. Otherwise, if a user selects a duplicate value, the value is set incorrectly in Salesforce.

Attribute	Description
`Component Type`	Modify a choice component type. If the user can select only one option, these component types become available: - Picklist - Radio Buttons If the user can select multiple options, these component types become available: - Checkbox Group - Multi-select Picklist
`Data Type`	Only Text choices are supported for this component.
`Default Value`	Pre-selected choice for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Let Users Select Multiple Options`	Specifies whether the user can choose only one option or multiple options. When you select Yes for Let Users Select Multiple Options, Data Type is automatically set to Text, and non-text Choice resources are cleared from the component configuration.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Require`	Requires users to select a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition.

Option	Description
	When custom conditional logic is met Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

When a user clicks the info bubble for a Checkbox Group component, the help text appears in a separate window. For other types of Salesforce-provided components, the help text appears in a popover.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Choice Lookup

Let users search for and select one option from a set of choices on a flow screen. The component supports only Text values.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Choice Lookup Component

Attribute	Description
`Label`	User-friendly text that appears above the component.
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Require`	Requires users to select a value before they can move to the next screen.

Attribute	Description
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Placeholder Text`	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts a resource with a single value. The value is treated as text.
`Let Users Select Multiple Options`	Specifies whether the user can choose only one option or multiple options. The user can select up to 25 options.
`Choice`	Add at least one Choice resource such as a record choice set or picklist choice set to this component. Available only when you add a choice component to the screen component. If you select a dynamic Choice resource such as a collection choice set or record choice set, ensure that each value in the Choice resource is unique. Otherwise, if a user selects a duplicate value, the value is set incorrectly in Salesforce. You can't reorder choices or select the same choice twice. Choices must be compatible with the component's `Data Type` setting.

Access the Choice Lookup Component's Values in the Flow

The flow stores these attributes automatically. You can't store output values for the Choice Lookup component manually.

Attribute	Description
`selectedChoiceLabels`	If users can select only one option, the label of the choice option that the user running the flow selected. If users can select multiple options, the semi-colon separated labels of all the choice options the user running the flow selected. Reference the value later in the flow as `{!choiceLookup.selectedChoiceLabels}`.
`selectedChoiceValues`	If users can select only one option, the value of the choice option that the user running the flow selected. If users can select multiple options, the semi-colon separated values of all the choice options the user running the flow selected. Reference the value later in the flow as `{!choiceLookup.selectedChoiceValues}`.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- The Choice Lookup flow screen component isn't compatible with mobile devices or standalone Aura apps.
- The component searches for matches only in the Choice Label field of the Choice resource that you specify.
- Like other Choice fields, the Choice Lookup component supports the Was Selected operator.
- The search is case-sensitive.
- Initially, 20 choice options display. As you scroll, more choice options load in groups of 100, up to the maximum of 1,020.
- If you apply a filter after loading your initial choices, the display resets, showing the new 20 choices.

- The Choice Lookup component doesn't support the Display text input field for Choice resources. For example, if you select the Display text input checkbox when you configure a Choice resource and add the resource to the Choice Lookup component, the component doesn't display a text input field when the user selects the corresponding choice at run time.

SEE ALSO:

Choose a Lookup Option for a Flow Screen

Flow Screen Input Component: Currency

Let users enter currency values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Currency Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.

Option	Description
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Data Table

Let users select records from a table in a flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Data Table Name

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Label`	If you select Use Label as the table title, the user-friendly text that appears above the component.
`Use Label as the table title`	Indicates whether to display the Label value above the table when you run the flow.

Configure the Data Table Source

Attribute	Description
`Source Collection`	A collection of records to use to populate the table.
`Show search bar`	Enables users to search and filter their record results.

Configure the Data Table Rows

Attribute	Description
`Row Selection Mode`	Indicates how many rows the user can select in the table. You can set the value to: **Multiple** The user can select any number of rows between the Minimum Row Selection and Maximum Row Selection values.

Attribute	Description
	Single The user can select up to one row. **View only** The user can't select any rows.
`Minimum Row Selection`	Specifies the minimum number of rows that the user must select.
`Maximum Row Selection`	Specifies the maximum number of rows that the user can select.
`Default Selection`	Collection that specifies which records to preselect in the table.
`Require user to make a selection`	Specifies whether the user must select a row before navigating to the next screen.

Configure the Data Table Columns

To add the first column to the table, configure these fields. To add subsequent columns, click **Add column**. Drag and drop the columns to reorder them.

Attribute	Description
`Source Field`	Field from the Source Collection object to display in the column. Fields with the anyType data type such as the NewValue field of the AccountHistory object aren't supported.
`Custom column label`	Indicates whether to display the column Label value you specify as the column header.
`Label`	If Custom column label is selected, the text to display as the column header. The text is also read by screen readers.
`Default Text Overflow Mode`	Specifies how text that is longer than the width of the column appears. You can set the value to: **Wrap Text** The screen displays the text on multiple lines. **Clip Text** The screen truncates the text to fit.

Note: If you're using a field that has a namespace, add the namespace to the beginning of the source field. For example, if your field's namespace is Acme, enter *`Acme__FieldName__c`*.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Store the Data Table Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but most likely you must store these attributes.

Attribute	Description
`First Selected Row`	First record in the table selected by the flow user. If a user selects two records, this record is the first selected record from top to bottom.
`Selected Rows`	The list of records that the user selects. The records are ordered according to their position in the table from top to bottom.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- The Data Table flow screen component isn't compatible with mobile devices.
- If you use the Get Records flow element to retrieve the records to display in the Data Table, select Choose fields and let Salesforce do the rest for the best performance.
- The maximum height of a Data Table is 400 pixels.
- If you choose to wrap the text in a Data Table, ensure that the text doesn't overflow when you test your flow. Wrapped text can overflow when a Data Table is compressed on a screen, for example, when it's in one of multiple columns.
- A Data Table can display up to 1,500 records. However, your search is performed on the entire dataset.
- You can select up to 200 records in a Data Table.
- If you apply a filter after loading your initial records, only the new results are shown. The initial records are no longer included in the display.
- If a Data Table includes a formula field and records or updates to records that haven't been committed to the database, the table doesn't evaluate the formula properly.

  For records that don't exist in the database, update the value of the formula field with an assignment using a static value or Formula resource. Doing so doesn't affect any subsequent Create or Update operations in the flow.

  For existing records that have been updated, use an invocable action to reevaluate the formula, or use the IN operator to refresh the records and formula field values.

- If you include a lookup or master-detail relationship field in a Data Table, the table doesn't display the field value. For example, a Data Table can't display the Name field of a related record. To display field values from related records, use object formula fields. You can also use object formula fields to link to related record fields, for example:

  ```
  HYPERLINK( "/" & CASESAFEID(Id), Related_Record__r.Name, "_self" )
  ```

- You can't search the Time field.
- In multi-currency orgs, the Data Table component doesn't support records that are in a different currency from the user's personal currency.
- To display multilingual column header labels in the Data Table component, use the `$Label` global variable to specify custom labels. For more information about creating and translating custom labels, see Custom Labels.
- Data Table selections at runtime are subject to the client payload data limit described in Lightning Aura Components Developer Guide. If you exceed this limit, the flow returns a generic error message. For example, if you include file data that exceeds the limit, the flow generates an error. We recommend avoiding fields like the VersionData field of ContentVersion records in your source collection.
- If you rename a field in Object Manager that's mapped to a column in a Data Table, Salesforce doesn't update the column name. To see the new name in the Data Table, remove the column and then add it again.
- If you have a flow open that has a Data Table component, and you update your user settings time zone on another page, refresh the flow page to show the updated date and time fields in the Data Table component.
- When you set the row selection, be careful if you want to use the row selection of another Data Table component. Salesforce doesn't support the use of row selections that have duplicate record variables without record IDs.
- If you set the row-selection mode to single and make it required, or if you set the minimum and maximum row selection to 1, Salesforce uses a radio button at run time. Otherwise, we use checkboxes at run time.
- If you package a flow that has a Data Table component, the fields used in the Data Table aren't automatically added to the package. If you use a field in the Data Table component, you must manually add it to the package.
- If you delete a custom field that a Data Table component uses, you must also remove the field from the screen flow where the Data Table component is used.
- If you use a Data Table component that uses a custom object or custom field in an org without a namespace, and then later add a namespace to the org, you must also add that namespace to the associated column fields in the Data Table.

SEE ALSO:

Use Multilingual Labels in Data Table Column Headers

Data Safety When Running Screen and Autolaunched Flows in System Context

Flow Screen Input Component: Date

Let users enter date values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Data Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.

Attribute	Description
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.

Option	Description
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Date & Time

Let users enter date and time values from a flow screen, such as to request an appointment.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Date & Time Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.

Attribute	Description
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value.

Option	Description
	If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Dependent Picklists

Display picklists in a flow screen in which the options for one picklist depend on the selected value of another picklist. The Dependent Picklists screen component determines which options to display in each picklist by using an existing field dependency in your org. A *field dependency* connects two picklist fields on the same object.

Note: This screen component requires Lightning runtime.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Dependent Picklists Component

Tip: Before you add a Dependent Picklists screen component to your flow, define field dependencies for the appropriate picklist fields in your org.

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component.

Attribute	Description
	An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Object API Name`	The API name of the object. The picklist fields that you identify in Picklist 1 API Name, Picklist 2 API Name, and Picklist 3 API Name must be associated with this object. This attribute accepts single-value resources. The value is treated as text.
`Picklist 1 API Name`	The API name of the first picklist field. For the specified object, this picklist field must be the controlling field in a field dependency between Picklist 1 and Picklist 2. This attribute accepts single-value resources. The value is treated as text.
`Picklist 1 Label`	The label for the first picklist field. This attribute accepts single-value resources. The value is treated as text.
`Picklist 1 Required`	If set to `$GlobalConstant.True`, the running user must enter a value. This attribute accepts single-value Boolean resources.
`Picklist 1 Value`	The default selection for the first picklist field. Configuring this attribute pre-selects an option for the field. This attribute accepts single-value resources. The value is treated as text.
`Picklist 2 API Name`	The API name of the second picklist field. For the specified object, this picklist field must be the dependent field in a field dependency between Picklist 1 and Picklist 2. If you display a third picklist field, Picklist 2 must be the controlling field in a field dependency between Picklist 2 and Picklist 3. This attribute accepts single-value resources. The value is treated as text.
`Picklist 2 Label`	The label for the second picklist field. This attribute accepts single-value resources. That value is treated as text.
`Picklist 2 Required`	If set to `$GlobalConstant.True`, the running user must enter a value. This attribute accepts single-value Boolean resources.
`Picklist 2 Value`	The default selection for the second picklist field. Configuring this attribute pre-selects an option for the field. This attribute accepts single-value resources. The value is treated as text.
`Picklist 3 API Name`	The API name of the third picklist field. For the specified object, this picklist field must be the dependent field in a field dependency between Picklist 2 and Picklist 3. This attribute accepts single-value resources. That value is treated as text.

Attribute	Description
`Picklist 3 Label`	The label for the third picklist field. This attribute accepts single-value resources. The value is treated as text.
`Picklist 3 Required`	If set to `$GlobalConstant.True`, the running user must enter a value. This attribute accepts single-value Boolean resources.
`Picklist 3 Value`	The default selection for the third picklist field. Configuring this attribute pre-selects an option for the field. This attribute accepts single-value resources. The value is treated as text.

Note: If your org has a namespace, add the namespace to the beginning of the object's API name, and each picklist API Name. For example, if you have a custom object called Insurance_Agent__c, and your org's namespace is Acme, enter *`Acme__Insurance_Agent__c`*.

Store the Dependent Picklists Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables. Most likely, you must store one of these attributes.

Attribute	Description
`Picklist 1 Value`	What the user selected for the first picklist field. You can store this value in a single-value Text variable or a Text field on a record variable.
`Picklist 2 Value`	What the user selected for the second picklist field. You can store this value in a single-value Text variable or a Text field on a record variable.
`Picklist 3 Value`	What the user selected for the third picklist field. You can store this value in a single-value Text variable or a Text field on a record variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

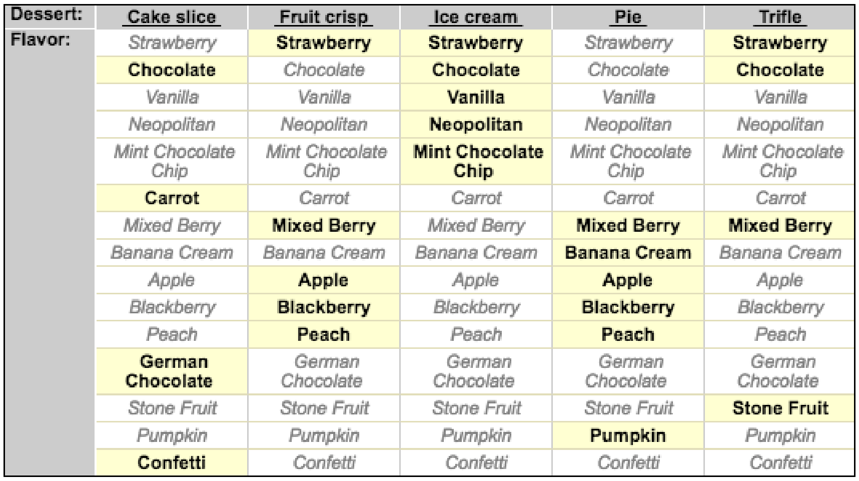
Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.

Option	Description
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Example: For example, in a Dinner Order flow, users select a specific dessert. Each dessert comes in different flavors, and the flavor options change based on the dessert that the user selects.

- On the Guest Order custom object, define two picklist fields: Dessert and Flavor.
- Define a field dependency between Dessert and Flavor, where Dessert is the controlling picklist. Identify which Flavor options apply to each Dessert option.

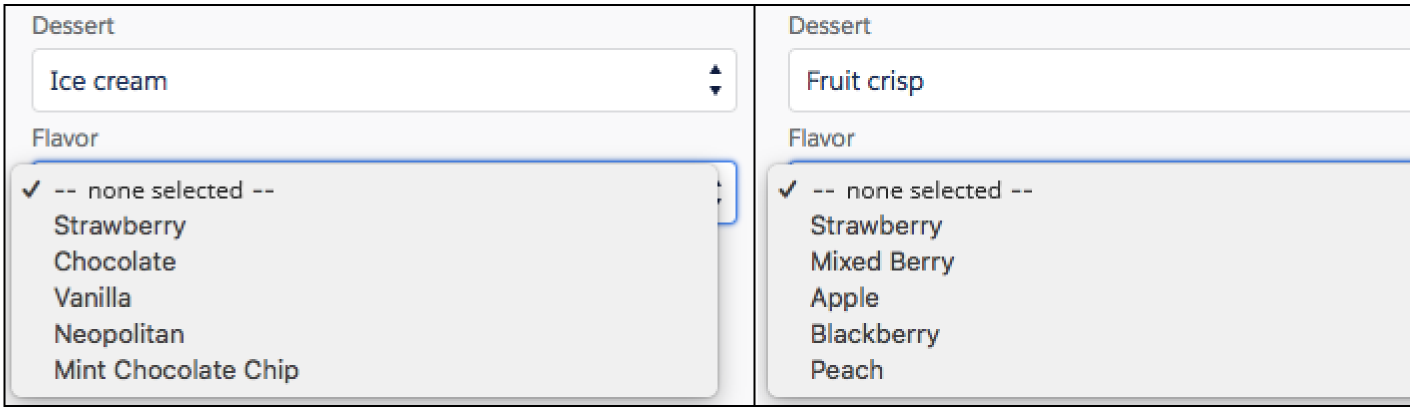


Dessert:	Cake slice	Fruit crisp	Ice cream	Pie	Trifle
Flavor:	Strawberry	**Strawberry**	**Strawberry**	Strawberry	**Strawberry**
	Chocolate	Chocolate	**Chocolate**	Chocolate	**Chocolate**
	Vanilla	Vanilla	**Vanilla**	Vanilla	Vanilla
	Neopolitan	Neopolitan	**Neopolitan**	Neopolitan	Neopolitan
	Mint Chocolate Chip	Mint Chocolate Chip	**Mint Chocolate Chip**	Mint Chocolate Chip	Mint Chocolate Chip
	Carrot	Carrot	Carrot	Carrot	Carrot
	Mixed Berry	**Mixed Berry**	Mixed Berry	**Mixed Berry**	**Mixed Berry**
	Banana Cream	Banana Cream	Banana Cream	**Banana Cream**	Banana Cream
	Apple	**Apple**	Apple	**Apple**	Apple
	Blackberry	**Blackberry**	Blackberry	**Blackberry**	Blackberry
	Peach	**Peach**	Peach	**Peach**	Peach
	German Chocolate	German Chocolate	German Chocolate	German Chocolate	German Chocolate
	Stone Fruit	Stone Fruit	Stone Fruit	Stone Fruit	**Stone Fruit**
	Pumpkin	Pumpkin	Pumpkin	**Pumpkin**	Pumpkin
	Confetti	Confetti	Confetti	Confetti	Confetti

- In your flow screen, add a Dependent Picklists screen component. Configure the component with these values.

Attribute	Value
`Object API Name`	Guest_Order__c
`Picklist 1 API Name`	Dessert__c
`Picklist 1 Label`	Dessert
`Picklist 2 Value`	Flavor__c
`Picklist 2 Label`	Flavor

When a user runs the flow, the options for Flavor change based on what’s selected for Dessert.



Dessert: Ice cream

Flavor

Dessert: Fruit crisp

Flavor

Considerations

Screen input component values are set to null when they're hidden by conditional visibility. But hidden picklists in a Dependent Picklists component aren't set to null unless the entire Dependent Picklists component is hidden.

SEE ALSO:

- Standard Flow Screen Components
- Define Dependent Picklists

Flow Screen Input Component: Display Image

Easily insert images in flow screens. Upload images to Salesforce as static resources and then you can reference them while configuring the component.

For information about adding screen components to your flow screen, see Flow Element: Screen.

Note: This screen component requires Lightning runtime.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Display Image Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Horizontal Alignment`	If you don't want the browser to determine the image's horizontal alignment, enter a specific alignment value. Valid values are: left, center, or right. This attribute accepts single-value resources. The value is treated as text.

Attribute	Description
`Image Alt Text`	Alternative text for screen readers and other assistive technology and for browsers that can't load the image. Provide a meaningful description unless the image is purely decorative or redundant. To have assistive technology skip the image, set `Image Alt Text` to `{!$GlobalConstant.EmptyString}`. If you don't set this attribute, assistive technology reads the file path from the image source (`img src`), which can confuse your users and potentially create an accessibility compliance issue. This attribute accepts single-value resources. The value is treated as text.
`Image CSS`	Override the CSS for your image by providing your own CSS string. Example: `border-radius: 8px; box-shadow: 10px 5px 5px blue; opacity: 0.75;` This attribute accepts single-value resources. The value is treated as text.
`Image Height`	If you don't want the browser to determine the image height, enter a specific height value. Valid values are a number and unit, or a percentage of the container. Examples: 200 px, 2 cm, 50%. If you enter a number value and don't enter a unit value, the unit value defaults to pixels. This attribute accepts single-value resources. The value is treated as text.
`Image Name`	Required. The name of a static resource that contains an image file. The image must be a `.png` or `.jpg` file. This attribute accepts single-value resources. The value is treated as text.
`Image Width`	If you don't want the browser to determine the image width, enter a specific width value. Valid values are a number and unit, or a percentage of the container. Examples: 200 px, 2 cm, 50%. If you enter a number value and don't enter a unit value, the unit value defaults to pixels. This attribute accepts single-value resources. The value is treated as text.

Store the Display Image Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.

Option	Description
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

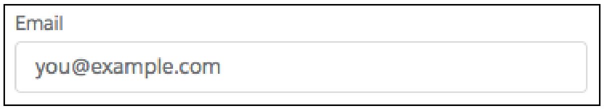
Flow Screen Input Component: Email

Let users enter email address values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Note: This screen component requires Lightning runtime.

Configure the Email Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.

Attribute	Description
`Label`	The label that appears above the email field. This attribute accepts single-value resources. The value is treated as text.
`Placeholder Text`	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts a resource with a single value. The value is treated as text.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.
`Value`	The value of the email field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.

Store the Email Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but Value is the most likely attribute you must store.

To store the email address that the user entered, store the Value attribute in a flow variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to:

Option	Description
	Always Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.

Option	Description
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Enhanced Message

Let users send a messaging component in an enhanced Messaging session.

EDITIONS

Messaging is available in: Lightning Experience with the Digital Engagement add-on SKU

Messaging is available in: **Enterprise**, **Unlimited**, and **Developer** Editions with Service Cloud or Sales Cloud

Configure the Enhanced Message Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Messaging Session ID`	The variable containing the record ID of the messaging session.
`Messaging Component Name`	The messaging component to incorporate into the screen-based flow—for example, a time selector component. Create messaging components on the Messaging Components page in Setup.
`Time Slot Options`	The list of time slot options to show to the customer, generated by a custom Apex action. This field is visible only if a time selector messaging component is selected.
`Object Type`	The type of record to show to the customer—for example, Case. This field is visible only if a question with dynamic options messaging component is selected.
`Record Variable`	The record variable that determines which records of the selected object type to show to the customer. This field is visible only if a question with dynamic options messaging component is selected.
`Parameter Name`	The name of a custom parameter on the selected messaging component.
`Parameter Value Type`	The type of custom parameter. Valid values are Variable and Literal. The value defaults to Variable.
`Value`	The value to pass into the parameter—for example, a question to precede a list of options.

Attribute	Description
`Variable`	The variable to pass into the parameter—for example, a variable containing the contact's name.

SEE ALSO:

Standard Flow Screen Components

Salesforce Help: Send Structured Content with Messaging Components

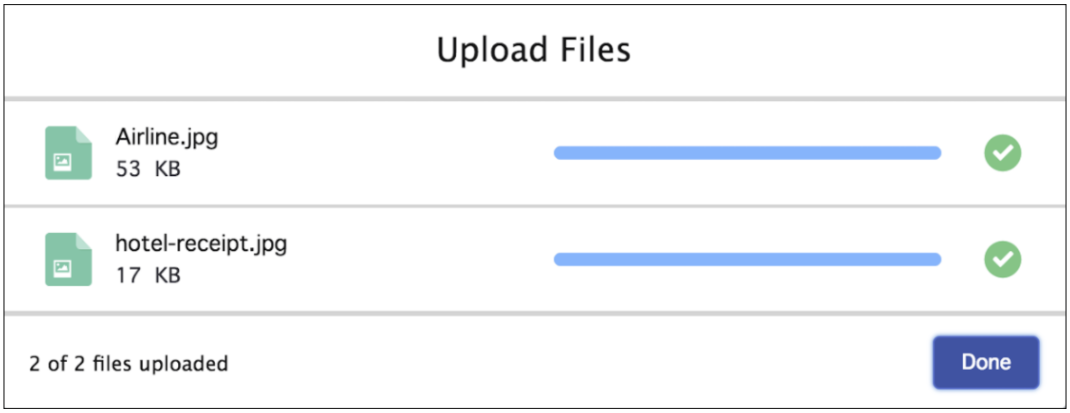
Flow Screen Input Component: File Upload

Let users upload files from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Upload Files

Airline.jpg
53 KB

hotel-receipt.jpg
17 KB

2 of 2 files uploaded

Note: This screen component requires Lightning runtime.

Configure the File Upload Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`Accepted Formats`	Using the format *`.ext`*, enter a comma-separated list of the file extensions that the user can upload. This attribute accepts single-value resources. The value is treated as text.

Attribute	Description
`Allow Multiple Files`	If set to *`$GlobalConstant.True`*, the user can upload multiple files. This attribute accepts single-value Boolean resources.
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`File Upload Label`	Required. Label that appears above the upload button. This attribute accepts single-value resources. The value is treated as text.
`Hover Text`	Tooltip that appears when the user hovers over the component. This attribute accepts single-value resources. The value is treated as text.
`Related Record ID`	Required. ID of the record to associate the files with. If no value is passed, the component is disabled. This attribute accepts single-value resources. The value is treated as text.

Note: Custom fields added to the ContentVersion object page are rendered in Experience Cloud sites through the contentVersionEditWizard. The contentVersionEditWizard is supported on desktop, but not mobile. Since there's no screen in mobile to edit or add details to custom fields, file uploads fail when custom fields are marked as required.

Store the File Upload Component's Values in the Flow

All attributes are available to store in flow variables, but usually you must store one of these attributes. The values are assigned to the flow variables when the user navigates to the next screen.

Attribute	Description
`Content Document IDs`	The IDs of the uploaded files. You can store this value in a Text collection variable.
`Uploaded File Names`	The names of the uploaded files. You can store this value in a Text collection variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

File Upload Limits

By default, you can upload up to 10 files simultaneously, unless Salesforce changed that limit. The org limit for the number of files simultaneously uploaded is 25 files with a minimum of one file. The maximum file size you can upload is 2 GB. In Experience Cloud sites, the file size limits and types allowed follow the settings determined by site file moderation. By default, guest user files are blocked from being uploaded. Admins can change the settings to let guest users upload files. From **Setup** > **, select** > **General Settings**, and then select **Allow site guest users to upload files**. This setting is only valid if the Secure guest user record access setting is enabled in the org.

Note: The file upload component isn't supported on mobile app or browser when used with flows that are accessed through URLs. This restriction doesn't apply when the file upload component is used in Lightning App Builder or Experience Builder.

Lightning Out doesn't support the File Upload component.

Considerations

If a user doesn't upload any files, the value of the `Content Document IDs` and `Uploaded File Names` outputs is an empty collection, represented as "`[]`". If you check the ISBLANK or ISNULL operator, the value is always `false`.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Long Text Area

Let users enter a paragraph or two of text from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Long Text Area Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.

Attribute	Description
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Lookup

Let users search for and select one or more records in a flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Lookup Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Field API Name`	The API name of a lookup field on the source object referenced in Object API Name. The lookup field referenced in Field API Name must be a field on the object referenced in Object API Name. For example, if you want to add a lookup for an account, find an object that has an account lookup field on it. In this case, let's use the account lookup field on the Contact object. The API name of the account lookup field on the Contact object is AccountId, so enter *`AccountId`* for Field API Name, then enter *`Contact`* for Object API Name.
`Label`	The text that shows at the top of the component that tells the running user how to use the screen component. For example, if you're adding an account lookup, the label could be Select Account.

Attribute	Description
`Object API Name`	The API name of the source object that has the lookup field referenced in Field API Name. The source object can be any object that has the type of lookup field that you want to use. The lookup field referenced in Field API Name must be a field on the object referenced in Object AI Name. To use the Lookup component, the running user of the flow must have the Create permission on the source object. For example, if you want to add a lookup for a contact, find an object that has a contact lookup field on it. In this case, let's use the contact lookup field on the Case object. The API name of the Case object is Case, so enter *Case* for Object API Name, then enter *ContactId* for Field API Name.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Maximum Selections`	The maximum number of records that the user can select. The default value is 1.
`Record Id`	Initially, if Maximum Selections is *1* or Maximum Selections is greater than 1 and the Record ID Collection field is *null*, the record ID selected by default for the lookup. When a user runs the flow, the value changes to the flow user's selection.
`Record Id Collection`	Initially, if Maximum Selections is greater than 1, the default record IDs for the lookup. If Maximum Selections is greater than 1 and the Record ID field is *null*, the first value is the record IDs selected by default for the lookup. You can specify any number of record IDs up to the Maximum Selections value. When a user runs the flow, the value changes to the flow user's selections.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.

Note: If your org has a namespace, add the namespace to the beginning of the object's API name, and field's API Name. For example, if you have a custom object called Insurance_Agent__c, and your org's namespace is Acme, enter *Acme__Insurance_Agent__c*.

Store the Lookup Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but most likely you must store these attributes.

Attribute	Description
`Record ID`	If the Maximum Selections value is 1, the ID of the record that the user selects. You can store this value in a Text variable.
`Record ID Collection`	If the Maximum Selections value is greater than 1, the list of IDs of the records that the user selects. If the Maximum Selections value is 1 and Record ID is null, the first value in the collection is the ID of the record that the user selects. You can store this value in a Text collection variable.
`Record Name`	If the Maximum Selections value is 1, the value of the Name field of the record that the user selects. If the Maximum Selections value is greater than 1, the value of the Name field of the first record that the user selects. You can store this value in a Text variable. This value isn't populated when the Name field of the record is an external object.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- The Lookup flow screen component isn't compatible with mobile devices or standalone Aura apps.
- Dependent lookup filters aren't enforced for the Lookup component in a flow. Other lookup filters are enforced the same as they are in Lightning Experience record pages. When the flow accesses the Salesforce database, lookup filters are enforced. For example, when the flow executes the Create Records element, the flow fails if the value of the lookup field doesn't meet the lookup filter requirements.
- To filter records based on resources and information from the flow, consider using a Choice Lookup component.
- A custom lookup field to a user record isn't supported.

  Tip: To let a flow user choose from a list of user records, employ a standard User lookup field like `CreatedById` or `LastModifiedById`. `OwnerId` isn't supported.

- At run time, when the flow user types two characters in the field, it shows up to five recent records whose `Name` field matches the query.
- Dependent lookup filters aren't supported.

- During run time, if the lookup field defined in `Field API Name` isn't on an assigned page layout, the lookup component displays `Search undefined...`. To display the correct text, add the defined lookup field to all of the source object's page layouts that are assigned to running users.
- Invalid Record IDs are ignored. A Record ID is invalid if it isn't a valid Salesforce Record ID or its key prefix doesn't match with the field API name object.
- If the Maximum Selections value is 1 and the Record ID Collection and Record ID are both changed, the Record ID takes precedence. The Record ID Collection is ignored.
- If the Maximum Selections value is greater than 1, the Record ID Collection takes precedence when Record ID is populated. But, if Record ID Collection isn't populated, the Record ID is used to populate Record ID Collection as a single it
- Relationship fields that are related to more than one object, also known as polymorphic fields, aren't supported. For example, because a task record's WhoId field can be related to a contact or a lead, it isn't supported for this component.
- `Field API Name` and `Object API Name` are case-sensitive.
- The Lookup flow screen component doesn't support filtering by the source object record type.

SEE ALSO:

Standard Flow Screen Components

Considerations for Lookup Filters

The Enhanced Page Layout Editor

Flow Screen Input Component: Multi-Select Picklist

Let users choose multiple options in a picklist format.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Multi-Select Picklist Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Choice`	Add at least one choice, record choice set, or picklist choice set to this component. Available only when you add a choice component to the screen component.
`Component Type`	Modify a choice component type. If the user can select only one option, these component types become available: • Picklist • Radio Buttons If the user can select multiple options, these component types become available: • Checkbox Group

Attribute	Description
	- Multi-select Picklist
`Data Type`	Only Text choices are supported for this component.
`Default Value`	Pre-selected choice for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Let Users Select Multiple Options`	Specifies whether the user can choose only one option or multiple options. When you select Yes for Let Users Select Multiple Options, Data Type is automatically set to Text, and non-text Choice resources are cleared from the component configuration.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Require`	Requires users to select a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- Rich text isn't supported in the Multi-Select Picklist component.

SEE ALSO:

Standard Flow Screen Components

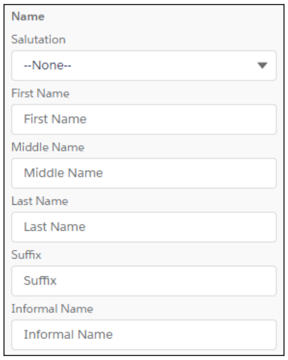
Flow Screen Input Component: Name

Let users enter multiple name values with one screen component. Instead of the Name screen component, you can use Text input fields to capture name information, but it takes a lot more configuration.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Name

Salutation

--None--

First Name

First Name

Middle Name

Middle Name

Last Name

Last Name

Suffix

Suffix

Informal Name

Informal Name

Note: This screen component requires Lightning runtime.

Configure the Name Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Fields to Display`	By default, the component displays only the First Name and Last Name fields, but other fields are available. To customize which fields to display at run time, set this attribute to a comma-separated list of the field names. • For First Name, use firstName • For Last Name, use lastName • For Middle Name, use middleName • For Informal Name, use informalName • For Salutation, use salutation • For Suffix, use suffix This attribute doesn't control the order that the fields display in. For example, to display all the fields, set this attribute to *`firstName, lastName, middleName, informalName, salutation, suffix`*. This attribute accepts single-value resources. The value is treated as text.
`First Name`	The value of the First Name field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.
`Informal Name`	The value of the Informal Name field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.
`Label`	The label that appears above the name fields. This attribute accepts single-value resources. The value is treated as text.
`Last Name`	The value of the Last Name field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.

Attribute	Description
`Middle Name`	The value of the Middle Name field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Salutation`	The value of the Salutation field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.
`Salutation Options`	By default, the options for Salutation are Mr., Mrs., and Ms. To override these options, set this attribute to a comma-separated list of values. This attribute accepts single-value resources. The value is treated as text.
`Suffix`	The value of the Suffix field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.

Store the Name Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables. Most likely, you must store one of these attributes.

Attribute	Description
`First Name`	What the user entered in the First Name field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Informal Name`	What the user entered in the Informal Name field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Last Name`	What the user entered in the Last Name field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Middle Name`	What the user entered in the Middle Name field. This value can be stored in a single-value Text variable or a Text field on a record variable.
`Salutation`	What the user entered in the Salutation field. This value can be stored in a single-value Text variable or a Text field on a record variable.

Attribute	Description
`Suffix`	What the user entered in the Suffix field. This value can be stored in a single-value Text variable or a Text field on a record variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component.

Option	Description
	If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Number

Let users enter number values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Number Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value.

Attribute	Description
	Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component.

Option	Description
	If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Order Management Product Selector

Let users select which fields show in columns during product selector for various transaction types, such as returns or exchanges.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Order Management Product Selector Component

Note: This screen component requires Lightning runtime.

Set the product fields by using data in the flow.

Attribute	Description
Configure Columns	Required. Select up to ten columns to display.
Order Product Summaries	Required. A collection of product summaries.
Selected Order Product Summaries	Required. The subset collection of product summaries being changed.
Selected Order Summary	Required. The order summary that the product summaries belong to.

Attribute	Description
Transaction Type	Optional. The type of transaction. Valid values are Cancel, RMS, Return, Reship, Discount, and Exchange.

Attributes to Output

Attribute	Description
Order Product Summaries	A collection of product summaries.
Selected Order Summary	The selected order summary.
Selected Order Product Summaries	The subset collection of product summaries being changed.
Transaction Type	The type of transaction.

Flow Screen Input Component: Password

Let users enter sensitive information in a flow screen, such as a social security number. Text entered by the user is masked.

Note: This screen component doesn't encrypt the value entered by the user. When the flow references a Password screen component, such as in an Assignment element or a Display Text screen component, the value isn't masked.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Password Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.

Attribute	Description
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

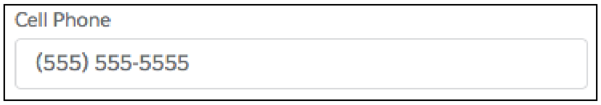
Flow Screen Input Component: Phone

Let users enter phone values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Cell Phone

(555) 555-5555

Note: This screen component requires Lightning runtime.

Configure the Phone Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component.

Attribute	Description
	An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Label`	The label that appears above the phone field. This attribute accepts single-value resources. The value is treated as text.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Pattern`	Determines whether the value is valid. By default, there's no pattern. This attribute accepts single-value resources. The value is treated as text.
`Placeholder Text`	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts a resource with a single value. The value is treated as text.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.
`Value`	The value of the phone field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.

Store the Phone Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but Value is the most likely attribute you must store.

To store the phone number that the user entered, map the Value attribute to a flow variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.

Option	Description
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Picklist

Let users choose from a list of options in a picklist format.

Starting with Flow Run-time API version 52, the first option listed for all picklists is --None--. If you don't set a default value for a picklist in Flow Builder, the --None-- option is automatically selected at run time. --None-- is treated as a null value. If you set the picklist as required and the user selects --None--, then the flow run time prevents the user from proceeding to the next screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Picklist Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Choice`	Add at least one choice, record choice set, or picklist choice set to this component. Available only when you add a choice component to the screen component. If you select a dynamic Choice resource such as a collection choice set or record choice set, ensure that each value in the Choice resource is unique. Otherwise, if a user selects a duplicate value, the value is set incorrectly in Salesforce.
`Component Type`	Modify a choice component type. If the user can select only one option, these component types become available: • Picklist • Radio Buttons If the user can select multiple options, these component types become available: • Checkbox Group • Multi-select Picklist

Attribute	Description
`Data Type`	Controls which choices are available for this component. For example, if you choose Number, you can't select a Text choice.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs. Available only when the data type is Number or Currency.
`Default Value`	Pre-selected choice for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Let Users Select Multiple Options`	Specifies whether the user can choose only one option or multiple options. When you select Yes for Let Users Select Multiple Options, Data Type is automatically set to Text, and non-text Choice resources are cleared from the component configuration.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Require`	Requires users to select a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Considerations

- Rich text isn't supported in the Picklist component.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Radio Buttons

Let users choose from a list of options in a radio button format.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Radio Buttons Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Choice`	Add at least one choice, record choice set, or picklist choice set to this component. Available only when you add a choice component to the screen component. If you select a dynamic Choice resource such as a collection choice set or record choice set, ensure that each value in the Choice resource is unique. Otherwise, if a user selects a duplicate value, the value is set incorrectly in Salesforce. All multi-select choice components use a text data type, but radio buttons and picklists can also use numbers or Boolean choices.
`Component Type`	Modify a choice component type.

Attribute	Description
	If the user can select only one option, these component types become available: - Picklist - Radio Buttons If the user can select multiple options, these component types become available: - Checkbox Group - Multi-select Picklist
`Data Type`	Controls which choices are available for this component. For example, if you choose Number, you can't select a Text choice.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your flow runs. Available only when the data type is Number or Currency.
`Default Value`	Pre-selected choice for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Let Users Select Multiple Options`	Specifies whether the user can choose only one option or multiple options. When you select Yes for Let Users Select Multiple Options, Data Type is automatically set to Text, and non-text Choice resources are cleared from the component configuration.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Require`	Requires users to select a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition.

Option	Description
	When any condition is met (OR) Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Slack Channel Selector

Let users select a Slack channel to send a Slack message from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Slack Channel Selector Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Note: This screen component requires Lightning runtime.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.

Attribute	Description
`Slack app id`	The ID of the Slack app connected to Salesforce. This attribute accepts Text variables. Only the Slack app owner can get the app ID. From https://api.slack.com, go to your apps, then Basic Information, and find the app's ID.
`Slack workspace id`	The ID of the Slack workspace where the Slack app is installed. This attribute accepts Text variables. To get the ID, open the web version of Slack and copy the alphanumeric section of the Slack URL starting with T.
`Use Bot Token`	Fetches a list of Slack channels based on the Slack app's bot token. This attribute accepts Boolean resources. If set to `$GlobalConstant.False`, the Slack app uses the user token instead of the bot token.
`Use Channel Search API`	Indicates whether to use type-ahead Slack channel search to fetch a list of Slack channels. This attribute accepts Boolean resources. Requires that the Slack app be registered with Slack to use the private API.
`Label for dropdown`	Text that appears in the selector heading. Use text to give users a hint of what the Slack channel selector is for. This attribute accepts single-value resources. The value is treated as text.
`Placeholder for dropdown`	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts single-value resources. The value is treated as text.
`Required`	If set to `$GlobalConstant.True`, the running user must enter a value. This attribute accepts single-value Boolean resources.
`Selected channel id`	The ID of the selected Slack channel. To get the channel ID, right-click the channel and select **View channel details**. The Channel ID is on the About tab.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component.

Option	Description
	When all conditions are met (AND) Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

Flow Screen Input Component: Slack Workspace Selector

Let users select a Slack workspace to send a Slack message to from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Slack Workspace Selector Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Note: This screen component requires Lightning runtime.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Slack appID`	The ID of the Slack app connected to Salesforce. This attribute accepts Text variables. Only the Slack app owner can get the app ID. From https://api.slack.com, go to your apps, then Basic Information, and find the app's ID.
`Workspace ID`	The ID of the Slack workspace where the Slack app is installed. This attribute accepts Text variables. To get the ID, open the web version of Slack and copy the alphanumeric section of the Slack URL starting with T.
`Select...`	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts single-value resources. The value is treated as text.
`Workspace Name`	The name of the Slack workspace where the Slack app is installed. This attribute accepts single-value resources. The value is treated as text.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic.

Option	Description
	You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

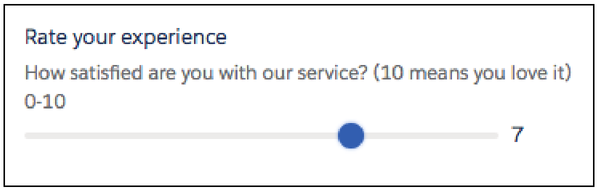
Flow Screen Input Component: Slider

Let users visually specify number values from a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions


Rate your experience

How satisfied are you with our service? (10 means you love it)

0-10

Note: This screen component requires Lightning runtime.

Configure the Slider Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Label`	This label appears above the slider. This attribute accepts single-value resources. The value is treated as text.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Range Maximum`	The maximum value of the slider range. The default is 100. This parameter accepts single-value Number resources.
`Range Minimum`	The minimum value of the slider range. The default is 0. This parameter accepts Number resources.
`Slider Size`	Controls the size of the slider. The accepted values are x-small, small, medium, or large. This parameter accepts single-value resources of any type. That value is treated as text.

Attribute	Description
`Step Size`	Divides the slider into a set of steps. The default is 1. For example, for a range of 0–100, set the Step Size to 10 to let the user select every 10th value. Other example step sizes are 0.1 and 5. This parameter accepts single-value Number resources.
`Value`	The default value represented by the slider position. Setting this attribute from the Inputs tab pre-sets the value. This parameter accepts single-value Number resources.

Store the Slider Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but Value is the most likely attribute you must store.

To store the value that the user selected, map the Value attribute to a Number flow variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition.

Option	Description
	When custom conditional logic is met Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Input Component: Text

Let users enter text from a flow screen, such as the name of the user's company.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Text Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Default Value`	Pre-populated value for the component. If the associated screen isn't executed or the conditions for component visibility aren't met, the stored value of the component is `null`.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The text that appears with the screen component that tells the running user how to use it.
`Provide Help`	Give your users more context with this screen component. The text you enter is available in an info bubble next to the component.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Require`	Requires users to enter a value before they can move to the next screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition.

Option	Description
	When any condition is met (OR) Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

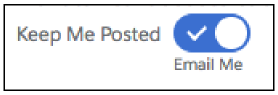
Flow Screen Input Component: Toggle

Let users flip a toggle in a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Note: This screen component requires Lightning runtime.

Configure the Toggle Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`Active Label`	When the toggle is active, this label appears underneath the toggle. Use it to clarify what active means. The default label is "Active." This attribute accepts single-value resources. The value is treated as text.
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Inactive Label`	When the toggle is inactive, this label appears underneath the toggle. Use it to clarify what inactive means. The default label is "Inactive." This attribute accepts single-value resources. The value is treated as text.
`Label`	This label appears next to the toggle and describes what the user is enabling. This attribute accepts single-value resources. The value is treated as text.

Attribute	Description
`Value`	Whether the toggle is active (*`$GlobalConstant.True`*) or inactive (*`$GlobalConstant.False`*). Setting this attribute from the Inputs tab controls the default state of the toggle. To store the user's selection in a flow variable, set this attribute from the Outputs tab. This parameter accepts single-value Boolean resources.

Store the Toggle Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but Value is the most likely attribute you must store.

To store the user's selection, map the Value attribute to a Boolean flow variable or a checkbox field on a record variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value. If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

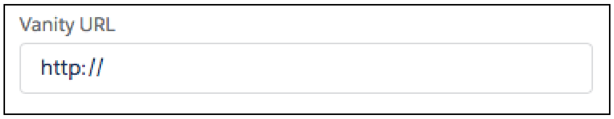
Flow Screen Input Component: URL

Let users enter URL values in a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Note: This screen component requires Lightning runtime.

Configure the URL Component

You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Disabled`	If set to true, the user can't modify the value. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Label`	The label that appears above the URL field. This attribute accepts single-value resources. The value is treated as text.
`Pattern`	Determines whether the value is valid. The default pattern verifies that the first character is a letter and that the value includes a colon (:). To force the user to enter a value in a specific format, use a regular expression. Make sure that your regular expression checks for a valid protocol in the URL, such as https:// or file:///. This example expression checks for a secure HTTP protocol (https://) and a specific domain (acmewireless.com). `^https?://(?:www\.)?acmewireless\.com/?.*` This attribute accepts single-value resources. The value is treated as text.
`Read Only`	If set to true, the user can't modify the value, but the user can copy it. The default value is false. This attribute accepts a resource with a Boolean value. Not supported in Classic runtime for flows.
`Required`	If set to true, the running user must enter a value. The default value is false. This attribute accepts a resource with a Boolean value.
`Value`	The value of the URL field. Setting this attribute prepopulates the field. To use the value that the user enters, store this attribute's output in a variable. This attribute accepts single-value resources. The value is treated as text.

Store the URL Component's Values in the Flow

The flow stores values automatically. If you store values manually, store the attribute's output value in a variable.

To store values manually, select **Manually assign variables (advanced)**.

All attributes are available to store in flow variables, but Value is the most likely attribute you must store.

To store the URL that the user entered, map the Value attribute to a flow variable.

Tip: By default, screen components that run on Lightning runtime version 58 and prior have no memory. If a user enters a value, and then does one of the following, the value is lost.

- Navigates to another screen and returns to the component's screen.
- Pauses the flow then resumes it.
- Navigates to the next screen and triggers an input validation error.

Setting the attribute enables a flow to remember the value. The flow stores the value automatically. If you store values manually, store the attribute's output value in a variable.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Validate Input

Provide a formula that evaluates whether what the user entered is valid and the error message to display if invalid.

Option	Description
`Error Message`	Specify the error message that appears below the component if the user enters an invalid value.
`Formula`	Provide a formula expression that returns a Boolean value.

Option	Description
	If the formula expression evaluates to `true`, the input is valid. If the formula expression evaluates to `false`, the error message appears below the component. If the user leaves the field blank and the field isn't required, the flow doesn't perform the validation. If the user leaves the field blank and the field is required, the flow shows the default error message and not your custom error message.

Specify the Behavior of Values on Revisited Screens

Specify what this component does when a user enters a value, navigates to a previous screen, and then returns to the screen with this component.

Option	Description
`Use values from when the user last visited this screen`	The component retains the values that the user specified and doesn't update the values to reflect changes made on previous screens.
`Refresh inputs to incorporate changes elsewhere in the flow`	The component updates the user-specified values to reflect changes made on previous screens. If you pause and then resume the flow, the flow retains user-specified values only in Checkbox, Checkbox Group, Currency, Long Text Area, Multi-Select Picklist, Number, Password, Picklist, Radio Buttons, and Text components.

SEE ALSO:

Standard Flow Screen Components

StackOverflow: Sample Regular Expressions for Valid URLs

MDN: What is a URL?

Flow Screen Output Component: Display Text

Display information in a flow screen.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Configure the Display Text Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
Text box	The text to display to the flow user. If you include a uniform resource identifier (URI), use one of these supported URI prefixes: • `http:`

Attribute	Description
	- `https:` - `//` - `/` - `file:` - `ftp:` - `mailto:` - `sfdc:` - `data:`

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Example: Display a confirmation message that summarizes what the flow did on the user’s behalf.

SEE ALSO:

Standard Flow Screen Components

Flow Screen Display Component: Repeater

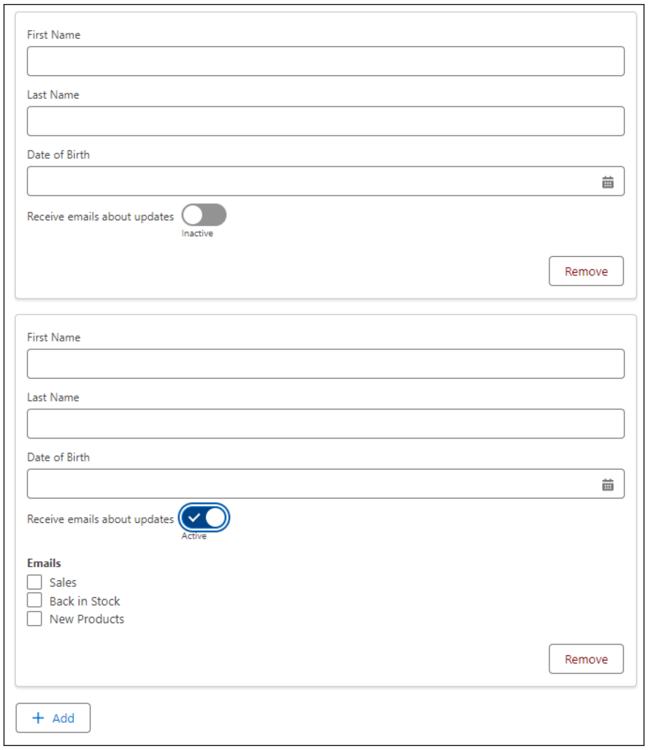
Collect information about multiple items of the same type on a screen with the Repeater component. To use the output of the component elsewhere in the flow, loop over the output and save the relevant data in a variable. Use the variable to build a list of records.

For the best performance, we recommend setting the flow and runtime to API version 58.0 and later.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



First Name

Last Name

Date of Birth

Receive emails about updates Inactive

Remove

First Name

Last Name

Date of Birth

Receive emails about updates Active

Emails

□ Sales
□ Back in Stock
□ New Products

Remove

+ Add

Configure the Repeater Component

Attribute	Description
`API Name`	The API name of the component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores. Screen readers use the API name to announce the Repeater component and its child components.

Configure Data Source

Select the collection of items that prepopulates the Repeater component at run time. The Repeater's child components can reference values from this collection.

Attribute	Description
`Collection for Prepopulated Items`	Fields from the selected collection become available to child components in the Repeater.
`Unique Identifier for Items`	The unique identifier for items is the API name of the field that contains a unique identifier for each item in the collection. This field is set automatically to the object's ID field.

Configure Display Options

Attribute	Description
`Let Users Add or Remove Items`	Choose whether screen flow end users can add new items or remove prepopulated items in your Repeater instance. End users can remove items that they added manually.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component. **When all conditions are met (AND)** Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

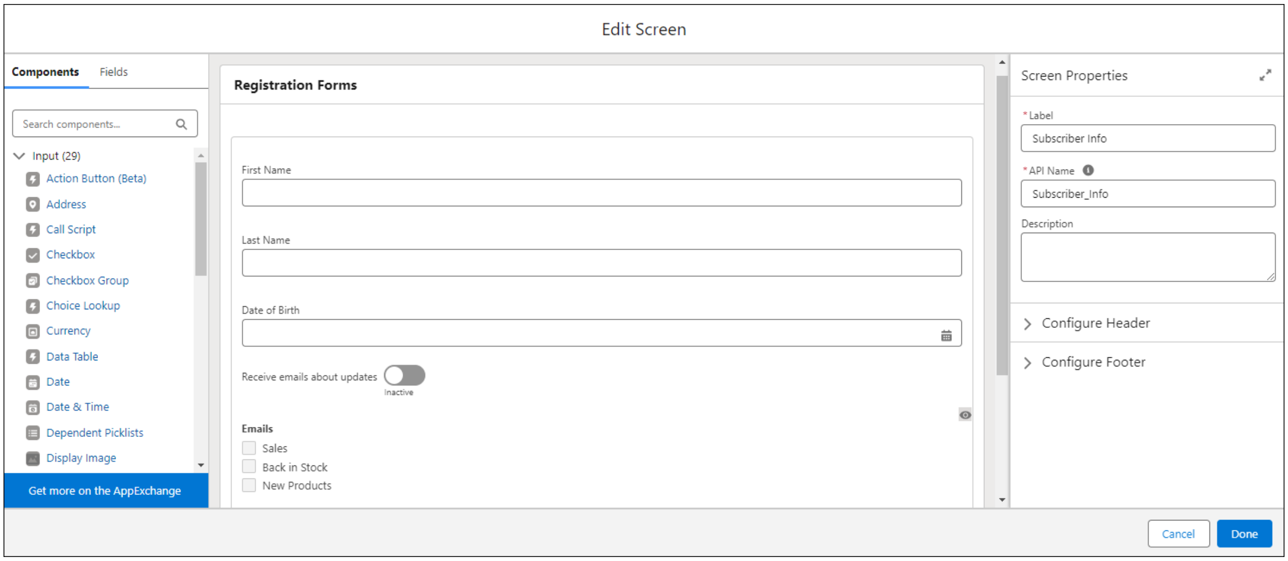
Usage

After you configure the Repeater component, add and configure one or more child components inside the Repeater. The flow stores user input for the Repeater component in the `AllItems` attribute of the component. You can loop over the items in this collection to create a collection variable that you can use later in the flow.

Considerations

- You can’t include the Action Button (Beta) component or record fields in a Repeater.
- The output of Repeater components isn’t supported in Transform, Collection Filter, or Collection Sort elements.
- You can’t reference the output of a different Repeater component in a Repeater child component.
- Choice components that reference a collection choice set resource in the Choice field aren’t reactive inside Repeater components.
- When you create or update a screen, you can move a component on the same screen into the Repeater component. You can also move a component from inside a Repeater component to a different place on the screen. However, any references to the moved component are broken.
- If you move a component with the Manually Assign Variables checkbox selected into a Repeater component, any manual assignments are removed and the checkbox is deselected. However, the variables still exist in the flow. We recommend reviewing the component after a move to ensure that it doesn’t include broken references.
- Users can add up to 30 instances of the Repeater component to the screen at runtime.
- The format for a reference to a Repeater component within the component itself is `{!`***`repeaterAPIName.fieldName`***`}`. In validation messages and the flow metadata package, the format for the same reference is `{!`***`repeaterAPIName`***`.AllItems[$`***`Items`***`].`***`fieldName`***`}`.
- The `AllItems` attribute is empty when:
  - The Repeater component contains only child components that don’t accept user input such as the Display Text component.
  - A user doesn’t add Repeater instances to the screen.
- The `AllItems` attribute is null when all the child components are hidden by conditional field visibility.

Example: This example shows a screen that includes a Repeater component with Text, Date, Toggle, and Checkbox Group child components to collect information about subscribers.



Edit Screen

Registration Forms

First Name

Last Name

Date of Birth

Receive emails about updates Inactive

Emails

□ Sales

□ Back in Stock

□ New Products

Screen Properties

*Label: Subscriber Info

*API Name: Subscriber_Info

Description

SEE ALSO:

Modify Records from User Input in Screens

Flow Example: Create a Contact for Each Beneficiary on a Policy

Flow Screen Output Component: Section

Organize screen components and record fields to give your users a better experience.

Note: This screen component requires Lightning runtime.

EDITIONS

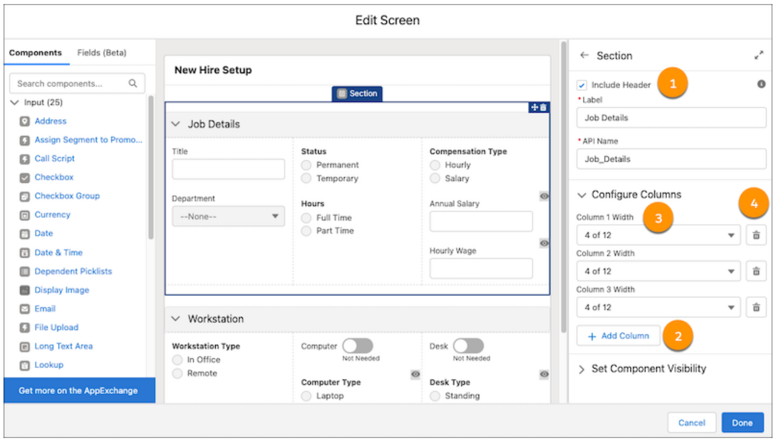
Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Usage

Use sections to organize screen components and fields to give users context and easier navigation. The Section component contains an optional header and up to four side-by-side columns. Each column can contain multiple components and fields. You can place multiple sections on a screen, each with its own header and number of columns.

Tip: Apply conditional visibility rules to a section to affect all components and fields in that section. Use this method to set visibility rules one time for a large number of components, even if you want only one column.



- Headers (1)—Use section headers to create a visual hierarchy to guide your users to the most important items on a screen. All sections with headers are collapsible and open by default each time a user visits the screen. Also, section header labels can be translated.
- Columns (2)—Use columns to organize your screen and save your users from unnecessary scrolling.
- Column Width (3)—When you add or delete a new column, Flow Builder sets the width of all columns in that section to be equal. To change a column’s width, select a width from the predefined options.
- Column Deletion (4)—When you delete a column, all components and fields in that column are deleted.

Tip: To center or indent your components and fields, or add padding, include empty columns on your screen.

Set the Component Visibility

Specify the logic that determines when the flow displays the component.

Option	Description
`When to Display Component`	Configure when the component is displayed using conditional logic. You can set the components to: **Always** Always display the component.

Option	Description
	When all conditions are met (AND) Display the component when all of the conditions that you define are met. Define at least one condition. **When any condition is met (OR)** Display the component when at least one of the conditions that you define is met. Define at least one condition. **When custom conditional logic is met** Display the component when the condition logic that you define is met. Define at least one condition and specify condition logic.

Considerations

- Sections are responsive to the size of the window that's showing the flow. On small form factor devices, columns are stacked vertically instead. However, it isn't responsive to the width of Lightning page columns and utility bars. For example, if a Lightning page shows a flow in a sidebar, the width of the entire window determines how the columns appear, even though the sidebar is narrower.
- If a screen contains a Section screen component, the screen ignores the Layout property when the flow is distributed in Experience Builder, the Lightning App Builder, or the utility bar. Screens with a Section screen component also ignore the `flowLayout` URL parameter when the flow is distributed via URL.

SEE ALSO:

Customize a Flow URL to Render Two-Column Screens

Set the Runtime Experience for URL-Based Flows

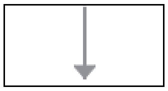
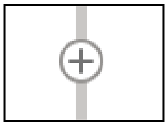
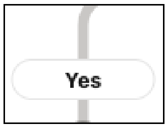
Flow Connectors

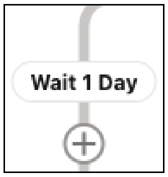
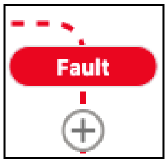
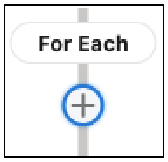
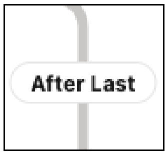
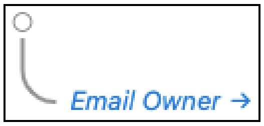
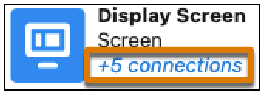
A connector determines the path that a flow takes at run time.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Type	Label	Example	Description
Default	*Unlabeled (Free-Form)*		Identifies which element to execute next.
Default	*Unlabeled (Auto-Layout)*		Identifies which element to execute next.
Decision	`Decision outcome label`		Identifies which element to execute when the criteria of a Decision element outcome are met.

Type	Label	Example	Description
Wait	*Wait configuration label*		Identifies which element to execute when an event that's defined in a Wait element occurs.
Fault	Fault		Identifies which element to execute when the previous element results in an error.
Loop	For each item		Identifies the first element to execute for each iteration of a Loop element.
Loop	After last item		Identifies which element to execute after a Loop element finishes iterating through a collection.
Outgoing Go To	*Destination element*		Identifies which element to go to and execute next.
Incoming Go To	+*x* connections		Identifies how many incoming go to connections an element has.

SEE ALSO:

Flow Elements

Move and Connect Elements to Change a Flow Route

Flow Operators

Operators behave differently, depending on what you're configuring. In Assignment elements, operators let you change resource values. In conditions and filters, operators let you evaluate information and narrow the scope of a flow operation.

Flow Operators in Assignment Elements

Use Assignment element operators to change the value of a selected resource.

Flow Operators in Decision, Wait, and Collection Filter Elements

Use condition operators to verify the value of a selected resource. Conditions are used in Decision, Wait, and Collection Filter elements.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flow Operators in Data Elements and Record Choice Sets

Filter conditions narrow the scope of records that the flow operates on. For example, use filter conditions to update only the contacts that are associated with the Acme Wireless account. When you add an Update Records element, use filter conditions to narrow the scope to just the contacts whose parent account is Acme Wireless. The In and Not In operators are available only in Create Records, Get Records, and Update Records elements.

Flow Operators in Assignment Elements

Use Assignment element operators to change the value of a selected resource.

Use this reference to understand the supported operators. The list is organized according to the data type that you select for Resource.

Note: Looking for the sObject data type from Cloud Flow Designer? In Flow Builder, we replaced sObject with the Record data type. So your sObject collection variables are now record collection variables.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Apex-Defined

Match the `@AuraEnabled` attribute's Apex data type with a flow data type in this reference to determine which operators are supported.

Apex Data Type	Flow Data Type
Boolean	Boolean
Date	Date
DateTime	Date/Time
Decimal	Number
Double	Number
Integer	Number
List	Collection
Long	Number
String	Text

Boolean

Replace a Boolean resource with a new value.

Operator	Description	Supported Data Types	Example
Equals	What you enter or select for Value replaces the value of Variable.	Boolean	Before Assignment: `{!varBoolean}` is `false` Assignment: `{!varBoolean}` **Equals** `{!$GlobalConstant.True}` After Assignment: `{!varBoolean}` is `true`

Collection

Update or replace the value of a collection variable or record collection variable.

Operator	Description	Supported Data Types	Example
Equals	Value replaces the value of Variable.	Collection of the same data type or object type Text, Picklist, and Multi-Select Picklist data types are compatible with each other.	Before the Assignment: - `{!collText}` is `Yellow, Green, Blue` - `{!collPicklist}` is `Blue, Red, Orange` Assignment: `{!collText}` **Equals** `{!collPicklist}` After the Assignment: `{!collText}` is `Blue, Red, Orange`
Add	Value is added as a new item at the end of the collection in Variable.	Variable of the same data type or record variable of the same object type Text, Picklist, and Multi-Select Picklist data types are compatible with each other. Stages (including $Flow.CurrentStage) can be added to text collections.	Before the Assignment: - `{!collText}` is `Yellow, Green, Blue` - `{!varPicklist}` is `Red` Assignment: `{!collText}` **Add** `{!varPicklist}` After the Assignment: `{!collText}` is `Yellow, Green, Blue, Red`
Remove After First	The first instance of Value is found within the collection in Variable. All items after the first instance are removed from the collection in Variable.	Variable of the same data type or record variable of the same object type For text collections only: - Multi-Select Picklist - Picklist - `$Flow.CurrentRecord`	Before the Assignment: - `{!collText}` is `Red, Orange, Yellow, Green, Blue` - `{!varText}` is `Yellow` Assignment: `{!collText}` **Remove After First** `{!varText}` After the Assignment: `{!collText}` is `Red, Orange, Yellow`
Add At Start	Value is added as a new item at the beginning of the collection in Variable.	Collection of the same data type or object type Variable of the same data type or record variable of the same object type For text collections only: - Multi-Select Picklist - Picklist - `$Flow.CurrentRecord`	Before the Assignment: - `{!collText}` is `Yellow, Green, Blue` - `{!varPicklist}` is `Red` Assignment: `{!collText}` **Add At Start** `{!varPicklist}` After the Assignment: `{!collText}` is `Red, Yellow, Green, Blue`

Operator	Description	Supported Data Types	Example
Remove All	All instances of Value are removed from the collection in Variable.	Collection of the same data type or object type Variable of the same data type or record variable of the same object type For text collections only: • Multi-Select Picklist • Picklist • `$Flow.CurrentRecord`	Before the Assignment: • `{!collText}` is `Red, Orange, Red, Yellow` • `{!varText}` is `Red` Assignment: `{!collText}` **Remove All** `{!varText}` After the Assignment: `{!collText}` is `Orange, Yellow`
Remove First	The first instance of Value is removed from the collection in Variable.	Collection of the same data type or object type For text collections only: • Multi-Select Picklist • Picklist • `$Flow.CurrentRecord`	Before the Assignment: • `{!collText}` is `Red, Orange, Red, Yellow` • `{!varText}` is `Red` Assignment: `{!collText}` **Remove First** `{!varText}` After the Assignment: `{!collText}` is `Orange, Red, Yellow`
Remove Before First	The first instance of Value is found within the collection in Variable. All items before the first instance are removed from the collection in Variable.	Variable of the same data type or record variable of the same object type For text collections only: • Multi-Select Picklist • Picklist • `$Flow.CurrentRecord`	Before the Assignment: • `{!collText}` is `Red, Orange, Yellow, Green, Blue` • `{!varText}` is `Yellow` Assignment: `{!collText}` **Remove Before First** `{!varText}` After the Assignment: `{!collText}` is `Yellow, Green, Blue`
Remove Position	Value specifies a position in the collection. The item at the position is removed from the collection in Variable. Make sure that Value at run time is a positive integer that is within the range of the number of items in the collection in Variable.	Number	Before the Assignment: • `{!collText}` is `Red, Orange, Yellow` • `{!varNum}` is `2` Assignment: `{!collText}` **Remove Position** `{!varNum}` After the Assignment: `{!collText}` is `Red, Yellow`
Remove Uncommon	The items in the Value collection are found within the Variable collection. The found items are kept, and all	Collection of the same data type or object type	Before the Assignment: • `{!collText1}` is `Red, Orange, Yellow, Green` • `{!collText2}` is `Orange, Green, Blue`

Operator	Description	Supported Data Types	Example
	other items are removed from the collection in Variable.		Assignment: `{!collText1}` **Remove Uncommon** `{!collText2}` After the Assignment: `{!collText1}` is `Orange, Green`

Currency and Number

Replace (Equals), add to (Add), or subtract from (Subtract) the value of a currency or number resource. Count (Equals Count) the number of active stages or the number of items in a collection.

Operator	Description	Supported Data Types	Example
Equals	The number that you enter or select for Value replaces the value of Variable.	• Currency • Number	Before the Assignment: `{!varCurrency}` is `10` Assignment: `{!varCurrency}` **Equals** `7` After the Assignment: `{!varCurrency}` is `7`
Add	The number that you enter or select for Value is added to the value of Variable.	• Currency • Number	Before the Assignment: `{!varCurrency}` is `10` Assignment: `{!varCurrency}` **Add** `7` After the Assignment: `{!varCurrency}` is `17`
Subtract	The number that you enter or select for Value is subtracted from the value of Variable.	• Currency • Number	Before the Assignment: `{!varCurrency}` is `10` Assignment: `{!varCurrency}` **Subtract** `7` After the Assignment: `{!varCurrency}` is `3`
Equals Count	The number of stages or collection items in what you enter for Value replaces the value of Variable.	• Collection • $Flow.ActiveStages	Before the Assignment: • `{!varNumber}` is `0` • `{!collText}` is `Yellow, Green, Blue` Assignment: `{!varNumber}` **Equals Count** `{!collText}` After the Assignment: `{!varNumber}` is `3`

Date

Replace (Equals), add to (Add), or subtract from (Subtract) the value of a date/time resource.

Operator	Description	Supported Data Types	Example
Equals	The date that you enter or select for Value replaces the value of Variable.	• Date • Date/Time	Before the Assignment: `{!varDate}` is `1/16/2016` Assignment: `{!varDate}` **Equals** `1/15/2016` After the Assignment: `{!varDate}` is `1/15/2016`

Operator	Description	Supported Data Types	Example
Add	Value is added, in days, to the selected Variable's value.	• Currency • Number	Before the Assignment: `{!varDate}` is `1/16/2016` Assignment: `{!varDate}` **Add** `7` After the Assignment: `{!varDate}` is `1/23/2016`
Subtract	Value is subtracted, in days, from the selected Variable's value.	• Currency • Number	Before the Assignment: `{!varDate}` is `1/16/2016` Assignment: `{!varDate}` **Subtract** `7` After the Assignment: `{!varDate}` is `1/9/2016`

Date/Time

Replace a date/time resource with a new value (Equals).

Operator	Description	Supported Data Types	Example
Equals	The date that you enter or select for Value replaces the value of Variable.	• Date • Date/Time	Before the Assignment: `{!varDateTime}` is `1/16/2016 01:00` Assignment: `{!varDateTime}` **Equals** `1/16/2016 08:00` After the Assignment: `{!varDateTime}` is `1/16/2016 08:00`

Picklist

Replace a picklist resource with a new value (Equals) or concatenate a value onto the original value (Add).

Note: Before values are assigned or added to a picklist resource, they're converted into string values.

Operator	Description	Supported Data Types	Example
Equals	What you enter or select for Value replaces the value of the selected picklist.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text	Before the Assignment: `{!varPicklist}` is `Blue` Assignment: `{!varPicklist}` **Equals** `Yellow` After the Assignment: `{!varPicklist}` is `Yellow`

Operator	Description	Supported Data Types	Example
Add	What you enter or select for Value is added to the end of the selected picklist.	- Boolean - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text	Before the Assignment: `{!varPicklist}` is `Blue` Assignment: `{!varPicklist}` **Add** `-green` After the Assignment: `{!varPicklist}` is `Blue-green`

Multi-Select Picklist

Replace a multi-select picklist resource with a new value (Equals), concatenate a value onto the original value (Add), or add a selection to the resource (Add Item).

Note: Before values are assigned or added to a multi-select picklist resource, they're converted into string values.

Operator	Description	Supported Data Types	Example
Equals	What you enter or select for Value replaces the value of the selected multi-select picklist.	- Boolean - Collection - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text	Before the Assignment: `{!varMSP}` is `Blue` Assignment: `{!varMSP}` **Equals** `Yellow` After the Assignment: `{!varMSP}` is `Yellow`
Add	What you enter or select for Value is added to the last item selected in the multi-select picklist. It doesn't create a selection. Easily add items to a multi-select picklist resource by using the "add item" operator. To add semi-colon-delimited items to a multi-select picklist variable with the "add" operator, always add	- Boolean - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text	Before the Assignment: `{!varMSP}` is `Blue; Green`. This value includes two separate selections Assignment: `{!varMSP}` **Add** `Yellow` After the Assignment: `{!varMSP}` is `Blue; GreenYellow`. This value includes two separate selections

Operator	Description	Supported Data Types	Example
	a single space after the semi-colon and don't include a space before the semi-colon. This way, you can compare the variable's values to the values of a multi-select picklist field from the Salesforce database. For example: `; Yellow`		
Add Item	What you enter or select for Value is added as a new selection to the end of the multi-select picklist. The Assignment automatically adds ";" before the new item. That way, Salesforce reads it as a separate item selected by the multi-select picklist.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text	Before the Assignment: `{!varMSP}` is `Blue; Green` Assignment: `{!varMSP}` **Add** item `Yellow` After the Assignment: `{!varMSP}` is `Blue; Green; Yellow`. This value includes three separate selections

Record

Replace a record variable with a new value (Equals).

Operator	Description	Supported Data Types	Example
Equals	The record variable that you select for Value replaces the value of Variable.	Record – with the same object type	Before the Assignment: • `{!account1}` contains field values for the Acme Wireless account • `{!account2}` contains field values for the Global Media account Assignment: `{!account1}` **Equals** `{!account2}` After the Assignment: both `{!account1}` and `{!account2}` contain the field values for the Global Media account

Stage

You can't update the value of a stage, but you can update the values of the stage global variables: `$Flow.CurrentStage` and `$Flow.ActiveStages`.

Note: Assignments use the stage's fully qualified name: *`namespace.flowName:stageName`* or *`flowName:stageName`*.

`$Flow.CurrentStage`

Replace the stage selected in `$Flow.CurrentStage`.

Operator	Description	Supported Data Types	Example
Equals	Value replaces the value of `$Flow.CurrentStage`.	• Stage • Fully qualified stage name	Before the Assignment: `$Flow.CurrentStage` is `stage1` Assignment: `{!$Flow.CurrentStage}` **Equals** `{!stage2}` After the Assignment: `$Flow.CurrentStage` is `stage2`

`$Flow.ActiveStages`

Add or remove active stages in the `$Flow.ActiveStages` global variable.

Operator	Description	Supported Data Types	Example
Add	Value is added to the end of `$Flow.ActiveStages`.	• Stage • `$Flow.ActiveStages` • `$Flow.CurrentStage` • Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2` Assignment: `{!$Flow.ActiveStages}` **Add** `{!stage3}` After the Assignment: `$Flow.ActiveStages` is `stage1, stage2, stage3`
Remove After First	The first instance of the stage in Value is found within `$Flow.ActiveStages`. All stages after the first instance are removed from `$Flow.ActiveStages`.	• Stage • `$Flow.CurrentStage` • Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2, stage3, stage4` Assignment: `{!$Flow.ActiveStages}` **Remove After First** `{!stage2}` After the Assignment: `$Flow.ActiveStages` is `stage1, stage2`
Add At Start	Value is added to the beginning of `$Flow.ActiveStages`.	• Stage • `$Flow.ActiveStages` • `$Flow.CurrentStage` • Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2` Assignment: `{!$Flow.ActiveStages}` **Add At Start** `{!stage0}` After the Assignment: `$Flow.ActiveStages` is `stage0, stage1, stage2`
Remove All	All instances of Value are removed from `$Flow.ActiveStages`.	• Stage • `$Flow.ActiveStages` • `$Flow.CurrentStage` • Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2, stage3` Assignment: `{!$Flow.ActiveStages}` **Remove All** `{!$Flow.ActiveStages}` After the Assignment: `$Flow.ActiveStages` is `empty`

Operator	Description	Supported Data Types	Example
Remove First	The first instance of the stage in Value is removed from `$Flow.ActiveStages`.	- Stage - `$Flow.CurrentStage` - Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2, stage3, stage1` Assignment: `{!$Flow.ActiveStages}` **Remove First** `stage1` After the Assignment: `$Flow.ActiveStages` is `stage2, stage3, stage1`
Remove Before First	The first instance of the stage in Value is found within `$Flow.ActiveStages`. All stages before that first instance are removed from `$Flow.ActiveStages`.	- Stage - `$Flow.CurrentStage` - Fully qualified stage name	Before the Assignment: `$Flow.ActiveStages` is `stage1, stage2, stage3, stage4` Assignment: `{!$Flow.ActiveStages}` **Remove Before First** `{!stage3}` After the Assignment: `$Flow.ActiveStages` is `stage3, stage4`
Remove Position	Value specifies a position in `$Flow.ActiveStages`. The stage at that position is removed from `$Flow.ActiveStages`. Make sure that Value at run time is a positive integer that is within the range of the number of stages in `$Flow.ActiveStages`.	Number	Before the Assignment: - `$Flow.ActiveStages` is `stage1, stage2, stage3` - `{!varNum}` is 2 Assignment: `{!$Flow.ActiveStages}` **Remove Position** `{!varNum}` After the Assignment: `$Flow.ActiveStages` is `stage1, stage3`

Text

Replace a text resource with a new value (Equals) or concatenate a value onto the end of the original value (Add).

Note: Before values are assigned or added to a text resource, they're converted into string values.

Operator	Description	Supported Data Types	Example
Equals	The text that you enter or select for Value replaces the value of Variable.	- Boolean - Currency - Date - Date/Time - Number - Multi-Select picklist - Picklist	Before the Assignment: `{!varText}` is `Blue` Assignment: `{!varText}` **Equals** `Yellow` After the Assignment: `{!varText}` is `Yellow`

Operator	Description	Supported Data Types	Example
		• Stage, including $Flow.CurrentStage and $Flow.ActiveStages • Text	
Add	The text that you enter or select for Value is added to the end of Variable.	• Boolean • Currency • Date • Date/Time • Number • Multi-Select picklist • Picklist • Stage, including $Flow.CurrentStage and $Flow.ActiveStages • Text	Before the Assignment: `{!varText}` is `Blue` Assignment: `{!varText}` **Add** `Yellow` After the Assignment: `{!varText}` is `BlueYellow`

Flow Operators in Decision, Wait, and Collection Filter Elements

Use condition operators to verify the value of a selected resource. Conditions are used in Decision, Wait, and Collection Filter elements.

Use this reference to understand the supported operators. The list is organized according to the data type that you select for Resource,

Note: Looking for the sObject data type from Cloud Flow Designer? In Flow Builder, we replaced sObject with the Record. So your sObject collection variables are now record collection variables.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Apex-Defined

Match the *`@AuraEnabled`* attribute's Apex data type with a flow data type in this reference to determine which operators are supported.

Apex Data Type	Flow Data Type
Boolean	Boolean
Date	Date

Apex Data Type	Flow Data Type
DateTime	Date/Time
Decimal	Number
Double	Number
Integer	Number
List	Collection
Long	Number
String	Text

Boolean

Check whether a Boolean resource's value matches another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value of the selected Resource doesn't match what you enter or select for Value.	Boolean
Equals	The value of the selected Resource matches what you enter or select for Value. • A decision outcome resolves to true if the flow interview took that outcome. • A wait configuration resolves to true if the flow waited for the associated resume event. • A data element or action element resolves to true if it executed without error. If a fault occurred, the element resolves to false. If the element wasn't executed, it resolves to null.	Boolean
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time.	Boolean
Was Visited	Only supported in Decision and Wait elements, the selected Resource is an element in the flow, and it has been visited during the flow interview.	Boolean

Choice

Every choice resource has a data type and obeys the operator rules for that data type. However, choice resources support one extra operator that other resources don't, no matter what their data type is.

Operator	True if...	Supported Data Types
Was Selected	A user selected that choice, picklist choice set, or record choice set in a screen component. If your flow references the same choice in multiple screens, Was Selected always respects the most recent screen that the flow visited. If your flow references the same choice with a text input in more than one place on the same screen, this operator always respects the first usage in the screen.	Boolean

Collection

Check whether a Collection resource's value contains or matches another value or resource.

Operator	True if...	Supported Data Types
Contains	An item in the collection that's selected for Resource contains the exact same value as Value	Resource of the same data type. For record collection variables, only record resources with the same object type are supported.
Does Not Equal	The collection that's selected for Resource doesn't match the collection that's selected for Value Two record collection variables are unequal if they include different fields or if the fields have different values.	Collection of the same data type. For record collection variables, only record collection variables with the same object type are supported.
Equals	The collection that's selected for Resource matches the collection that's selected for Value Two record collection variables are equal if they include the same fields and those fields have the same values.	Collection of the same data type. For record collection variables, only record collection variables with the same object type are supported.
Is Empty	An empty collection	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean

Currency and Number

Check whether a Currency or Number resource's value matches, is larger than, or is smaller than another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	• Currency • Number
Equals	The value for Resource matches what's entered or selected for Value	• Currency • Number

Operator	True if...	Supported Data Types
Greater Than	The value for Resource is larger than what's entered or selected for Value	• Currency • Number
Greater Than or Equal	The value for Resource is larger than what's entered or selected for Value or is the same	• Currency • Number
Less Than	The value for Resource is smaller than what's entered or selected for Value	• Currency • Number
Less Than or Equal	The value for Resource is smaller than what's entered or selected for Value or is the same	• Currency • Number
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time	Boolean

Date and Date/Time

Check whether a Date or Date/Time resource's value matches, is before, or is after another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	• Date • Date/Time
Equals	The value for Resource matches what's entered or selected for Value	• Date • Date/Time
Greater Than	The value for Resource is a later date or time than what's entered or selected for Value	• Date • Date/Time
Greater Than or Equal	The value for Resource is a later date or time than what's entered or selected for Value or is the same date or time	• Date • Date/Time
Less Than	The value for Resource is an earlier date or time than what's entered or selected for Value	• Date • Date/Time

Operator	True if...	Supported Data Types
Less Than or Equal	The value for Resource is an earlier date or time than what's entered or selected for Value or is the same date or time	- Date - Date/Time
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value..	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time	Boolean

Picklist

Check whether a Picklist resource's value matches or contains another value or resource.

Note: These operators treat the resource's value as a text value.

Operator	True if...	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value For example, if the value of `{!varPicklist}` is `yellow-green`, the condition `{!varPicklist}` **Contains** `green` evaluates to true.	- Boolean - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	- Boolean - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text
Equals	The value for Resource matches what's entered or selected for Value	- Boolean

Operator	True if...	Supported Data Types
		- Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time	Boolean

Multi-Select Picklist

Check whether a multi-select picklist resource's value matches or contains another value or resource.

Note: These operators treat the resource's value as a text value. If the resource's value includes multiple items, the operators treat the value as one string that happens to include semi-colons. It doesn't treat each selection as a different value. For example, the operators treat `red; blue; green` as a single value rather than three separate values.

Operator	True if...	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value. When you use this operator for a multi-select picklist resource, be aware of the values that a user can enter. If you want to check that a specific value is included and that value is also included as part of another value, create a flow formula resource that uses the INCLUDES() function. For example, your org has a Color multi-select picklist value. Among the possible values are "green" and "yellow-green". If both "green" and "yellow-green" are acceptable values, use the Contains operator in a flow condition. If only "green" is an acceptable value, create a formula that uses the INCLUDES() function.	- Boolean - Currency - Date - Date/Time - Multi-Select Picklist - Number - Picklist - Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value. Order matters. If you aren't sure which order the values that you're checking for appear, use the INCLUDES() function in a flow formula. For example, if you compare "red; blue; green" to "blue; green; red" using the Does Not Equal operator, that condition resolves to true.	- Boolean - Currency - Date - Date/Time

Operator	True if...	Supported Data Types
		• Multi-Select Picklist • Number • Picklist • Text
Equals	The value for Resource exactly matches what's entered or selected for Value. Order matters. If you aren't sure which order the values that you're checking for appear, use the INCLUDES() function in a flow formula. For example, if you compare "red; blue; green" to "blue; green; red" using the Equals operator, that condition resolves to false.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time	Boolean

Record

Check whether a record resource's value matches another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	Record with the same object type
Equals	The value for Resource matches what's entered or selected for Value	Record with the same object type
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean

Stage

Note: Stages resolve to the fully qualified stage name: `namespace.flowName:stageName` or `flowName:stageName`.

Check whether a Stage resource or the `$Flow.CurrentStage` global variable matches, ends with, or starts with another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	• Stage • Text
Equals	The value for Resource matches what's entered or selected for Value	• Stage • Text
Ends With	The end of the value for Resource matches what's entered or selected for Value	• Stage • Text
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 652.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Starts With	The beginning of the value for Resource matches what's entered or selected for Value	• Stage • Text

Check whether `$Flow.ActiveStages` contains a particular stage, matches the value of a Text collection, or is null.

Operator	True if...	Supported Data Types
Contains	`$Flow.ActiveStages` contains an item that matches the resource that's selected for Value.	• Stage • Text
Does Not Equal	The collection that's selected for Resource doesn't match `$Flow.ActiveStages`.	Text collection
Equals	The collection that's selected for Resource doesn't match `$Flow.ActiveStages`.	Text collection
Is Null	`$Flow.ActiveStages` value is either not set or references no value.	Boolean

Text

Check whether a Text resource's value matches, contains, ends with, or starts with another value or resource.

Note:

- Before values are compared to a text resource, they're converted into string values.
- Stages resolve to the fully qualified stage name: `namespace.flowName:stageName` or `flowName:stageName`.

Operator	True if...	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Equals	The value for Resource matches what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Ends With	The end of the value for Resource matches what's entered or selected for Value	• Boolean

Operator	True if...	Supported Data Types
		• Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Is Blank	The value for Resource is set to zero characters or only white space. Use to determine whether a field or variable value is set to zero characters or only white space.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Starts With	The beginning of the value for Resource matches what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value in the flow at least one time	Boolean

Flow Operators in Data Elements and Record Choice Sets

Filter conditions narrow the scope of records that the flow operates on. For example, use filter conditions to update only the contacts that are associated with the Acme Wireless account. When you add an Update Records element, use filter conditions to narrow the scope to just the contacts whose parent account is Acme Wireless. The In and Not In operators are available only in Create Records, Get Records, and Update Records elements.

Use this reference, organized by the data type of the field that you select, to understand the supported operators.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Checkbox Fields

When you select a checkbox field under Field, these operators are available. A flow treats `null` as a different value than `false`. If you filter for records whose checkbox field is null, no records are returned.

Operator	Filters to records where the selected field's value ...	Supported Data Types
Does Not Equal	Doesn't match what you enter or select for Value	Boolean
Equals	Matches what you enter or select for Value	Boolean
Is Null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
In	Matches a value in the collection of data that's entered for Value. If the collection is empty, the flow stores no records in the output. See Limits in Flow Data Considerations.	Boolean
Not In	Doesn't match a value in the collection of data that's entered for Value. If the collection is empty, the flow stores all records in the output. See Limits in Flow Data Considerations.	Boolean

Currency, Number, and Percent Fields

When you select a currency, number, or percent field under Field, these operators are available.

Operator	Filters to records where the selected field's value ...	Supported Data Types
Does Not Equal	Doesn't match what's entered or selected for Value	• Currency • Number
Equals	Matches what's entered or selected for Value	• Currency • Number
Greater Than	Is larger than what's entered or selected for Value	• Currency • Number
Greater Than or Equal	Is larger than what's entered or selected for Value or is the same	• Currency • Number
Is Null	Hasn't been populated with a value yet (if you select True for Value)	Boolean

Operator	Filters to records where the selected field's value ...	Supported Data Types
Less Than	Is smaller than what's entered or selected for Value	- Currency - Number
Less Than or Equal	Is smaller than what's entered or selected for Value or is the same.	- Currency - Number
In	Matches a value in the collection of data that's entered for Value. If the collection is empty, the flow stores no records in the output. See Limits in Flow Data Considerations.	- Currency - Number
Not In	Doesn't match a value in the collection of data that's entered for Value. If the collection is empty, the flow stores all records in the output. See Limits in Flow Data Considerations.	- Currency - Number

Date and Date/Time

When you select a date or date/time field under Field, these operators are available.

Operator	Filters to records where the selected field's value ...	Supported Data Types
Does Not Equal	Doesn't match what's entered or selected for Value	- Date - Date/Time
Equals	Matches what's entered or selected for Value	- Date - Date/Time
Greater Than	Is a later date or time than what's entered or selected for Value	- Date - Date/Time
Greater Than or Equal	Is a later date or time than what's entered or selected for Value or is the same date or time	- Date - Date/Time
Is Null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
Less Than	Is an earlier date or time than what's entered or selected for Value	- Date - Date/Time

Operator	Filters to records where the selected field's value ...	Supported Data Types
Less Than or Equal	Is an earlier date or time than what's entered or selected for Value or is the same date or time	• Date • Date/Time
In	Matches a value in the collection of data that's entered for Value. If the collection is empty, the flow stores no records in the output. See Limits in Flow Data Considerations.	• Date • Date/Time
Not In	Doesn't match a value in the collection of data that's entered for Value. If the collection is empty, the flow stores all records in the output. See Limits in Flow Data Considerations.	• Date • Date/Time

Picklist and Text Fields

When you select a picklist or text field under Field, these operators are available. The In and Not In operators don't support picklist fields.

Operator	Filters to records where the selected field's value ...	Supported Data Types
Contains	Contains what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Does Not Equal	Doesn't match what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text

Operator	Filters to records where the selected field's value ...	Supported Data Types
Equals	Matches what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Ends With	Ends with what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Is Null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
Starts With	Begins with what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
In	Matches a value in the collection of data that's entered for Value. If the collection is empty, the flow stores no records in the output. See Limits in Flow Data Considerations.	• Text
Not In	Doesn't match a value in the collection of data that's entered for Value. If the collection	• Text

Operator	Filters to records where the selected field's value ...	Supported Data Types
	is empty, the flow stores all records in the output. See Limits in Flow Data Considerations.	

Multi-Select Picklist Fields

When you select a multi-select picklist field under Field, these operators are available.

Tip: Be careful when using these operators to filter records based on a multi-select picklist field. Even if two resources have the same items in a multi-select picklist, they can be mismatched if these cases differ.

- The spacing before or after the semi-colon. For example, one resource's value is "red; green; blue" and the other's value is "red;green;blue"
- The order of the items. For example, one resource's value is "red; green; blue" and the other's value is "red; blue; green"

For best results, use the INCLUDES function in a flow formula.

Operator	Filters to records where the selected field's value ...	Supported Data Types
Does Not Equal	Doesn't match what's entered or selected for Value	Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Stage Text
Equals	Matches what's entered or selected for Value	Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Stage Text
Ends With	Ends with what's entered or selected for Value	Boolean Currency

Operator	Filters to records where the selected field's value ...	Supported Data Types
		• Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text
Is Null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
Starts With	Begins with what's entered or selected for Value	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Stage • Text

Flow Version Properties

A flow version's properties consist of its label, description, interview label, and type. These properties drive the field values that appear on the flow's detail page.

To change the properties of a flow version, open it in Flow Builder. Then click ⚙.

Property	Description
`Flow Label`	The label for the flow version. The label appears in the flow detail page and list views. When a user runs this flow, the header displays the flow label. You can edit the label for inactive flows and flow versions.
`Flow API Name`	The API name for the flow. The API name is used to refer to this flow from other parts of Salesforce, such as in a URL or Lightning web component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores. The API name appears on the flow detail page. You can't edit the API name after saving the flow.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Property	Description
`Description`	Differentiates the flow version from other versions. The description appears in the flow detail page and list views. You can edit the description for inactive flows and flow versions.
`Template`	Specifies whether the flow is a template. When a template is installed from a managed package, the subscriber can view the flow and save it as a new, editable flow. Non-template flows that are installed from managed packages can only be activated and deactivated. Suppose that your company needs a flow that differs slightly for each country where you do business. You can create or install a template for the base flow and then clone it to create each country-specific flow. Even if you don't use managed packages, you can use this field to clearly identify the base flow.
`How to Run the Flow`	Determines the context that the flow runs in. You can choose to always run flow in system context with sharing or system context without sharing. By default, the context that the flow runs in depends on how the flow is launched.
`Type`	Determines which elements and resources are supported in the flow and the ways that the flow can be implemented. The type appears in the flow detail page and list views. In flow version properties, the type appears in the Advanced section. For details, see Flow Types on page 24. Flow Builder doesn't support saving a new version as a different flow type. To change a flow's type, save it as a new flow. Before you change the flow type, make sure that the flow contains only the elements, resources, and functionality that the new flow type supports. You can fix some, but not all, compatibility issues in the new flow.
`API Version for Running the Flow`	Determines which versioned run-time behavior improvements the flow adopts. Changing this field requires the Manage Flow permission. Before you select a new API version, review all run-time improvements that were delivered between the currently selected API version and the new API version. You can find all flow and process run-time improvements for an API version in the Salesforce Release Notes. By default, when you create a flow, it runs in the latest API version. If an existing flow is saved as a new flow or flow version, the existing flow's run-time API version is used in the new flow or flow version. The run-time API version doesn't change as future Salesforce releases roll out. You decide when, if ever, to change the API version for running each flow version. This field lets you test and upgrade your flows one by one, and at your own pace. You can even opt to never adopt versioned updates for one or all your flows.
`Interview Label`	The label for the flow's interviews. An interview is a running instance of a flow. This label appears in the following areas. • List views of paused interviews in Setup • Paused Interviews component on the Home page or in an Experience Builder site • Paused Interviews item in the Salesforce mobile app • Actions & Recommendations component in a Lightning page

Property	Description
	You can edit the interview label for inactive flows and flow versions. By default, the interview label contains the flow name and the `{!$Flow.CurrentDateTime}` global variable. Use a text template to reference multiple resources in the label. For example, ***`Flow Name`*** `- {!Account.Name} - {!$Flow.CurrentDateTime}`.

SEE ALSO:

Change the Flow Run Context

API Version for Running a Flow

Automate Complex Processes with Orchestrations

As your company grows, so does the complexity of your workflows. Processes often require input from multiple users in multiple departments across multiple time zones. This increased complexity results in an increased amount of time spent waiting for each person to complete their task in the proper order. Flow Orchestration helps you streamline this process with orchestrations: multi-step processes that interact with multiple users and systems.

What Is Flow Orchestration?

An orchestration is a sequence of stages, each comprised of one or more steps. A stage can contain background, interactive, and MuleSoft steps.

Interactive steps have an assigned user and execute a designated screen flow. An admin places the Flow Orchestration Work Guide Lightning App Builder component on the page layout for the type of record where a person can complete the interactive step assigned to them. When an orchestration runs an interactive step, the designated user receives an email with a link to their assigned action. The assigned user clicks the link to go to the record where they complete their action in the Work Guide.

Background steps call an autolaunched flow that Salesforce executes. They can run synchronously or asynchronously and have no user interaction.

MuleSoft steps call a MuleSoft action that Salesforce executes. They run asynchronously and have no user interaction.

When Should You Use Flow Orchestration?

Use Flow Orchestration to create advanced approval processes, task lists for groups, or any other processes that require multiple interrelated steps. For example, consider employee onboarding that requires a new employee to go through a multi-level, multi-user, multi-system approval process to get equipment and access to digital company resources. Use Flow Orchestration to compose and orchestrate that complex process, and enjoy a top-level experience to manage and monitor every onboarding.

Flow Builder for Flow Orchestration

Get to know the Flow Builder requirements and user interface for Flow Orchestration.

Flow Orchestration Concepts

Learn about what an orchestration is made of and how it relates to flows.

Build an Orchestration

Use Flow Orchestration to build sophisticated business processes by combining and coordinating flows.

Deploy an Orchestration

After you design and test your orchestration, it's time to put it to work!

Orchestration Run

An orchestration run is created for each instance of an orchestration.

Manage Orchestrations and Work Items

Manage orchestrations and work items with list views. Cancel or suspend a running orchestration. Resume an orchestration run that failed within the previous 14 days because of an error in an action or flow called by a step. Or resume an orchestration run that was manually suspended. Reassign work items that have been assigned, but not completed.

Troubleshoot Orchestrations

To troubleshoot a failed orchestration run, use the orchestration fault email. To test an orchestration and observe what happens as it runs, use the debug option.

Flow Orchestration Limits and Considerations

When designing, managing, and running orchestrations, consider these issues.

Flow Orchestration Entitlements

Flow Orchestration has usage-based entitlements. An orchestration *run* is a running instance of an orchestration. An *orchestration* is an application built by your admin that uses stages, steps, and decisions to organize a complex business process.

Flow Orchestration Reference

Bookmark this page for quick access to information about orchestration elements, resources, events, and more.

Flow Builder for Flow Orchestration

Get to know the Flow Builder requirements and user interface for Flow Orchestration.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

EDITIONS

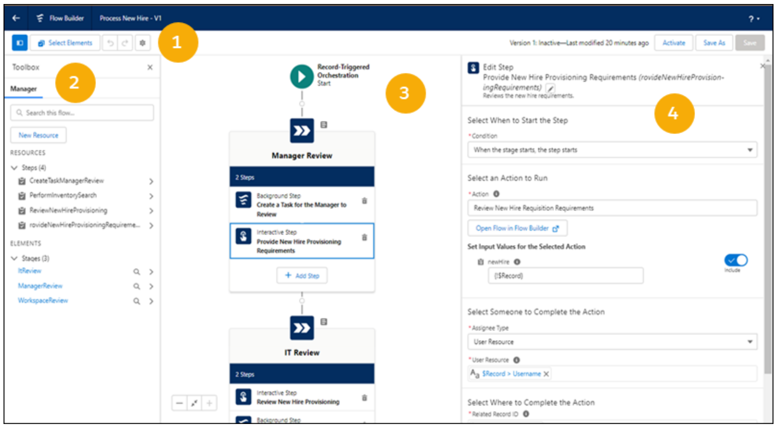
Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Tour the Flow Builder User Interface for Flow Orchestration

Flow Orchestration uses the Auto-Layout canvas in Flow Builder.



Version 1: Inactive—Last modified 20 minutes ago

Edit Step
Provide New Hire Provisioning Requirements *(rovideNewHireProvisioningRequirements)*
Reviews the new hire requirements.

Select When to Start the Step
*Condition: When the stage starts, the step starts

Select an Action to Run
*Action: Review New Hire Requisition Requirements

Set Input Values for the Selected Action
newHire: {!$Record}

Select Someone to Complete the Action
*Assignee Type: User Resource
*User Resource: $Record > Username

Select Where to Complete the Action
*Related Record ID

Button Bar (1)—Manage your orchestration as you build it.

- To run the most recent saved version of the orchestration that's open, click **Run**.

  Note: The Run button is only available for autolaunched orchestrations.

- To the left of the buttons, you can see the version's active or inactive status and when it was last saved.
- If the orchestration has warnings or errors, the Show Warnings icon (⚠) or the Show Errors (⊘) icon appears. To see their details, click the icon.

Toolbox (2)—Create variables, constants, formulas, or text templates to use in your orchestration. Or view a list of all resources and elements that you added.

Canvas (3)—Build an orchestration on the canvas. As you add elements to the canvas and connect them, you can see a diagram of your orchestration.

Note: To insert an element, in the desired location, click ⊕. Flow Builder then shows the options and possible elements for this location.

Details (4)—Set attributes for the element selected in the canvas. The Details panel closes when no element is selected.

Keyboard Shortcuts

Use these handy keyboard shortcuts for macOS and Windows to quickly navigate orchestrations.

Action	macOS	Windows
Zoom in	Cmd+Option+=	Ctrl+Alt+=
Zoom out	Cmd+Option+-	Ctrl+Alt+-
Zoom to fit	Cmd+Option+1	Ctrl+Alt+1
Zoom to view	Cmd+Option+0	Ctrl+Alt+0
View available keyboard shortcuts	Cmd+/	Ctrl+/

Flow Orchestration Concepts

Learn about what an orchestration is made of and how it relates to flows.

Orchestrations

An orchestration uses stages, steps, and decisions to organize complex business processes.

Building Blocks of Orchestrations

Stages and steps are the building blocks of an orchestration.

Anatomy of an Orchestration

Combine elements, connectors, and resources to build orchestrations.

Orchestration Types

An orchestration's type determines how the orchestration can be distributed.

Triggers for Orchestrations

Creating or updating a record can trigger an orchestration that requires additional input from users, approval from assigned users, other updates to the record, or changes to related records. In the Start element of a record-triggered orchestration, you can specify new and changed records of a specific object. Autolaunched orchestrations don't use triggers. Use another mechanism to launch an autolaunched orchestration, such as custom Apex classes or custom URLs. Use Flow Orchestration to automate complex processes, and use Flow Trigger Explorer to order record-triggered flows.

What's the Difference Between a Flow and an Orchestration?

Salesforce offers several features that automate internal procedures and business processes to save time across your org.

Advanced Orchestration Concepts

After you understand the basics, you're ready for a closer look at the context in which orchestrations run and how they perform work items at the same time.

Orchestrations

An orchestration uses stages, steps, and decisions to organize complex business processes.

Build orchestrations using the Flow Orchestration tiles in Flow Builder. Flow Orchestration tiles limit the available elements and available resources in your orchestration and include Stage elements and Step resources that aren't available in flows. Flow Orchestration always uses Auto-Layout in Flow Builder.

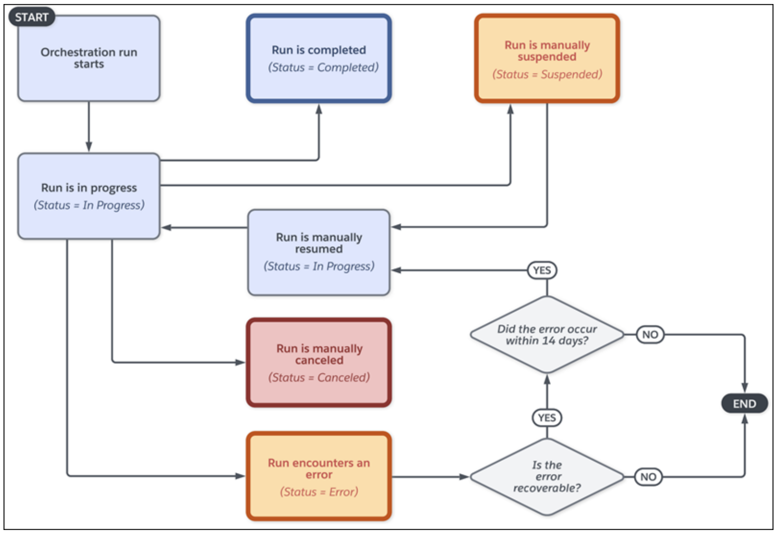
EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Orchestration Run Life Cycle



Flow Orchestration Types

The Flow Orchestration tiles are Autolaunched Orchestration (No Trigger) and Record-Triggered Orchestration and can be found on the All+Templates tab of the New Flow window. Trigger an autolaunched orchestration using a custom Apex class or a custom URL. The creation or update of a record can trigger a record-triggered orchestration, but only after the record is saved.

Variables in Orchestrations

Autolaunched orchestrations can use input variables to require input from a process that calls it.

To reference output values from flows called by orchestration steps, use the step's automatic output.

Variable	Description	Notes
Internal-only	Orchestrations don't have a way to assign a value to an internal-only variable.	
Input	The user-defined variable is marked Available for input. When the autolaunched orchestration is called by a process, its input variables are inputs from the parent process.	Orchestrations can't change the value of variables, but they can pass input variables to input parameters for evaluation flows or flows called by steps in an orchestration.
Output	Orchestrations don't use output variables.	

Record Refresh in Orchestrations

When you reference a record variable or a record collection in an orchestration configured to run on API version 58.0 and later, records are refreshed with their latest values each time the orchestration run resumes. In an autolaunched orchestration run, all referenced records are refreshed. In a record-triggered orchestration, all referenced records except $Record_Prior are refreshed.

Flow Orchestration Run Record Ownership

For flow orchestration run records created in Winter '23 or later, the Owner ID field is set to the ID of the automated process user.

SEE ALSO:

Use Automatic Output in Orchestrations

Flow Orchestration Resource: Global Variables

Building Blocks of Orchestrations

Stages and steps are the building blocks of an orchestration.

Orchestration Stages

A stage groups related steps, organizing them into a logical phase. Stages are executed sequentially, and only one stage in an orchestration can be in progress at a time. You configure the conditions that must be met for the stage to be considered complete.

Orchestration Steps

Steps are grouped in stages and can be run sequentially or concurrently. Interactive steps assign the completion of an active screen flow to a person, group, or queue and require user intervention. Background steps run an active autolaunched flow synchronously or asynchronously and have no user interaction. MuleSoft steps run an action imported from a MuleSoft operation and have no user interaction.

Flows in Orchestrations

Each background and interactive step in an orchestration runs an associated flow. If the logic for controlling stage and step execution calls for more than 3 requirements, use an evaluation flow to create more complex criteria.

Flow Orchestration Work Items

When an interactive step in an orchestration runs, it creates a work item and assigns it to a user, group, or queue. The orchestration run then sends an email with a link to the specified record page to all assigned users. They complete the work in the Orchestrator Work Guide component on the specified record page.

Orchestration Stages

A stage groups related steps, organizing them into a logical phase. Stages are executed sequentially, and only one stage in an orchestration can be in progress at a time. You configure the conditions that must be met for the stage to be considered complete.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

General

An orchestration must contain at least one stage. You can't control when a stage starts because stages run sequentially. To control when a stage completes, select one of the exit conditions.

Note: The Stage element in Flow Orchestration isn't related to the Stage resource in Flow Builder.

Exit Condition

To control when a stage completes, select an exit condition.

Stage Condition	Requires
When all steps have been marked Complete, the stage is marked Complete.	—
When the specified requirements are met, the stage is marked Completed	Up to three requirements that determine whether the stage can be completed
When the specified evaluation flow returns True, the stage is marked Complete.	The name of the evaluation flow that determines whether the stage can be completed

Automatic Output

An orchestration has access to a stage's status after it's in progress. At design time, however, automatic output resources are available throughout an orchestration, even before associated orchestration runs have access to the automatic output. This capability means that when you create an orchestration you must reference automatic output resources only when associated orchestration runs have access to it.

Status

When an orchestration is running, it manages the status for each stage. Because stages run sequentially and have no entry conditions, they only have a status after they're in progress. The corresponding orchestration stage run record is created after the stage is in progress.

Stage Status	Description
In Progress	The stage was started.
Completed	• The stage was completed, and the orchestration was completed. • The stage was completed, and the orchestration is in progress. • The stage was completed, and the orchestration was canceled.
Suspended	The stage was in-progress when the orchestration was manually suspended.
Canceled	The stage was in progress when the orchestration was canceled.
Discontinued	The stage was in progress when the orchestration encountered an error.
Error	• The stage encountered an error. • A background step within the stage encountered an error. • An autolaunched flow called by a background step within the stage encountered an error. • The stage was in progress when an interactive step within the stage encountered an error. • The stage was in progress when a screen flow associated with an interactive step within the stage encountered an error. • The stage was in progress when a MuleSoft step within the stage encountered an error.

Stage Status	Description
	• The stage was in progress when a MuleSoft action associated with a MuleSoft step within the stage encountered an error.

History

In history, an orchestration stage has several possible milestones.

Stage Milestone	Description
Start Stage	The stage started or was resumed when its parent orchestration was resumed.
End Stage	The stage was completed.
Suspend Stage	The stage was manually suspended.
Discontinue Stage	• The orchestration encountered an error after the stage was completed. • The orchestration encountered an error while the stage was in progress.
Fail Stage	• The stage encountered an error. • A background step within the stage encountered an error. • An autolaunched flow called by a background step within the stage encountered an error.

Flow Orchestration Stage Run Record Ownership

For flow orchestration stage run records created in Winter '23 or later, the Owner ID field is set to the ID of the automated process user.

SEE ALSO:

Evaluation Flows in Orchestrations

Orchestration Steps

Steps are grouped in stages and can be run sequentially or concurrently. Interactive steps assign the completion of an active screen flow to a person, group, or queue and require user intervention. Background steps run an active autolaunched flow synchronously or asynchronously and have no user interaction. MuleSoft steps run an action imported from a MuleSoft operation and have no user interaction.

Note: The Step resource in Flow Orchestration isn't related to the discontinued Step element in Flow Builder.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Automatic Output

At design time, automatic output resources are available throughout an orchestration, even before associated orchestration runs have access to the automatic output. This capability means that when you create an orchestration you must reference automatic output resources only when associated orchestration runs have access to it.

Note: To allow an orchestration access to a user-defined output variable in a flow called by a step, mark it as **Available for output** in the flow.

Note: An orchestration uses the isOrchestrationConditionMet output variable in evaluation flows. All other user-defined output variable values are discarded.

Table 2: Orchestration Run Access to Automatic Output

Automatic Output	Run-Time Availability
Stage Status	After the stage is in progress
Step Status	After the parent stage is in progress
Flow Output Variable	After the parent step has been completed

Step Milestone	Description
Start Step	The step started.
End Step	The step was completed.
Discontinue Step	• The step's stage was completed while the step was in progress. • The orchestration encountered an error after the step was completed. • The orchestration encountered an error while the step was in progress. • The step's stage encountered an error while the step was in progress.
Fail Step	The step encountered an error.

Status

When an orchestration is running, it manages the status for each step.

Step Status	Description
Not Started	The step hasn't met its entry condition.
In Progress	The step was started.
Completed	• The interactive step met its exit condition. • The background step's flow was completed. • The step was completed when its associated stage encountered an error.
Discontinued	• The step was in progress when its associated stage completed. • The step was in progress when the orchestration encountered an error. • The step was in progress when its associated stage encountered an error.
Error	• The step encountered an error. • The autolaunched flow associated with a background step encountered an error. • The screen flow associated with an interactive step encountered an error. • The MuleSoft action associated with a MuleSoft step encountered an error.

History

In history, a step in an orchestration has several possible milestones.

Step Milestone	Description
Start Step	The step started.
End Step	The step was completed.
Discontinue Step	• The step's stage was completed while the step was in progress. • The orchestration encountered an error after the step was completed. • The orchestration encountered an error while the step was in progress. • The step's stage encountered an error while the step was in progress.
Fail Step	The step encountered an error.

Flow Orchestration Step Run Record Ownership

For flow orchestration step run records created in Winter '23 or later, the Owner ID field is set to the ID of the automated process user.

Flow Orchestration Background Steps

A background step launches an active autolaunched flow and has no user interaction. You can control when a background step is ready to start.

Flow Orchestration Interactive Steps

An interactive step launches an active screen flow and requires user interaction. You can control when an interactive step is ready to start or when its status is set to completed.

Flow Orchestration MuleSoft Steps

A MuleSoft step asynchronously runs an operation imported from a MuleSoft API and has no user interaction. You can control when a MuleSoft step is ready to start.

Flow Orchestration Background Steps

A background step launches an active autolaunched flow and has no user interaction. You can control when a background step is ready to start.

Note: The Step resource in Flow Orchestration isn't related to the discontinued Step element in Flow Builder.

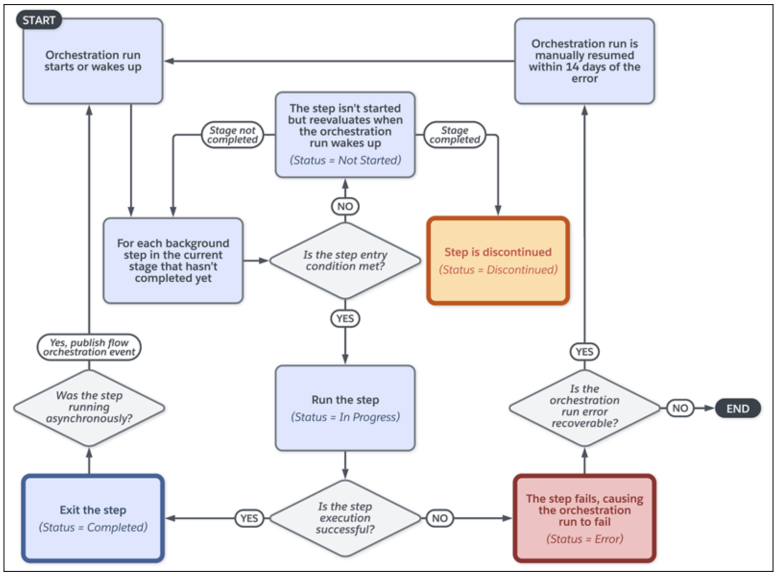
EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Background Step Work Cycle


START
Orchestration run starts or wakes up
Orchestration run is manually resumed within 14 days of the error
Stage not completed
The step isn't started but reevaluates when the orchestration run wakes up
(Status = Not Started)
Stage completed
NO
For each background step in the current stage that hasn't completed yet
Is the step entry condition met?
Step is discontinued
(Status = Discontinued)
YES
Yes, publish flow orchestration event
YES
Was the step running asynchronously?
Run the step
(Status = In Progress)
Is the orchestration run error recoverable?
NO
END
Exit the step
(Status = Completed)
YES
Is the step execution successful?
NO
The step fails, causing the orchestration run to fail
(Status = Error)

Asynchronous Background Step

By default, background steps are processed synchronously. When you select **Contains external callouts or wait elements**, a background step is processed asynchronously. Use an asynchronous background step when the background step calls an autolaunched flow that contains a Pause or Wait element or an external callout.

When the autolaunched flow called by an asynchronous step is completed, it publishes a Flow Orchestration Event platform event. That event causes the orchestration to evaluate the status of the current stage and each step with a status of Not Started or In Progress contained within the stage.

When to Start the Step

To control when a background step starts, select a condition.

Step Condition	Requires
When the stage starts, the step starts	—
When another step is marked Complete, the step starts	The name of the step that must be completed before the step can start
When the specified requirements are met, the step starts	Up to three requirements that determine whether the step can start
When the specified evaluation flow returns True, the step starts	The name of the active evaluation flow that determines whether the step can start

Running Context of an Action Called by a Background Step

For API version 60.0 and later, by default, an active autolaunched flow called by a background step runs in the context of the Automated Process User. To run a background step in the context of a different user, use the Select Who to Run the Action As section in the background step's Properties panel. To control the system context's record-level access, use the How to Run the Flow advanced option of the autolaunched flow.

For API version 59.0 and earlier, an active autolaunched flow called by a background step runs in the same context that the orchestration runs in.

Table 3: Running Contexts of Background Steps in API Version 59.0 and Earlier

Orchestration Runs In	Associated Flow Runs In
System context	• System context • To control the sharing access, use the How to Run the Flow advanced option of the autolaunched flow.
User context	• User context • To control the permission and sharing access, use the How to Run the Flow advanced option of the autolaunched flow.

For API version 59.0 and earlier, the context that an active autolaunched flow called by an asynchronous background step runs in depends on the context of the parent orchestration run

Table 4: Running Contexts of Asynchronous Background Steps in API Version 59.0 and Earlier

Orchestration Runs In	Associated Flow Runs In
System context	• The Automated Process user in system context • Control sharing access with the How to Run the Flow advanced option of the autolaunched flow
User context	• The same context that the orchestration ran in before the step • Control the permission and sharing access with the How to Run the Flow advanced option of the autolaunched flow

SEE ALSO:

Evaluation Flows in Orchestrations

Flow Orchestration Interactive Steps

An interactive step launches an active screen flow and requires user interaction. You can control when an interactive step is ready to start or when its status is set to completed.

Note: The Step resource in Flow Orchestration isn't related to the discontinued Step element in Flow Builder.

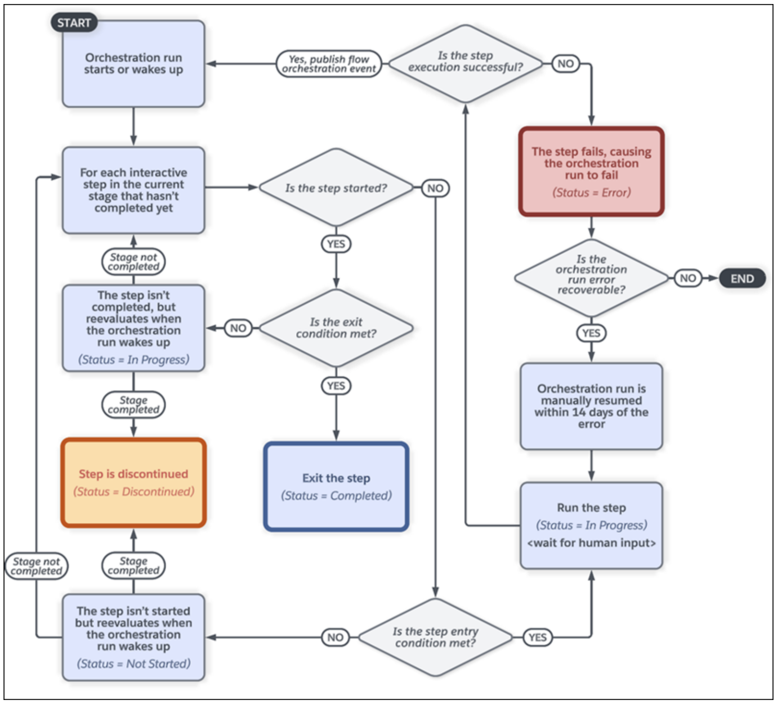
EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Interactive Step Work Cycle


START
Orchestration run starts or wakes up
Yes, publish flow orchestration event
Is the step execution successful?
NO
The step fails, causing the orchestration run to fail
(Status = Error)
For each interactive step in the current stage that hasn't completed yet
Is the step started?
NO
YES
Is the orchestration run error recoverable?
NO
END
YES
Stage not completed
The step isn't completed, but reevaluates when the orchestration run wakes up
(Status = In Progress)
NO
Is the exit condition met?
YES
Orchestration run is manually resumed within 14 days of the error
Stage completed
Step is discontinued
(Status = Discontinued)
Exit the step
(Status = Completed)
Run the step
(Status = In Progress)
<wait for human input>
Stage not completed
Stage completed
The step isn't started but reevaluates when the orchestration run wakes up
(Status = Not Started)
NO
Is the step entry condition met?
YES

For flows running in version 57.0 and earlier, after an interactive step is marked as complete, an orchestration run resumes in the context of the user who completed the associated work item. If the person who completed a work item has granular access to specific flows without the Run Flows permission, the orchestration run can't resume. To resume the orchestration, someone with the Run Flows permission can run another work item or an admin can trigger a Flow Orchestration Event with the ID of the paused orchestration run.

When to Start the Step

To control when an interactive step starts, select a condition.

Step Condition	Requires
When the stage starts, the step starts	—
When another step is marked Complete, the step starts	The name of the step that must be completed before the step can start
When the specified requirements are met, the step starts	Up to three requirements that determine whether the step can start
When the specified evaluation flow returns True, the step starts	The name of the active evaluation flow that determines whether the step can start

When to Complete the Step

To control when an interactive step completes, select a condition.

Step Condition	Requires
When the assigned user has completed the screen flow, the step is marked Complete	—
When the specified requirements are met, the step is marked Completed	Up to three requirements that determine whether the interactive step can be completed
When the specified evaluation flow returns True, the step is marked Complete	The name of the evaluation flow that determines whether the interactive step can be completed

Who Completes the Step

When an orchestration is designed, an interactive step is assigned to a user, group, or queue.

Table 5: Interactive Step Assignees

Type	Description
User	• The username of an internal user • The username of a credentialed Experience Cloud site visitor
Group	• The API name of a group that includes only internal users • The API name of a group that includes only credentialed Aura site visitors • The API name of a group that includes only credentialed LWR site visitors
Queue	• The API name of a queue that includes only internal users • The API name of a queue that includes only credentialed Aura site visitors • The API name of a queue that includes only credentialed LWR site visitors

At run time, assigned users, groups, and queues receive a notification email with a link to their assigned work by default. You can stop Flow Orchestration from sending these email notifications, but you can’t customize the default email. See Disable Default Email Notifications for Work Item Assignments.

Assignee Type	Notification
Internal user	Personalized email notification with a link to the internal related record page You can't customize the email notification.

Assignee Type	Notification
Credentialed Experience Cloud site visitor	Personalized email notification with a link to the related record page of the oldest live Experience Cloud site that they're a member of You can't customize the email notification.
Group or queue of internal users	General email notification with a link to the internal related record page
Group or queue of credentialed Aura or LWR site visitors who are all members of the same site	General email notification with a link to the related record page of the live Experience Cloud site that they're all members of
Group or queue of credentialed Aura or LWR site visitors who are all members of one or more common sites	General email notification with a link to the record page of the oldest live Experience Cloud site that they're all members of
Group or queue of credentialed Aura or LWR site visitors who have no site in common	General email notification with a link to the internal related record page The link is invalid for Experience Cloud site visitors, because they don't have access to internal pages. They can use the Orchestration Work Item Object List page to view and access their assigned work items from an Experience Cloud site that they're a member of.
Group or queue of internal users and credentialed Aura or LWR site visitors	General email notification with a link to the internal related record page The link is invalid for Experience Cloud site visitors, because they don't have access to internal pages. They can use the Orchestration Work Item Object List page to view and access their assigned work items from an Experience Cloud site that they're a member of.

Where to Complete the Step

The assignee or a person from the assigned group or queue completes the associated screen flow on a related record page. A link to this related record page is included in the email sent to the assigned person, group, or queue.

Running Context of a Flow Called by an Interactive Step

An active screen flow called by an interactive step runs in the context of the person who's completing it.

SEE ALSO:

Running Context of an Orchestration

Evaluation Flows in Orchestrations

Flow Orchestration MuleSoft Steps

EDITIONS

Available in: Lightning Experience

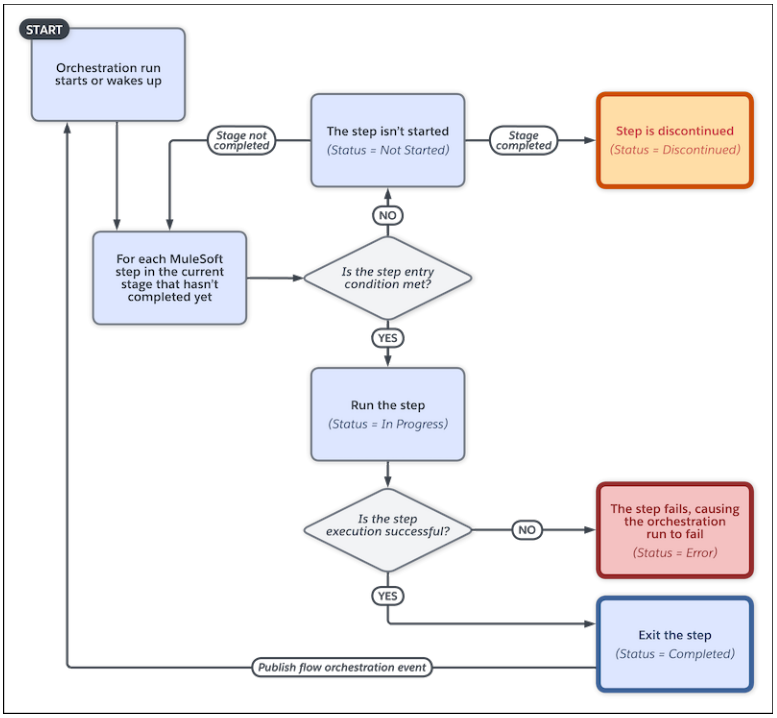
Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

A MuleSoft step asynchronously runs an operation imported from a MuleSoft API and has no user interaction. You can control when a MuleSoft step is ready to start.

Note: The Step resource in Flow Orchestration isn't related to the discontinued Step element in Flow Builder.

MuleSoft Step Work Cycle


START
Orchestration run starts or wakes up
Stage not completed
The step isn't started (Status = Not Started)
Stage completed
Step is discontinued (Status = Discontinued)
NO
For each MuleSoft step in the current stage that hasn't completed yet
Is the step entry condition met?
YES
Run the step (Status = In Progress)
Is the step execution successful?
NO
The step fails, causing the orchestration run to fail (Status = Error)
YES
Exit the step (Status = Completed)
Publish flow orchestration event

When to Start the Step

To control when a MuleSoft step starts, under Select When to Start the Step, select a condition.

Step Condition	Requires
When the stage starts, the step starts	—
When another step is marked Complete, the step starts	The name of the step that must be completed before the step can start
When the specified requirements are met, the step starts	Up to three requirements that determine whether the step can start

Step Condition	Requires
When the specified evaluation flow returns True, the step starts	The name of the active evaluation flow that determines whether the step can start

Running Context of a MuleSoft Action Called by a MuleSoft Step

For API version 60.0 and later, by default, a MuleSoft action called by a MuleSoft step runs in the context of the Automated Process User. To run a MuleSoft action in the context of a different user, use the Select Who to Run the Action As section in the MuleSoft step's Properties panel.

For API version 59.0 and earlier, a MuleSoft action called by a MuleSoft step runs in the context of the user that the orchestration ran as before the MuleSoft step starts.

SEE ALSO:

Evaluation Flows in Orchestrations

Flows in Orchestrations

Each background and interactive step in an orchestration runs an associated flow. If the logic for controlling stage and step execution calls for more than 3 requirements, use an evaluation flow to create more complex criteria.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Background Steps

Each background step calls an autolaunched flow.

Interactive Steps

Each interactive step assigns a screen flow to a user, group, or queue.

When to Start the Step

Each step can call an evaluation flow to determine whether the step can be started. An evaluation flow is a flow with a process type of Evaluation Flow. It's an autolaunched flow that contains a predefined Boolean output variable named `isOrchestrationConditionMet`. To indicate that the custom entry conditions are met, the output variable must be set to true.

Note: The Boolean `isOrchestrationConditionMet` variable defined in an evaluation flow must be initialized to false.

When to Complete the Step

An interactive step or a stage can call an evaluation flow to determine whether the step can be considered complete. An evaluation flow is a flow with a process type of Evaluation Flow. It's an autolaunched flow that contains a predefined Boolean output variable named `isOrchestrationConditionMet`. To indicate that the custom exit conditions are met, the output variable must be set to true.

Note: The Boolean `isOrchestrationConditionMet` variable defined in an evaluation flow must be initialized to false.

Flow Variables

Flows can have internal-only, input, and output variables.

If the combined input values for a flow called by an orchestration step is more than 32,768 characters, the orchestration fails. This error can be caused by passing one or more records to a flow called by a step. To avoid this error, pass a record ID to the referenced flow, and use a Get Records element in the flow with the passed ID. Using a passed ID with a Get Records element also means that you always have the latest version of the record.

Variable	Description	Notes
Internal-only	The variable is used inside the flow and isn't available for input or output in an orchestration.	You can create your own internal variables, or you can let flow create them for you. Orchestrations can't use these variables.
Input	The user-defined variable is marked Available for input. When the flow is called by an orchestration step, its input variables are inputs for the step.	Flows that can have input values from an orchestration for input variables are: • Autolaunched flows • Screen flows • Evaluation flows
Output	The user-defined variable is marked Available for output. Output variables are accessible as automatic outputs for the associated step and are available throughout the orchestration.	Orchestrations can access user-defined output variables from: • Autolaunched flows • Screen flows When you reference automatic output that contains a record or a record collection in an orchestration configured to run on API version 58.0 and later, records are refreshed with their latest values each time the orchestration run resumes. To add custom comments to the Orchestration Run Log, create a text output variable named Comments in a flow called by an orchestration step and assign it a value.

Evaluation Flows in Orchestrations

When you need more than 3 requirements to control stage and step execution, use an evaluation flow. Select the Evaluation Flow tile in the New Flow window to create an evaluation flow.

SEE ALSO:

Flow Types

Automate Tasks with Flows

Evaluation Flows in Orchestrations

When you need more than 3 requirements to control stage and step execution, use an evaluation flow. Select the Evaluation Flow tile in the New Flow window to create an evaluation flow.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Variables in Evaluation Flows

When you select the Evaluation Flow tile in the New Flow window, you create an evaluation flow that contains a predefined Boolean output variable named `isOrchestrationConditionMet`.

Initialize `isOrchestrationConditionMet` to false, and to indicate that the custom conditions are met, set `isOrchestrationConditionMet` to true.

Evaluation flows only return a value for `isOrchestrationConditionMet`. Values for any other output variables are discarded.

Evaluation Flow Execution

Every time an asynchronous background step, an interactive step, or a MuleSoft step within the current stage is completed, the orchestration evaluates the conditions for that stage and its steps. To trigger an evaluation of conditions for the current stage and its steps, publish an orchestration event with $Orchestration.Instance

The status of each stage or step determines which conditions the orchestration checks. If the condition being checked relies on an evaluation flow, the evaluation flow runs.

- When a stage is in progress, the orchestration determines whether it can be completed.
- For each not started step within the current stage, the orchestration determines whether the step is ready to start.
- For each in progress interactive step within the current stage, the orchestration determines whether the step can be marked complete.

Running Context of an Evaluation Flow

In API version 60.0 and later, evaluation flows can be run only in system context without sharing and have access to all data.

In API version 58.0 and 59.0, evaluation flows always run in system context.

In API version 57.0 and earlier, evaluation flows run as specified in the flow's How to Run the Flow advanced option.

SEE ALSO:

Trigger an Evaluation of Orchestration Stage and Step Conditions

Flow Orchestration Work Items

When an interactive step in an orchestration runs, it creates a work item and assigns it to a user, group, or queue. The orchestration run then sends an email with a link to the specified record page to all assigned users. They complete the work in the Orchestrator Work Guide component on the specified record page.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Who Gets Email Notifications When Work Is Assigned

By default, when an interactive step runs, the orchestration creates a work item and assigns it to the specified user, group, or queue. The type of notification emailed to the assignee depends on the type of assignee.

To disable default email notifications to all assigned users, see Disable Default Email Notifications for Work Item Assignments.

Assignee Type	Notification
Internal user	Personalized email notification with a link to the internal related record page You can't customize the email notification.
Credentialed Experience Cloud site visitor	Personalized email notification with a link to the related record page of the oldest live Experience Cloud site that they're a member of You can't customize the email notification.
Group or queue of internal users	General email notification with a link to the internal related record page
Group or queue of credentialed Aura or LWR site visitors who are all members of the same site	General email notification with a link to the related record page of the live Experience Cloud site that they're all members of
Group or queue of credentialed Aura or LWR site visitors who are all members of one or more common sites	General email notification with a link to the record page of the oldest live Experience Cloud site that they're all members of
Group or queue of credentialed Aura or LWR site visitors who have no site in common	General email notification with a link to the internal related record page The link is invalid for Experience Cloud site visitors, because they don't have access to internal pages. They can use the Orchestration Work Item Object List page to view and access their assigned work items from an Experience Cloud site that they're a member of.
Group or queue of internal users and credentialed Aura or LWR site visitors	General email notification with a link to the internal related record page The link is invalid for Experience Cloud site visitors, because they don't have access to internal pages. They can use the Orchestration Work Item Object List page to view and access their assigned work items from an Experience Cloud site that they're a member of.

When Assignment Notifications Are Made with the Omni-Channel Widget

When an interactive step is assigned to a queue associated with the Orchestration Work Item object, the queue members receive notifications based on your defined routing logic in the Omni-Channel widget. The notification by Omni-Channel widget is in addition to the default email notification sent to queue members.

Internal User Access to Work Items

Internal users get a link in their email notification to the related record page where they can complete their assigned work item. They can also view and access their assigned work in the Flow Orchestration Work Items list view or in their To Do List. To allow internal users to complete assigned work, before the orchestration runs, place the Flow Orchestration Work Guide component on the related record page layout in Lightning App Builder.

Experience Cloud Site Visitor Access to Work Items

Credentialed Experience Cloud site visitors usually get a link in their email notification to the related record page of the oldest live Experience Cloud site that they're a member of. They can also view and access their assigned work in the Orchestration Work Item object list view.

Before the orchestration runs, in Experience Builder, the admin sets up site visitor access to orchestration work items.

- In Experience Builder, the admin places the Flow Orchestration Work Guide component on the related record page in Aura and LWR sites.
- In Experience Builder, the admin adds the Orchestration Work Item List object page to Aura and LWR sites.

Work Assigned to an Internal User

When an interactive step runs, the orchestration creates a work item and assigns it to the specified internal user. The assigned user receives an email with a link to the internal related record page notifying them that they have a work item to complete. The work item also appears in the assigned user's To Do List. When the user clicks the link to the work item, they then can complete the screen flow associated with the interactive step in the Work Guide on the internal related record page.

Work Assigned to a Credentialed Experience Cloud Site Visitor

When an interactive step runs, the orchestration creates a work item and assigns it to the specified credentialed Experience Cloud site visitor. The assignee receives an email that notifies them that they have a work item to complete. The email contains a link to the related record page on the oldest live site that they're a member of. When the site visitor clicks the email link, they then complete the associated screen flow in the Work Guide on the related record page that they've been directed to.

Work Assigned to an Internal Group or Queue

When the interactive step runs, the orchestration creates a work item and assigns it to the specified group or queue. All users in the assigned internal group or internal queue receive an email with a link to the internal related record page notifying them that they have an action to complete. When a user clicks the link in the email and the work item opens, the user can run the screen flow in the Work Guide on the internal related record page.

An assigned work item is completed by the first user to complete the screen flow. If two users execute the screen flow simultaneously, the user who completes the flow second receives an error. After the work item is completed, other users from the assigned group or queue see no related work in the Work Guide component on the internal related record page.

Work Assigned to a Group or Queue of Credentialed Experience Cloud Site Visitors

When the interactive step runs, the orchestration creates a work item and assigns it to the specified group or queue of credentialed Experience Cloud site visitors. Site visitors in an assigned group or queue receive an email notifying them that they have an action to complete. The email usually contains a link to the related record page of the oldest live Experience Cloud site that the visitors are all members of. When a group or queue member clicks the link, the visitor is taken to the related record page of the oldest live Experience Cloud site that all group or queue members are credentialed for. From that related record page, the site visitor can complete the associated screen flow in the Work Guide.

Experience Cloud site visitors can also view work items assigned to them in the Orchestration Work Item List object page. From the Orchestration Work Item List, they can also access their assigned work. When the credentialed site visitor goes to the related record page of the Aura or LWR site that they're a member of, the visitor can run the screen flow in the Work Guide.

An assigned work item is completed by the first credentialed site visitor to complete the screen flow. If two credentialed site visitors execute the screen flow simultaneously, the one who completes the flow second receives an error. After the work item is completed, other site visitors from the assigned group or queue see no related work in the Work Guide component on the related record page.

Work Items Reassigned to a User, Group, or Queue

You can reassign open work items for a running orchestration to a different user, group, or queue. After reassignment, a work item is processed like it was after the running orchestration created it.

Work Item Statuses

Work Item Status	Description
Assigned	• The work item has been created, but the assigned user hasn't completed it, and the orchestration is in progress. • The work item has been reassigned to a different user, and the orchestration is in progress. • The assigned user hasn't completed the work item, and the orchestration was canceled. • The assigned user completed the work item, and the orchestration was canceled. • The assigned user hasn't completed the work item, and the orchestration encountered an error. • The assigned user hasn't completed the work item, and the screen flow associated with the work item's interactive step encountered an error.
Completed	• The assigned user completed the work item, and the orchestration was completed. • The assigned user hasn't completed the work item, and the stage that contains its associated step was completed. • The assigned user has completed the work item, and the orchestration was canceled. • The assigned user has completed the work item, and the orchestration encountered an error.

Work Item Status	Description
	• The assigned user has completed the work item, and the stage that contains its associated step encountered an error. • The assigned user hasn't completed the work item, and the stage that contains its associated step encountered an error.

History

In history, an orchestration work item has several possible milestones.

Work Item Milestone	Description
Start Work Item	The work item was created and assigned.
End Work Item	The assigned user completed the work item. When the work item reaches the End Work Item milestone, the Completed By field is set to the username of the user or credentialed Experience Cloud site visitor who completed the work. When the work item reaches this milestone, the Duration field is set to the duration of the step in seconds.
Reassign Work Item	The work item was reassigned.

Flow Orchestration Work Item Record Ownership

For flow orchestration work item records created in Winter '23 or later, the owner is either the assigned user or the automated process user.

For flow orchestration work items created before Summer '24 and assigned to a queue, the owner is the automated process user.

Scenario	Owner ID
Assignee is a single internal user.	ID of the assigned user
Assignee is a single credentialed Experience Cloud site visitor.	ID of the credentialed Experience Cloud site visitor
Work item is reassigned to a single internal user.	ID of the user reassigned the work item
Work item is reassigned to a single credentialed Experience Cloud site visitor.	ID of the credentialed Experience Cloud site visitor reassigned the work item
Assignee is a group.	ID of the automated process user
Work item is reassigned to a group.	ID of the automated process user
Assignee is a queue associated with the Orchestration Work Item object.	ID of the assigned queue
Work item is reassigned to a queue associated with the Orchestration Work Item object.	ID of the assigned queue

Scenario	Owner ID
Assignee is a queue not associated with the Orchestration Work Item object.	ID of the automated process user
Work item is reassigned to a queue not associated with the Orchestration Work Item object.	ID of the automated process user

SEE ALSO:

Salesforce Help: Route Work with Omni-Channel

Anatomy of an Orchestration

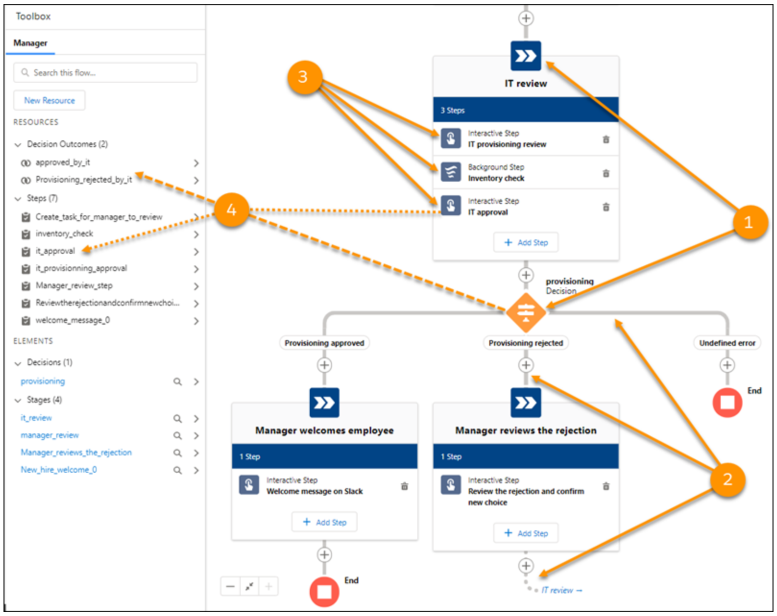
Combine elements, connectors, and resources to build orchestrations.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.



- Each element (1) represents an action that the flow can execute. Orchestrations use Stage and Decision elements.
- Each connector (2) defines an available path that the orchestration can take at run time.
- Each stage consists of one or more steps (3).

- Each resource (4) represents a value that you can reference through a stage, step, or decision.

Orchestration Types

An orchestration's type determines how the orchestration can be distributed.

All orchestrations are made up of steps grouped within a series of stages. Interactive steps contain a screen flow and require user interaction. Background steps contain an autolaunched flow and don't require user interaction. An orchestration's type affects how an orchestration is launched.

Orchestration Type	Available Distribution Methods	Notes
Autolaunched Orchestration (No Trigger)	• Custom Apex classes • Custom buttons or custom links	
Record-triggered Orchestration	A record-triggered flow orchestration only runs when a record is created or updated.	
Managed Content Authoring Workflow (Beta)	Salesforce CMS in the Digital Experiences app	Used to create an orchestration for CMS. Includes the `mContentVariantId` and `mContentSpaceId` input variables. For more details, see CMS Workflows and Approvals.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Triggers for Orchestrations

Creating or updating a record can trigger an orchestration that requires additional input from users, approval from assigned users, other updates to the record, or changes to related records. In the Start element of a record-triggered orchestration, you can specify new and changed records of a specific object. Autolaunched orchestrations don't use triggers. Use another mechanism to launch an autolaunched orchestration, such as custom Apex classes or custom URLs. Use Flow Orchestration to automate complex processes, and use Flow Trigger Explorer to order record-triggered flows.

In Flow Orchestration, the trigger occurs after a record is saved.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

What's the Difference Between a Flow and an Orchestration?

Salesforce offers several features that automate internal procedures and business processes to save time across your org.

Flow

A *flow* is an application that automates a business process by collecting data and doing something in your Salesforce org or an external system. Flows can provide screens to guide users through your business process.

Flows aren't tied to any one object, but they are record-centric. Flows can look up, create, update, and delete records for multiple objects. You can build flows with Flow Builder, a point-and-click tool.

Orchestration

An *orchestration* is an application that builds sophisticated business processes by combining and coordinating a series of flows. Orchestrations are user-centric. You can manage processes that involve different users and different parts of your organization through one orchestration. Flow Orchestration lets you monitor operations and improve efficiency.

Advanced Orchestration Concepts

After you understand the basics, you're ready for a closer look at the context in which orchestrations run and how they perform work items at the same time.

Running Context of an Orchestration

The running context determines the access that an orchestration has to Salesforce data and the context used by a paused orchestration to resume. By default, the running context of an orchestration is the Automated Process User in system context.

Orchestration Versioning

Flow Orchestration has two levels of versioning: the version of the orchestration and the version of a flow called by an orchestration.

Running Context of an Orchestration

The running context determines the access that an orchestration has to Salesforce data and the context used by a paused orchestration to resume. By default, the running context of an orchestration is the Automated Process User in system context.

The default running user for an orchestration depends on the type of orchestration and the API version that it runs in.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Autolaunched Orchestration

For API version 60.0 and later, an autolaunched orchestration always launches and resumes in the context of the Automated Process User in system context.

For API version 59.0 and earlier, an autolaunched orchestration usually launches in the context of the user who launched the orchestration. If the orchestration is launched from Apex, it runs in a system context. Control the context that an autolaunched orchestration launches and resumes in with the How to Run the Orchestration advanced option.

For API version 59.0 and earlier, the context that a paused, autolaunched orchestration resumes in depends on how it was launched or what caused it to resume.

Table 6: Resume Contexts for Autolaunched Orchestrations in API Version 59.0 and API Version 58.0

Scenario	Resumes In
Started in system context	System context
Resumed by a manually published platform event	The context of the user who published the platform event in system context
Resumed because an asynchronous background step was completed	The same context it ran in before the asynchronous background step
Resumed because a MuleSoft step was completed	The context of the Automated Process User in system context

Scenario	Resumes In
Resumed because an interactive step was completed	The context of the user who completed the interactive step in system context
Resumed because an entry or exit condition was met when a record changed	The context of the Automated Process User in system context

Record-Triggered Orchestration

For API version 60.0 and later, a record-triggered orchestration always launches and resumes in the context of the Automated Process User in system context.

For API version 59.0 and earlier, a record-triggered orchestration always launches in the context of the user who triggered the orchestration in system context.

For API version 59.0 and earlier, the context that a paused, record-triggered orchestration resumes in a user in system context. The user that the record-triggered orchestration resumes as depends on what caused it to resume.

Table 7: User Contexts of Record-Triggered Orchestrations in API Version 59.0 and Earlier

Orchestration Resumed as a Result Of	User
A manually published platform event	The user who published the platform event
An asynchronous background step was completed	The same user that the orchestration ran as before the asynchronous background step
A MuleSoft step was completed	The Automated Process User
An interactive step was completed	The user who completed the interactive step
An entry or exit condition was met because a record changed	The Automated Process User

Orchestration Versioning

Flow Orchestration has two levels of versioning: the version of the orchestration and the version of a flow called by an orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Orchestration Definition Versioning

An orchestration definition can have 1 active version at a time. The orchestration definition version used by an orchestration run is the version that's active at the time the run starts.

- If you activate a new version of an orchestration's definition after an orchestration run based on that definition starts, the orchestration run continues to run the definition version that it started in.
- Only orchestration runs that start after the new version was activated use the new active version.

Flow Definition Versioning

Interactive and background steps call flows. A step uses the definition version of the flow that's active when the step starts.

- If you activate a new definition version of a referenced flow after the orchestration run starts and the associated step run is created, the old version of the flow runs.

- If you activate a new definition version of a referenced flow after the orchestration run starts but before the associated step run is created, the new version of the flow runs.

Build an Orchestration

Use Flow Orchestration to build sophisticated business processes by combining and coordinating flows.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

It's easier to automate a business process when you understand how the pieces fit. Before you create your orchestration, talk to your stakeholders to understand the requirements. You can save draft orchestrations without knowing all the required information, but you must specify all associated flows and details before you can activate and run your orchestration.

Orchestrations are made up of Stage elements and Decision elements. Stages contain at least one step, each step calling an action to run. Background and interactive steps call flows. MuleSoft steps call MuleSoft actions imported from MuleSoft APIs. Whenever possible, create the flows and import the MuleSoft actions that you need before you build your orchestration.

1. From Setup, in the Quick Find box, enter *Flows*, select **Flows**, and then click **New Flow**.
2. Select **Start from Scratch**, and then click **Next**.
3. Select the orchestration type, and then click **Create**.
4. (Optional) To configure the Start element for a record-triggered orchestration, click **Edit**.
5. To add an element between the Start and End elements, click ⊕, and select the element.
6. To add steps to a stage, click **Add Step**.
7. To create a loop or connect to a different element, after the stage, click ⊕, click **Connect to element**, and then click + on the desired element.
8. Save your orchestration.

After you build an orchestration, activate it, and then test it to make sure that it's working as you expect. You're then ready to use it.

Use Decision Elements in an Orchestration
Control when an orchestration takes a specific decision outcome.

Define Requirements for Stages and Steps in an Orchestration
Use requirements to resume an orchestration when a record changes. Define up to three requirements to determine when a step is ready to start or when to mark an interactive step or stage complete.

Assign an Interactive Step in an Orchestration
When you create an interactive step, you assign it to a user, group, or queue. A user can be an internal user or a credentialed Experience Cloud site visitor. Groups or queues can include internal users, credentialed Aura site visitors, or credentialed LWR site visitors. You can also assign an interactive step to a resource that contains a username, group API name, or queue API name when the orchestration runs. When the active screen flow associated with the interactive step runs, an assigned user completes the flow on the related context record.

Route Orchestration Work Items with Omni-Channel

To use Omni-Channel routing in Service Cloud with orchestration work items, you must have at least one queue associated with the Orchestration Work Item. When you assign an interactive step to that queue, members of the queue receive notifications via the Omni-Channel widget based on your routing logic. Unless you disable default email notifications for work items, queue members also receive email notifications.

Redirect an Orchestration Path

Flow Orchestration uses Auto-Layout in Flow Builder. In Auto-Layout, elements on the canvas are spaced and connected automatically. Use Go To connectors when you have elements that don't follow the usual consecutive auto-layout path.

Add an End Element to an Orchestration Path

All elements in an orchestration are connected automatically or connected by Go To connectors that you add manually. To finish a path in your orchestration, add an End element.

Use Automatic Output in Orchestrations

An orchestration has access to output for its stages, steps, and decisions. Query the status of any stage or step in the orchestration. Use output parameters from any step's associated flow. In an orchestration configured to run on API version 58.0 and later, referenced automatic outputs that contain a record or a record collection are refreshed with their latest values each time the orchestration run resumes.

Trigger an Evaluation of Orchestration Stage and Step Conditions

Every time a step within the current stage completes, the orchestration evaluates the conditions for that stage and its steps. You can also publish an orchestration event from a flow to trigger an evaluation of orchestration stage and step conditions.

Integrate an Orchestration with External Systems

Add a MuleSoft step to your orchestration to call an imported MuleSoft action. You can also use the `$Orchestration.Instance` system variable to integrate external systems with your orchestration.

Create an Orchestration Template

You can save a new or existing orchestration as a template, and then use it as a starting point for creating other orchestrations in Flow Builder. You can also distribute the template via a managed package so that subscribers can create orchestrations based on the template.

Make Work Accessible to Assigned Users

When an orchestration runs an interactive step, it emails a notification to the assigned user, group, or queue. Credentialed Experience Cloud site visitors can see and access their assigned Flow Orchestration work items on the Orchestration Work Item List object page. Internal users and credentialed Experience Cloud site visitors complete their assigned work in the Work Guide.

Use Decision Elements in an Orchestration

Control when an orchestration takes a specific decision outcome.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

Before you begin, add the Decision element to your orchestration.

1. Set up the conditions.

   At run time, the conditions are evaluated in the order you specify.

Option	Behavior for Decision Outcomes
`All Conditions Are Met`	If one of the conditions is false, the orchestration evaluates the next outcome's conditions.
`Any Condition Is Met`	If one of the conditions is true, the orchestration immediately takes this outcome's path.
`Custom Condition Logic Is Met`	When you select this option, provide the condition logic by entering up to 1000 characters. Use: - Numbers to refer to each condition - *`AND`*, *`OR`*, or *`NOT`* to identify which combination of conditions must be true - Parentheses to group parts of the string together If you enter *`AND`*, it's the same as if you selected **All Conditions Are Met**. If you enter *`OR`*, it's the same as if you selected **Any Condition Is Met**. If you enter any other logic, make sure that you include a number for each condition. For example, for *`1 AND NOT(2 OR 3)`*, the flow evaluates whether the first condition is `true` and neither the second nor third condition is `true`.

2. Identify the logic between the conditions.

Column Header	Description
Resource	Options: - Select an input variable or automatic output from a stage or step. - Select a Decision element. - Select a global variable.
Operator	The available operators depend on the data type selected for `Resource`. See Flow Orchestration Operators in Decision Elements.
`Value`	**Resource** and **Value** in the same row must have compatible data types. Options: - Select an orchestration resource, such as an input variable or automatic output from a stage or step. - Select a global variable. - Manually enter a literal value. When you add or subtract a number from a date value, the date adjusts in days, not hours.

Define Requirements for Stages and Steps in an Orchestration

Use requirements to resume an orchestration when a record changes. Define up to three requirements to determine when a step is ready to start or when to mark an interactive step or stage complete.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

Before you begin, add a Stage element to your orchestration or a Step resource to a stage.

1. In the Properties panel, select the condition that allows you to create up to three requirements to start a step or complete a stage or interactive step.

To Use Requirements to	Select
Start a step	**When the specified requirements are met, the step starts**
Complete a step	**When the specified requirements are met, the step is marked Completed**
Complete a stage	**When the specified requirements are met, the stage is marked Completed**

2. Set up the logic for the requirements.

Option	Behavior for Requirements
`All Requirements Are Met`	If one of the requirements is false, then the requirements aren't met.
`Any Requirement Is Met`	If one of the requirements is true, then the requirements are met.
`Custom Requirement Logic Is Met`	When you select this option, provide the requirement logic by entering up to 1000 characters. Use: - Numbers to refer to each requirement - *`AND`*, *`OR`*, or *`NOT`* to identify which combination of requirements must be true - Parentheses to group parts of the string together If you enter *`AND`*, it's the same as if you selected **All Requirements Are Met**. If you enter *`OR`*, it's the same as if you selected **Any Requirement Is Met**. If you enter any other logic, make sure that you include a number for each requirement. For example, for *`1 AND NOT(2 OR 3)`*, the flow evaluates whether the first requirement is `true` and neither the second nor third condition is `true`.

3. Define up to three requirements.

   A change to a record referenced in a requirement can trigger the orchestration to evaluate the status of the current stage and the outstanding steps within it. Some requirement resources don't trigger condition evaluations.

Column Header	Description
Resource	Options: • Select an orchestration resource – Select a variable – Select a record variable field – Select automatic output from a step. • Select a Stage element's status • Select a Step resource's status. • For record-triggered orchestrations, select the $Record global variable. • Select a global variable.
Operator	The available operators depend on the data type selected for **Resource** and work the same as operators used for Decision elements. See Flow Orchestration Operators in Decision Elements.
Value	**Resource** and **Value** in the same row must have compatible data types. Options: • Select an orchestration resource, such as a variable or automatic output from a step. • Select a global constant • Select a global variable. • Manually enter a literal value. When you add or subtract a number from a date value, the date adjusts in days, not hours.

SEE ALSO:

Considerations for Orchestrations

Object Reference for the Salesforce Platform : StandardObjectNameChangeEvent

Assign an Interactive Step in an Orchestration

When you create an interactive step, you assign it to a user, group, or queue. A user can be an internal user or a credentialed Experience Cloud site visitor. Groups or queues can include internal users, credentialed Aura site visitors, or credentialed LWR site visitors. You can also assign an interactive step to a resource that contains a username, group API name, or queue API name when the orchestration runs. When the active screen flow associated with the interactive step runs, an assigned user completes the flow on the related context record.

The User field for an interactive step's assigned user includes internal users and credentialed Experience Cloud site visitors. Whenever you assign an interactive step to a user or a credentialed site visitor, ensure that they have the required access to the related record.

For an internal user to complete an interactive step, they must have access to the associated internal Salesforce Lightning record page. For a credentialed Experience Cloud site visitor to complete an interactive step, they must have access to the associated related record page in an Aura or LWR site.

To use Omni-Channel routing with Flow Orchestration, set up Omni-Channel and associate at least one queue with the Orchestration Work Item object. Then, to notify assigned users with the Omni-Channel widget based on your defined routing logic, assign an interactive step to a queue that's associated with the Orchestration Work Item object.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To open, edit, or create an orchestration in Flow Builder:

- Manage Flow

To complete assigned work and resume a paused orchestration

- Run Flows

1. Add an interactive step to a stage in your orchestration.
2. In the Properties panel for the interactive step, under Select Someone to Complete the Action, select an assignment type.
   - To specify a user, select **User**.
   - To specify a regular public group, select **Group**.
   - To specify a group that's a queue, select **Queue**.
   - To specify a resource that contains a user's username when the orchestration runs, select **User Resource**.
   - To specify a resource that contains a group's API name when the orchestration runs, select **Group Resource**.
   - To specify a resource that contains a queue's API name when the orchestration runs, select **Queue Resource**.
3. Specify the assigned user, group, or queue.
   - If you selected User, search for the name of an internal user or a credentialed Experience Cloud site visitor, and select it from the list.
   - If you selected Group, search for a group's label, and select it from the list.
   - If you selected Queue, search for a queue's label, and select it from the list.
   - If you selected User Resource, specify the API name of the variable that contains the assignee's username when the orchestration runs.

   Important: Don't select $User for User Resource. The $User global variable evaluates to the system user when the orchestration is running in system context and an interactive step can't be assigned to the system user.

   - If you selected Group Resource, specify the API name of the variable that contains the group API name when the orchestration runs.

- If you selected Queue Resource, specify the API name of the variable that contains the assigned queue's API name when the orchestration runs.

SEE ALSO:

Flow Orchestration Interactive Steps

Running Context of an Orchestration

Salesforce Help: Route Work with Omni-Channel

Route Orchestration Work Items with Omni-Channel

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To open, edit, or create an orchestration in Flow Builder:

- Manage Flow

To complete assigned work and resume a paused orchestration

- Run Flows

To use Omni-Channel routing in Service Cloud with orchestration work items, you must have at least one queue associated with the Orchestration Work Item. When you assign an interactive step to that queue, members of the queue receive notifications via the Omni-Channel widget based on your routing logic. Unless you disable default email notifications for work items, queue members also receive email notifications.

When you assign an interactive step to a group or queue, each group or queue member receives an email notification by default. The email notification contains a link to the record where one of the members can complete the assigned work.

When a queue associated with the Orchestration Work Item object is assigned to an interactive step, the work item owner is the queue.

1. Set up Omni-Channel.
2. Associate a queue with the Orchestration Work Item object.
3. Assign an interactive step to a queue associated with the Orchestration Work Item object.

SEE ALSO:

Salesforce Help: Route Work with Omni-Channel

Redirect an Orchestration Path

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Flow Orchestration uses Auto-Layout in Flow Builder. In Auto-Layout, elements on the canvas are spaced and connected automatically. Use Go To connectors when you have elements that don't follow the usual consecutive auto-layout path.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

To add a Go To connector, you must have at least two elements in your orchestration.

1. Directly after the element that you want to change the connector for, click ⊕.
2. Click **Connect to element**.

3. Click + on the element that you want to connect to.

The original element now has a dotted line connection to the specified element.

Add an End Element to an Orchestration Path

All elements in an orchestration are connected automatically or connected by Go To connectors that you add manually. To finish a path in your orchestration, add an End element.

To add an End element, you must have at least one Decision element and two paths in your orchestration.

1. After the last element in the path where you want to add the End element, click ⊕.
2. Select **End**.

The path now ends execution when this element is reached.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Use Automatic Output in Orchestrations

An orchestration has access to output for its stages, steps, and decisions. Query the status of any stage or step in the orchestration. Use output parameters from any step's associated flow. In an orchestration configured to run on API version 58.0 and later, referenced automatic outputs that contain a record or a record collection are refreshed with their latest values each time the orchestration run resumes.

User Permissions Needed

To open, edit, or create an orchestration in Flow Builder:	Manage Flow

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Add an element or resource to your orchestration.

When you build an orchestration, automatic outputs for every stage and step in that orchestration are universally available. This universal availability means that you can potentially use automatic output in your designed orchestration before it's available in an orchestration run. So, when using automatic output, consider the order in which an orchestration executes its elements and resources.

1. In a resource, value, or input parameter field, select a stage or step from the dropdown list.
2. Select the automatic output field from the dropdown list.
3. Save your work.

Trigger an Evaluation of Orchestration Stage and Step Conditions

Every time a step within the current stage completes, the orchestration evaluates the conditions for that stage and its steps. You can also publish an orchestration event from a flow to trigger an evaluation of orchestration stage and step conditions.

SEE ALSO:

Publish an Orchestration Event

Integrate an Orchestration with External Systems

Add a MuleSoft step to your orchestration to call an imported MuleSoft action. You can also use the `$Orchestration.Instance` system variable to integrate external systems with your orchestration.

Publish an Orchestration Event

To allow an external system to make a paused orchestration evaluate its stage and step conditions, publish an orchestration event from a record-triggered orchestration.

SEE ALSO:

Flow Orchestration MuleSoft Steps

Publish an Orchestration Event

Publish an Orchestration Event

To allow an external system to make a paused orchestration evaluate its stage and step conditions, publish an orchestration event from a record-triggered orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

User Permissions Needed	
To open, edit, or create an orchestration in Flow Builder:	Manage Flow

Add a custom field to the object to hold an orchestration run ID.

Create an autolaunched flow with an input variable that accepts an orchestration run ID and passes it to the action that invokes an external system.

Add logic at the end of the action invoking the external system. After the external system finishes its task, it must update the custom orchestration run ID field on the affected record with the orchestration run ID it was passed.

Call the autolaunched flow from an asynchronous background step in an orchestration, and pass $Orchestration.Instance to the appropriate input parameter.

1. Create a record-triggered flow that runs when the custom orchestrator run ID field is updated on a record. If you have records of more than one object affected by an external system, create a record-triggered flow for each object.
2. Add a Create Records element to the record-triggered flow.
3. Enter a label, API name, and description for the element.
4. Select **Use separate resources, and literal values.**
5. For Object, search for and select *`Orchestration Event`*.
6. For Field, enter *`Orchestration`*, and then select **OrchestrationInstanceId**.
7. For Value, enter *`$Record`*, and then select **$Record**. Then select the name of the custom orchestration run ID field on the triggering record.
8. Click **Done**.

9. Save and activate the new record-triggered flow.

SEE ALSO:

Extend Salesforce with Clicks, Not Code : Create Custom Fields

Automate Your Business Processes : Flow Resource: Variable

Create an Orchestration Template

You can save a new or existing orchestration as a template, and then use it as a starting point for creating other orchestrations in Flow Builder. You can also distribute the template via a managed package so that subscribers can create orchestrations based on the template.

User Permissions Needed

To open, edit, or create an orchestration in Flow Builder:	Manage Flow

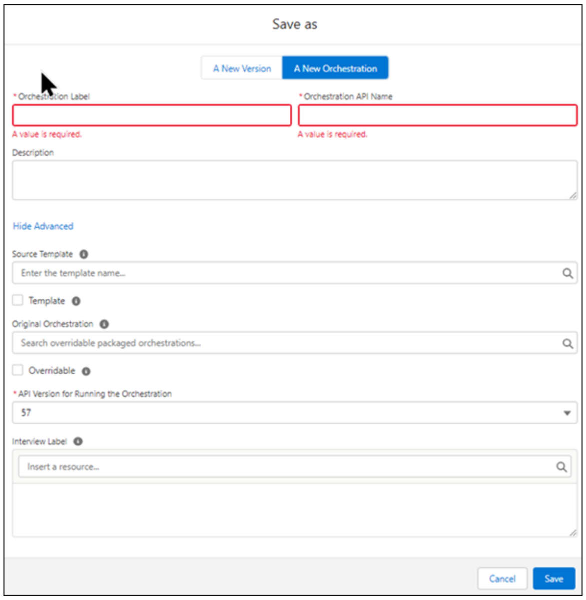
EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

1. To create an orchestration template from an orchestration:
   a. Open an orchestration and click **Save As**.
      The Save as dialog opens



* API Version for Running the Orchestration: 57

.

   b. Click **A New Orchestration** and enter a label, API name, and description for your orchestration template.

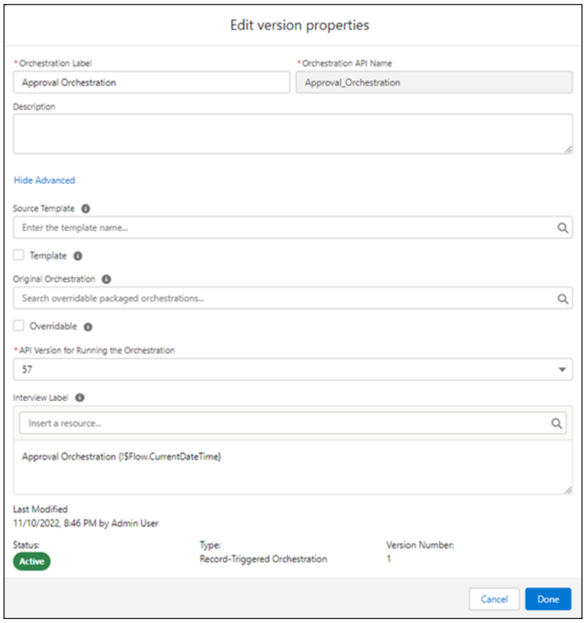
      The description appears under the orchestration template's name in the New Flow dialog and gives users information about what your template does.

   c. Click **Show Advanced**.

   d. Select **Template** and click **Done**.

2. To make an orchestration into a template.

   a. Open an orchestration and click ⚙.
      The Edit version properties dialog opens.



Edit version properties

* Orchestration Label: Approval Orchestration

* Orchestration API Name: Approval_Orchestration

Description:

Source Template: Enter the template name...

□ Template

Original Orchestration: Search overridable packaged orchestrations...

□ Overridable

* API Version for Running the Orchestration: 57

Interview Label: Insert a resource...

Approval Orchestration {!$Flow.CurrentDateTime}

Last Modified
11/10/2022, 8:46 PM by Admin User

Status:	Type:	Version Number:
Active	Record-Triggered Orchestration	1

   b. Ensure that the orchestration has a description.

      The description appears under the orchestration template's name in the New Flow dialog and gives users information about what your template does.

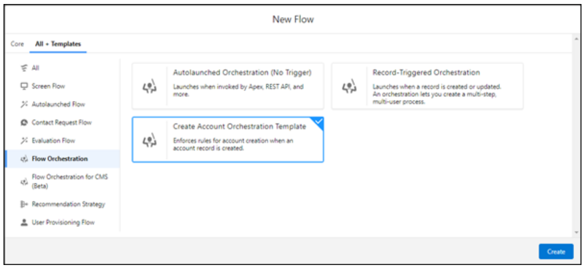
   c. Click **Show Advanced**.

   d. Select **Template** and click **Done**.

3. To use the new template, select it in the New Flow dialog. .

   a. In Setup, from the Flows listview, click **New Flow**.

   b. In the New Flow dialog, click **All + Templates**, and then click **Flow Orchestration**.
      The new orchestration template is shown in the Flow Orchestration category on the All + Templates tab of the New Flow dialog.



New Flow

Autolaunched Orchestration (No Trigger)
Launches when invoked by Apex, REST API, and more.

Record-Triggered Orchestration
Launches when a record is created or updated. An orchestration lets you create a multi-step, multi-user process.

Create Account Orchestration Template
Enforces rules for account creation when an account record is created.

   c. Select the new template and click **Done**.

Make Work Accessible to Assigned Users

When an orchestration runs an interactive step, it emails a notification to the assigned user, group, or queue. Credentialed Experience Cloud site visitors can see and access their assigned Flow Orchestration work items on the Orchestration Work Item List object page. Internal users and credentialed Experience Cloud site visitors complete their assigned work in the Work Guide.

Add an Orchestration Work Item List Object Page to an Experience Cloud Site

Internal users can see and access their assigned work in the Flow Orchestration Work Items list view. Add the Orchestration Work Item List object page to your Aura or LWR site so that credentialed site visitors can see and access their assigned Flow Orchestration work items.

Add the Work Guide to a Record Page Layout

Add the Flow Orchestration Work Guide Lightning App Builder component to the page layouts for record types referenced by interactive steps.

Add the Work Guide to an Experience Cloud Site

Add the Flow Orchestration Work Guide component to the related record page in your Aura and LWR sites for record types referenced by interactive steps.

SEE ALSO:

Flow Orchestration Work Items

Add an Orchestration Work Item List Object Page to an Experience Cloud Site

Internal users can see and access their assigned work in the Flow Orchestration Work Items list view. Add the Orchestration Work Item List object page to your Aura or LWR site so that credentialed site visitors can see and access their assigned Flow Orchestration work items.

1. In Experience Builder, select **Pages** > **New Page**.
2. Select **Object Pages**.
3. In the New Object Pages dialog box, enter *`work item`* in the Search box.
4. Select **Orchestration Work Item**, and click **Create**.
5. In the dialog box, click **Create**.
6. Select **Pages** > **Orchestration Work Item List**.
7. Preview and publish your site.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To create an Experience Cloud site:

- Create and Set Up Experiences AND View Setup and Configuration

To customize an Experience Cloud site:

- Be a member of the site AND Create and Set Up Experiences
- OR
- Be a member of the site AND an experience admin, publisher, or builder in that site

To publish an Experience Cloud site:

- Be a member of the site AND Create and Set Up Experiences
- OR
- Be a member of the site AND an experience admin or publisher in that site

To run a flow in an Experience Builder site:

- Run Flows

Add the Work Guide to a Record Page Layout

Add the Flow Orchestration Work Guide Lightning App Builder component to the page layouts for record types referenced by interactive steps.

1. To add the component to an existing page layout, on a page for a record type associated with an interactive step, click ⚙, and then select **Edit Page**.
2. To create a page layout for a record type associated with an interactive step, from Setup, in the Quick Find box, enter `App Builder`, and then select **Lightning App Builder**.
   a. Click **New**
   b. Select **Record Page**, and then click **Next**.
   c. Give your record page a label, and then click **Next**.

      The label can be up to 80 characters.
   d. Select a page template, and click **Finish**.
3. Under Components, drag **Flow Orchestration Work Guide** onto the page layout.

   If this page layout is new, add other components as needed.
4. Save your work.
5. If the page layout isn't already activated, the Page Saved window appears and asks if you want to activate the page.

Activate your orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To create and save Lightning pages in the Lightning App Builder:

- Customize Application

Add the Work Guide to an Experience Cloud Site

Add the Flow Orchestration Work Guide component to the related record page in your Aura and LWR sites for record types referenced by interactive steps.

Your org must have Flow Orchestration enabled.

1. In Experience Builder, navigate to the related record page.
2. From the Components panel, drag **Flow Orchestration Work Guide** onto the page.
3. Save your work.

Add the Orchestration Work Item Object List page to your Aura or LWR site and ensure that the site is published. Then activate the orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To create an Experience Cloud site:

- Create and Set Up Experiences AND View Setup and Configuration

To customize an Experience Cloud site:

- Be a member of the site AND Create and Set Up Experiences

  OR

- Be a member of the site AND View Setup and Configuration AND an experience admin, publisher, or builder in that site

To publish an Experience Cloud site:

- Be a member of the site AND Create and Set Up Experiences

  OR

- Be a member of the site AND an experience admin or publisher in that site

Deploy an Orchestration

After you design and test your orchestration, it’s time to put it to work!

Set Up an Org-Wide Email Address

To receive emails from Flow Orchestration, create an org-wide email address.

Activate or Deactivate an Orchestration

You can have multiple versions of an orchestration in Salesforce, but only one version of each orchestration can be active at a time. You can activate or deactivate an orchestration in Flow Builder or from the orchestration's detail page in Setup.

Deploy Orchestrations with Change Sets

Create, test, and debug your orchestrations in a sandbox. Use a change set to send the orchestration and its associated flows to production when you're ready to deploy.

Set Up an Org-Wide Email Address

To receive emails from Flow Orchestration, create an org-wide email address.

The email address you set up in this step acts as the From address in your emails from Flow Orchestration. If you don't have a From address, your notification emails don't send.

Note: If you have an existing org-wide email address, you don't have to set up a new one, but make sure you've specified it as your Email Approval Sender in Process Automation Settings in Setup.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

1. From Setup, in the Quick Find box, enter `Email`, and select **Organization-Wide Address**.
2. Select **Add**.
3. Fill in the Organization-wide address form.
    a. For **Display Name**, enter a name that labels your org-wide address.
    b. For **Email Address**, enter a company email address that can be used as the **From Address** in your email alert.
    c. Select **Allow All Profiles to Use this From Address**.
    d. Save your work.
4. View your org-wide address and the status, which reads **Verification Request Sent**.
5. Navigate to the email address you specified in the Email Address field.
6. When Salesforce sends an email to the company address you entered previously, approve and verify the company email address.
7. Navigate back to Salesforce, and check to make sure that the status of your address is **Verified**.
8. From Setup, in the Quick Find box, enter `automation settings`, and then select **Process Automation Settings.**.
9. For **Email Approval Sender**, specify your org-wide email address.
10. Save your changes.

Activate your orchestration.

Important: If the Sender Type is OrgWideEmailAddress, ensure that the user running the flow has the proper profile configurations required by the specific org-wide email address being used. Proceeding without the proper configuration results in an error.

Activate or Deactivate an Orchestration

You can have multiple versions of an orchestration in Salesforce, but only one version of each orchestration can be active at a time. You can activate or deactivate an orchestration in Flow Builder or from the orchestration's detail page in Setup.

When you activate an orchestration version, the previously activated version, if one exists, is deactivated. Any running orchestration continues to run using the version that it started with.

1. In Flow Builder, open the orchestration version.
2. On the button bar, click **Activate** or **Deactivate**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To activate or deactivate an orchestration:

- Manage Flow

To activate a record-triggered orchestration:

- View All Data

Deploy Orchestrations with Change Sets

Create, test, and debug your orchestrations in a sandbox. Use a change set to send the orchestration and its associated flows to production when you're ready to deploy.

User Permissions Needed	
To create, edit, or view processes:	Manage Flow AND View All Data
To edit deployment connections:	Deploy Change Sets AND Modify Metadata Through Metadata API Functions
To use outbound change sets:	Create and Upload Change Sets
To use inbound change sets:	Deploy Change Sets AND Modify Metadata Through Metadata API Functions

Create and upload the outbound change set in your sandbox, and deploy the inbound change set in production.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

1. Ensure that all group names and queue names used in the source org to assign interactive steps to users duplicate the names used in the target org.
2. Ensure that no interactive steps are directly assigned to a specific user.
   a. Create constants for each assigned user who's directly assigned to an interactive step in the orchestration.
   b. Assign each interactive step to the appropriate assigned-user constant.
3. Activate your orchestration and all its referenced flows.
4. Create an outbound change set.
5. Add components to the new change set. These components include the orchestration, its associated flows, and any new custom actions or new custom flow screen components that the associated flows depend on.
6. Upload your outbound change set.
7. Deploy your inbound change set in your target org.
8. Update any assigned-user constants in the orchestration, and save a new version of the orchestration.
9. Activate the new version of the orchestration.

Ensure that the page layouts for each context record referenced in the orchestration include the Orchestrator Work Guide Lightning App Builder component.

SEE ALSO:

Sandboxes: Staging Environments for Customizing and Testing : Change Sets

Orchestration Run

An orchestration run is created for each instance of an orchestration.

An *orchestration* is an application built by your admin that uses stages, steps, and decisions to organize a complex business process.An orchestration *run* is a running instance of an orchestration. The context an orchestration run uses depends on the orchestration type. You can also specify a context with the How to Run the Orchestration advanced option.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Resuming a Failed Orchestration

If an orchestration run fails because of an error in an action called by one of its steps, you have up to 14 days to fix the error in the action and resume the orchestration. If the orchestration run failed because of some other type of error, it can't be resumed. If the orchestration run failed but wasn't resumed within 14 days, it can no longer be resumed.

Item	Status In Failed Run	When Run is Resumed	Status In Resumed Run
Stage	Error	The stage is resumed.	In Progress
Background or MuleSoft Step	Error	The step is restarted.	In Progress
Background or MuleSoft Step	Discontinued (no pending outputs)	The step is restarted.	In Progress

Item	Status In Failed Run	When Run is Resumed	Status In Resumed Run
Background or MuleSoft Step	Discontinued (pending outputs)	The stored outputs are processed, and the step isn't restarted.	Completed Error
Interactive Step	Error Discontinued	A new work item is created for the step.	In Progress

Statuses and Milestones

After it's created, an orchestration run has an associated status.

Orchestration Run Status	Description
In Progress	The orchestration started.
Completed	The orchestration was completed.
Suspended	The orchestration was manually suspended.
Canceled	The orchestration was manually canceled.
Error	• The orchestration encountered an error. • A stage within the orchestration encountered an error. • A step within the orchestration encountered an error. • The screen flow or autolaunched flow associated with a step within the orchestration encountered an error.

In logging, an orchestration run has several milestones.

Orchestration Run Milestone	Description
Start Run	The orchestration started or was manually resumed.
End Run	The orchestration was completed.
Suspend Run	The orchestration was manually suspended.
Cancel Run	The orchestration was manually canceled.
Resume Run	The orchestration was manually resumed.
Fail Run	• The orchestration encountered an error. • A stage within the orchestration encountered an error.

Orchestration Run Milestone	Description
	• A step within the orchestration encountered an error.

SEE ALSO:

Running Context of an Orchestration

Manage Orchestrations and Work Items

Manage orchestrations and work items with list views. Cancel or suspend a running orchestration. Resume an orchestration run that failed within the previous 14 days because of an error in an action or flow called by a step. Or resume an orchestration run that was manually suspended. Reassign work items that have been assigned, but not completed.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

View All Orchestration Work Items

Use the All Work Items list view to see all work items. Use the All Open Orchestration Work Items list view to see all assigned but not completed work items. Assigned users can see and access only their pending work items in the All Open Orchestration Work Items list view.

View Orchestration Work Items on a Record

To see your assigned work items for a specific record, go to the associated record page. If you have multiple work items assigned to you for that record, you can see them all in the Orchestration Work Guide component. You can sort the work item list by last-modified date or select the item you want to complete first. When you complete a work item, the work item list refreshes automatically.

View Orchestration Runs

Use the Orchestration Runs list view to see all in-progress, canceled, and completed orchestration runs in your org. Access orchestration details and history through the orchestration runs list view.

Reassign an Orchestration Work Item

Reassign an assigned work item to a different user, group, or queue.

Disable Default Email Notifications for Work Item Assignments

By default, an orchestration sends an email notification when an orchestration work item is assigned or reassigned to a user, group, or queue. Disable default work item notifications to stop sending emails to internal users and credentialed Experience Cloud site visitors.

Suspend an In-Progress Orchestration

Suspend an orchestration run to wait until you're ready to continue. When you suspend a running orchestration, the current stage is also suspended. In-progress steps continue to run, but no new steps are started. If an in-progress step has output, it's stored so it can be processed when the orchestration is resumed.

Resume a Suspended Orchestration

Resume a suspended orchestration run to continue its processing. When a suspended orchestration run is resumed, the suspended stage is also resumed. When the orchestration run is resumed, it evaluates the status of in-progress steps and updates the step status where appropriate. Stored outputs from steps that were in progress when the orchestration run was suspended are processed.

Resume a Failed Orchestration

When an orchestration run failed within the previous 14 days because of an error in an action called by a step, you can fix the error and resume the orchestration.

Cancel a Running Orchestration

Cancel an in-progress orchestration from the orchestration runs list view.

Use Orchestration Reports

Use sample flow orchestration reports to track orchestration usage. Sample reports include Orchestration Runs, Orchestration Stage Runs, Orchestration Step Runs, Orchestration Work Items, and Orchestration Run Logs. These sample reports are based on the Orchestration Runs Spring '24, Orchestration Stage Runs Spring '24, Orchestration Step Runs Spring '24, Orchestration Work Items Spring '24, and Orchestration Run Logs Spring '24 custom report types.

Orchestration Statuses and Milestones

Each part of an orchestration has a status assigned when an orchestration runs.

View All Orchestration Work Items

Use the All Work Items list view to see all work items. Use the All Open Orchestration Work Items list view to see all assigned but not completed work items. Assigned users can see and access only their pending work items in the All Open Orchestration Work Items list view.

1. In the App Launcher, find and select **Orchestration Work Items**.
2. To see assigned and completed orchestration work items, from the dropdown list, select **All Work Items**.
3. To see assigned orchestration work items, from the dropdown list, select **All Open Work Items**.
4. To see an assigned work item on its associated record page, click the assigned work item record in the list view.

Note: Only the assigned user or a member of the assigned group or queue can see an assigned work item on its associated record page.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To view all orchestration work items:

- View access is based on sharing settings

View Orchestration Work Items on a Record

To see your assigned work items for a specific record, go to the associated record page. If you have multiple work items assigned to you for that record, you can see them all in the Orchestration Work Guide component. You can sort the work item list by last-modified date or select the item you want to complete first. When you complete a work item, the work item list refreshes automatically.

Assigned orchestration work items are shown in the Work Guide on the record page of their associated record.

1. Go to a record that you have assigned work items for.
2. To sort orchestration work items in the Work Guide, click ⇅ and then select how you want to sort your assigned work.
3. To filter displayed orchestration work items in the Work Guide, click 🔍, and enter the term to search for.
   The Word Guide lists only those work items with labels that include the specified search term.
4. To complete a work item:
   a. In the Work Guide, click > for the item you want to complete.
      The screen flow opens in the Work Guide.
   b. After you've finished the screen flow, click **Finish**.
      The work item status is set to Completed, and you're returned to the refreshed list of work items in the Work Guide.
   c. If you select an item that you don't want to complete, click ←, and then click **OK**.
      You return to the list of work items in the Work Guide.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To run a flow:

- Run Flows

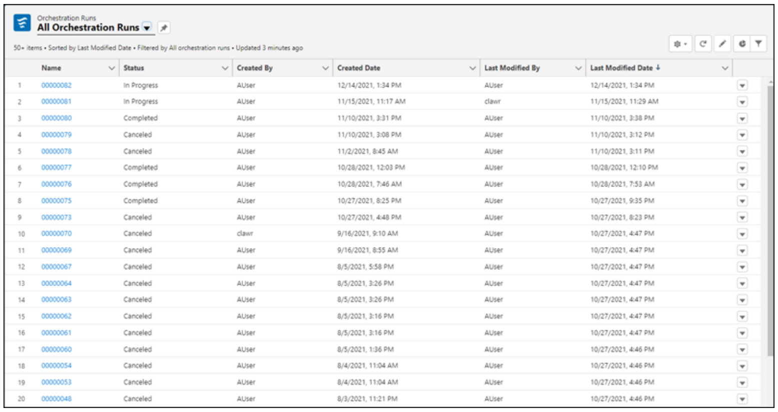
SEE ALSO:

Make Work Accessible to Assigned Users

View Orchestration Runs

Use the Orchestration Runs list view to see all in-progress, canceled, and completed orchestration runs in your org. Access orchestration details and history through the orchestration runs list view.

1. In the App Launcher, find and select **Orchestration Runs**.



Orchestration Runs
All Orchestration Runs

50+ items • Sorted by Last Modified Date • Filtered by All orchestration runs • Updated 3 minutes ago

	Name	Status	Created By	Created Date	Last Modified By	Last Modified Date ↓
1	00000082	In Progress	AUser	12/14/2021, 1:34 PM	AUser	12/14/2021, 1:34 PM
2	00000081	In Progress	AUser	11/15/2021, 11:17 AM	clawr	11/15/2021, 11:29 AM
3	00000080	Completed	AUser	11/10/2021, 3:31 PM	AUser	11/10/2021, 3:38 PM
4	00000079	Canceled	AUser	11/10/2021, 3:08 PM	AUser	11/10/2021, 3:12 PM
5	00000078	Canceled	AUser	11/2/2021, 8:45 AM	AUser	11/10/2021, 3:11 PM
6	00000077	Completed	AUser	10/28/2021, 12:03 PM	AUser	10/28/2021, 12:10 PM
7	00000076	Completed	AUser	10/28/2021, 7:46 AM	AUser	10/28/2021, 7:53 AM
8	00000075	Completed	AUser	10/27/2021, 8:25 PM	AUser	10/27/2021, 9:35 PM
9	00000073	Canceled	AUser	10/27/2021, 4:48 PM	AUser	10/27/2021, 8:23 PM
10	00000070	Canceled	clawr	9/16/2021, 9:10 AM	AUser	10/27/2021, 4:47 PM
11	00000069	Canceled	AUser	9/16/2021, 8:55 AM	AUser	10/27/2021, 4:47 PM
12	00000067	Canceled	AUser	8/5/2021, 5:58 PM	AUser	10/27/2021, 4:47 PM
13	00000064	Canceled	AUser	8/5/2021, 3:26 PM	AUser	10/27/2021, 4:47 PM
14	00000063	Canceled	AUser	8/5/2021, 3:26 PM	AUser	10/27/2021, 4:47 PM
15	00000062	Canceled	AUser	8/5/2021, 3:16 PM	AUser	10/27/2021, 4:47 PM
16	00000061	Canceled	AUser	8/5/2021, 3:16 PM	AUser	10/27/2021, 4:47 PM
17	00000060	Canceled	AUser	8/5/2021, 1:36 PM	AUser	10/27/2021, 4:46 PM
18	00000054	Canceled	AUser	8/4/2021, 11:04 AM	AUser	10/27/2021, 4:46 PM
19	00000053	Canceled	AUser	8/4/2021, 11:04 AM	AUser	10/27/2021, 4:46 PM
20	00000048	Canceled	AUser	8/3/2021, 11:21 PM	AUser	10/27/2021, 4:46 PM

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

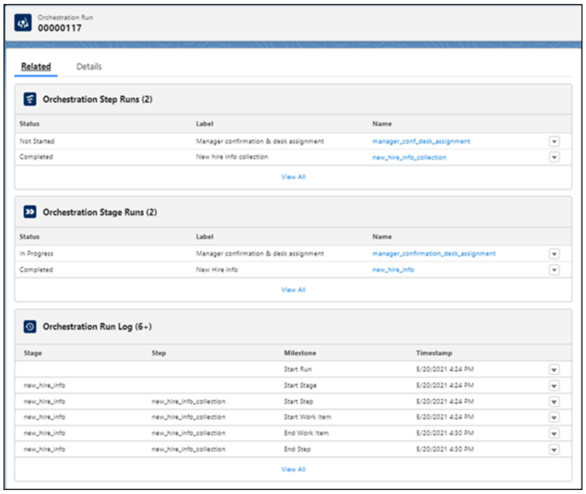
This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To view orchestration runs:

- View access is based on sharing settings

2. To see details for a specific orchestration run, on the All Orchestration Runs list view, click the link for an orchestration, and then click the **Related** tab.



Orchestration Run
00000117

Orchestration Step Runs (2)

Status	Label	Name
Not Started	Manager confirmation & desk assignment	manager_conf_desk_assignment
Completed	New hire info collection	new_hire_info_collection

Orchestration Stage Runs (2)

Status	Label	Name
In Progress	Manager confirmation & desk assignment	manager_confirmation_desk_assignment
Completed	New Hire info	new_hire_info

Orchestration Run Log (6+)

Stage	Step	Milestone	Timestamp
		Start Run	5/20/2021 4:24 PM
new_hire_info		Start Stage	5/20/2021 4:24 PM
new_hire_info	new_hire_info_collection	Start Step	5/20/2021 4:24 PM
new_hire_info	new_hire_info_collection	Start Work Item	5/20/2021 4:24 PM
new_hire_info	new_hire_info_collection	End Work Item	5/20/2021 4:30 PM
new_hire_info	new_hire_info_collection	End Step	5/20/2021 4:30 PM

3. To see the full orchestration run history, under **Orchestration Run Log**, click **View All**.

Add Comments to the Orchestration Run Log

Add custom comments to the Orchestration Run Log using variables in flows called by orchestration steps.

Add a Comments Column to the Orchestration Run Log

Add comments from flows called by orchestration steps to the Orchestration Run Log to customize log information.

Add Comments to the Orchestration Run Log

Add custom comments to the Orchestration Run Log using variables in flows called by orchestration steps.

1. In a flow called by an orchestration step, add a variable named Comments.
   - a. For Resource Type, select **Variable**.
   - b. For API name, enter `Comments`.
   - c. For Description, enter `Stores custom text to be added to the Comments field in the Flow Orchestration Log`.
   - d. Select **Available for output**.
   - e. For Data Type, select **Text**.
2. In an Assignment element in your flow, set the `Comments` variable to a string.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Add a Comments Column to the Orchestration Run Log

Add comments from flows called by orchestration steps to the Orchestration Run Log to customize log information.

1. From the Orchestration Run List View, click ⚙ and select **Edit Object**.
2. In the Orchestration Run setup page, click **Page Layouts**, and select **Orchestration Instance Layout**.
3. In the Related Lists section, click the 🔧 for Orchestration Run Log.
4. In the Related List Properties - Orchestration Run Log window, under Available Fields, select **Comments** and click ▶.
   The Comments field is added to the Selected Fields list.
5. To change the Comments field's location in the Orchestration Run Log, use the up and down arrows.
6. Click **OK**, and then click **Save**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Reassign an Orchestration Work Item

Reassign an assigned work item to a different user, group, or queue.

You can reassign an assigned work item for an orchestration that's still in progress.

1. In the App Launcher, find and select **Orchestration Work Items**.
2. On the All Open Work Items page, from the dropdown for the assigned work item, select **Reassign Orchestration Work Item**.
3. In the Reassign Orchestration Work Item window, select the user, group, or queue to reassign the work item to.
4. Click **Reassign Orchestration Work Item**.

SEE ALSO:

Salesforce Winter '23 Release Notes: Enable Sharing for Flow Orchestration Objects (Release Update)

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To reassign a work item

- Reassign Orchestration Work Items

  OR

- Manage Orchestration Runs and Work Items

To complete assigned work

- Run Flows

Disable Default Email Notifications for Work Item Assignments

By default, an orchestration sends an email notification when an orchestration work item is assigned or reassigned to a user, group, or queue. Disable default work item notifications to stop sending emails to internal users and credentialed Experience Cloud site visitors.

1. From Setup, in the Quick Find box, enter `process automation`, and then select **Process Automation Settings**.
2. On the Process Automation Settings page, select **Stop Sending Orchestration Work Item Default Email Notifications**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To edit process automation settings:

- Customize Application

Suspend an In-Progress Orchestration

Suspend an orchestration run to wait until you're ready to continue. When you suspend a running orchestration, the current stage is also suspended. In-progress steps continue to run, but no new steps are started. If an in-progress step has output, it's stored so it can be processed when the orchestration is resumed.

You can suspend only an in-progress orchestration.

1. In the App Launcher, find and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the in-progress orchestration, select **Suspend**.
3. Click **Suspend**.

When you're ready, resume the orchestration run.

SEE ALSO:

Resume a Suspended Orchestration

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To suspend a running orchestration:

- Manage Orchestration Runs

  OR

- Manage Orchestration Runs and Work Items

Resume a Suspended Orchestration

Resume a suspended orchestration run to continue its processing. When a suspended orchestration run is resumed, the suspended stage is also resumed. When the orchestration run is resumed, it evaluates the status of in-progress steps and updates the step status where appropriate. Stored outputs from steps that were in progress when the orchestration run was suspended are processed.

You can resume a suspended orchestration or an orchestration that failed within the previous 14 days because of an error in an action or flow called by a step.

1. In the App Launcher, find and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the suspended orchestration, select **Resume**.
3. Click **Resume**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To suspend a running orchestration:

- Manage Orchestration Runs

  OR

- Manage Orchestration Runs and Work Items

Resume a Failed Orchestration

When an orchestration run failed within the previous 14 days because of an error in an action called by a step, you can fix the error and resume the orchestration.

You can resume a failed orchestration if it failed within the previous 14 days because of an error in an action called by a step.

Remember to fix the error in the called flow or action before resuming the failed orchestration run.

1. In the App Launcher, find and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the orchestration with a status of Error, select **Resume**.
3. Click **Resume**.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To suspend a running orchestration:

- Manage Orchestration Runs

  OR

- Manage Orchestration Runs and Work Items

Cancel a Running Orchestration

Cancel an in-progress orchestration from the orchestration runs list view.

You can only cancel an in-progress orchestration.

1. In the App Launcher, find and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the in-progress orchestration, select **Cancel**.
3. Click **Cancel**.

SEE ALSO:

Salesforce Winter '23 Release Notes: Enable Sharing for Flow Orchestration Objects (Release Update)

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To cancel a running orchestration:

- Manage Orchestration Runs

  OR

- Manage Orchestration Runs and Work Items

Use Orchestration Reports

Use sample flow orchestration reports to track orchestration usage. Sample reports include Orchestration Runs, Orchestration Stage Runs, Orchestration Step Runs, Orchestration Work Items, and Orchestration Run Logs. These sample reports are based on the Orchestration Runs Spring '24, Orchestration Stage Runs Spring '24, Orchestration Step Runs Spring '24, Orchestration Work Items Spring '24, and Orchestration Run Logs Spring '24 custom report types.

Note: Orchestration reports aren't added to your org when it has the maximum number of defined custom reports.

Note: If a sample report is deleted, you can't regenerate it.

Sample orchestration reports are public reports. The reports only show work items assigned directly to a user. To view work assignments for the groups or queues the user belongs to, change the filter to view all work items assigned to the user's groups or queues.

1. In the Reports list view, click **Public Reports**.
2. In the Search public reports box, enter `orchestration`.
   The five sample orchestration reports are listed.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

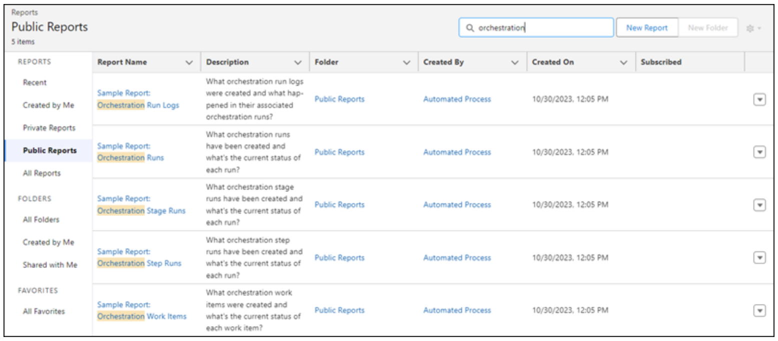
USER PERMISSIONS

To create, edit, and delete reports in public and private folders:

- Report Builder

  OR

- Report Builder (Lightning Experience)



Report Name	Description	Folder	Created By	Created On	Subscribed
Sample Report: Orchestration Run Logs	What orchestration run logs were created and what happened in their associated orchestration runs?	Public Reports	Automated Process	10/30/2023, 12:05 PM	
Sample Report: Orchestration Runs	What orchestration runs have been created and what's the current status of each run?	Public Reports	Automated Process	10/30/2023, 12:05 PM	
Sample Report: Orchestration Stage Runs	What orchestration stage runs have been created and what's the current status of each run?	Public Reports	Automated Process	10/30/2023, 12:05 PM	
Sample Report: Orchestration Step Runs	What orchestration step runs have been created and what's the current status of each run?	Public Reports	Automated Process	10/30/2023, 12:05 PM	
Sample Report: Orchestration Work Items	What orchestration work items were created and what's the current status of each work item?	Public Reports	Automated Process	10/30/2023, 12:05 PM	

3. To customize a sample report, edit the desired report.

SEE ALSO:

Salesforce Help: Build a Report in Lightning Experience

Salesforce Help: What are some common report limits?

Orchestration Statuses and Milestones

Each part of an orchestration has a status assigned when an orchestration runs.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Orchestration Details

The orchestration details page gives the status of an orchestration that's currently running.

Orchestration Status

When an orchestration runs, it can be completed, it can be canceled, it can end due to an error with a flow, or it can remain in progress. Orchestration stages, steps, and work items statuses are situation-dependent.

Statuses of Items in a Completed Orchestration

Item	Status
Orchestration Run	Completed
Completed Stages	Completed
Completed Steps	Completed
Running Steps	Discontinued
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Completed

Statuses of Items in a Canceled Orchestration

Item	Status
Orchestration Run	Canceled
Completed Stages	Completed
Running Stage	Canceled
Completed Steps	Completed
Running Steps in the Running Stage	Canceled
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Completed

Statuses of Items in an Orchestration Stopped by Orchestration Error

Item	Status
Orchestration Run	Error
Running Stage	Discontinued
Running Steps	Discontinued
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Assigned

Statuses of Items in an Orchestration Stopped by Stage Error

Item	Status
Orchestration Run	Error
Failed Stage	Error
Completed Stages	Completed
Completed Steps	Completed
Running Steps	Discontinued
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Completed

Statuses of Items in an Orchestration Stopped by Interactive Step Error

Item	Status
Orchestration Run	Error
Completed Stages	Completed
Running Stage	Error
Failed Step	Error
Completed Steps	Completed
Running Steps	Discontinued
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Assigned

Note: These statuses apply when the interactive step fails. When the screen flow associated with the interactive step fails, the status for running stage and failed step is In Progress and the status for not started work items is Assigned.

Statuses of Items in an Orchestration Stopped by Background Step Error

Item	Status
Orchestration Run	Error
Completed Stages	Completed
Running Stage	Error
Failed Step	Error
Completed Steps	Completed
Running Steps	Discontinued
Not Started Steps	Not Started
Completed Work Items	Completed
Not Started Work Items	Completed

Orchestration Run Milestones

When an orchestration runs, it logs milestones to the orchestration history.

Orchestration Run Milestone	Description
Start Run	The orchestration started or was manually resumed.
End Run	The orchestration was completed.
Suspend Run	The orchestration was manually suspended.

Orchestration Run Milestone	Description
Cancel Run	The orchestration was manually canceled.
Resume Run	The orchestration was manually resumed.
Fail Run	• The orchestration encountered an error. • A stage within the orchestration encountered an error. • A step within the orchestration encountered an error.

Troubleshoot Orchestrations

To troubleshoot a failed orchestration run, use the orchestration fault email. To test an orchestration and observe what happens as it runs, use the debug option.

Emails about Orchestration Errors
When an orchestration run fails, Salesforce sends an error email. The email is sent to either the admin who last modified the associated orchestration or the Apex exception email recipients.

Debug an Orchestration
You can view debug information for in-progress and failed orchestrations.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Emails about Orchestration Errors

When an orchestration run fails, Salesforce sends an error email. The email is sent to either the admin who last modified the associated orchestration or the Apex exception email recipients.

The email includes the error message with details about the:

- Orchestration
- Executed orchestration elements
- Flows called from orchestration steps

For activated orchestrations, the error email also has a link to show the failed orchestration run details in Flow Builder.

If an orchestration fails because of a flow it calls, then the recipients receive an error email for the orchestration failure and an error email for the flow failure.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Example:

```
Error element Stage_1 (FlowOrchestratedStage).
An error occurred when executing a flow interview.

Flow Details
Flow API Name: Create_Customer_Record
Type: Orchestrator
Version: 1
Status: Inactive
```

```
Org: signup.org.test.1640285093849 (00DRM000000G0SV)

Flow Interview Details
Interview Label: Create New Customer 2/11/2022, 1:57 PM
Interview GUID: 1fb36a45416070aa772cba20517eea2a1236-7f18
Current User: Test User (005RM0000025zTa)
Start time: 2/11/2022, 1:57 PM
Duration: 3 seconds

How the Interview Started
Orchestration Run ID: 0jERM0000004CQT
Test User (005RM0000025zTa) started the flow interview.
API Version for Running the Flow: 54

ENTER STAGE: Stage 1
ID: 0jFRM0000004CQY
Status: Error

BACKGROUND STEP: Create Account for New Customer
ID: 0jLRM0000004Cfd
Status: Error
Entry Condition:
When the stage starts, the step starts = true
Flow (Create_Account_for_New_Customer)
Inputs:
None.
Outputs:
None.

Error Occurred: An error occurred when executing a flow interview.

Salesforce Error ID: 904995012-1848 (1749972898)
```

Debug an Orchestration

You can view debug information for in-progress and failed orchestrations.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

How Does Debugging Work for Orchestrations?

View debug details in Flow Builder for only in-progress and failed orchestrations runs. View debug details in error emails for failed flows.

Note: When an orchestration fails, it doesn't necessarily roll back record additions, changes, or deletions that were made before the orchestration failed. As a result, we recommend that you design and debug your orchestration in a sandbox environment before deploying it to production.

The debug information for in-progress and failed orchestrations is similar to the information displayed for flow. In addition, orchestration debug details show milestones for orchestrations, stages, steps, and work items.

Milestones

Orchestration milestones are a part of orchestration debugging details.

Orchestration Run Milestone	Description
Start Run	The orchestration started or was manually resumed.
End Run	The orchestration was completed.
Suspend Run	The orchestration was manually suspended.
Cancel Run	The orchestration was manually canceled.
Resume Run	The orchestration was manually resumed.
Fail Run	• The orchestration encountered an error. • A stage within the orchestration encountered an error. • A step within the orchestration encountered an error.

Stage milestones are a part of orchestration debugging details.

Stage Milestone	Description
Start Stage	The stage started or was resumed when its parent orchestration was resumed.
End Stage	The stage was completed.
Suspend Stage	The stage was manually suspended.
Discontinue Stage	• The orchestration encountered an error after the stage was completed. • The orchestration encountered an error while the stage was in progress.
Fail Stage	• The stage encountered an error. • A background step within the stage encountered an error. • An autolaunched flow called by a background step within the stage encountered an error.

Step milestones are a part of orchestration debugging details.

Step Milestone	Description
Start Step	The step started.
End Step	The step was completed.
Discontinue Step	• The step's stage was completed while the step was in progress.

Step Milestone	Description
	• The orchestration encountered an error after the step was completed. • The orchestration encountered an error while the step was in progress. • The step's stage encountered an error while the step was in progress.
Fail Step	The step encountered an error.

Work item milestones are a part of orchestration debugging details.

Work Item Milestone	Description
Start Work Item	The work item was created and assigned.
End Work Item	The assigned user completed the work item. When the work item reaches the End Work Item milestone, the Completed By field is set to the username of the user or credentialed Experience Cloud site visitor who completed the work. When the work item reaches this milestone, the Duration field is set to the duration of the step in seconds.
Reassign Work Item	The work item was reassigned.

Debug an In-Progress Orchestration

Debug an in-progress orchestration to better understand the path an orchestration takes with different scenarios and the variable values at points in the automation

Debug a Failed Orchestration

Troubleshoot a failed orchestration, and gain insights about why it failed. You can debug a failed orchestration within 14 days of it failing.

Debug an In-Progress Orchestration

Debug an in-progress orchestration to better understand the path an orchestration takes with different scenarios and the variable values at points in the automation

Sharing must be enabled for orchestration runs and flow interviews.

- The orchestration run to be debugged must be shared with the user.
- The flow interview associated with the orchestration run to be debugged must be shared with the user.

1. In the App Launcher, find and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the desired in-progress orchestration, select **Debug Orchestration**.

   Note: If you started running an orchestration before upgrading to Spring '22, stage and step instance IDs are shown as null in orchestration debug information. Evaluation flow output is also shown as null.

SEE ALSO:

Salesforce Winter '23 Release Notes: Enable Sharing for Flow Orchestration Objects (Release Update)

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To access the debug action for a running orchestration:

- Manage Orchestration Runs

  OR

- Manage Orchestration Runs and Work Items

Debug a Failed Orchestration

Troubleshoot a failed orchestration, and gain insights about why it failed. You can debug a failed orchestration within 14 days of it failing.

1. From Setup, in the Quick Find box, enter, and select **Orchestration Runs**.
2. On the Orchestration Runs page, from the dropdown for the desired failed orchestration, select **Debug**.

   Note: If you started running an orchestration before upgrading to Spring '22, stage and step instance IDs are shown as null in orchestration debug information. Evaluation flow output is also shown as null.

SEE ALSO:

Salesforce Winter '23 Release Notes: Enable Sharing for Flow Orchestration Objects (Release Update)

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

USER PERMISSIONS

To debug a failed orchestration:

- Manage Flow

Flow Orchestration Limits and Considerations

When designing, managing, and running orchestrations, consider these issues.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

General Flow Orchestration Limits

When using orchestrations, keep orchestration limits, flow limits, and Apex governor limits in mind.

Considerations for Orchestrations

Keep these considerations in mind when designing and using orchestrations.

Considerations for Evaluation Flows

Keep these considerations in mind when using evaluation flows as entry or exit conditions.

Security Considerations for Orchestrations

When designing orchestrations, keep these security considerations in mind.

General Flow Orchestration Limits

When using orchestrations, keep orchestration limits, flow limits, and Apex governor limits in mind.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Per-Org Limit	Enterprise, Unlimited, Performance, or Developer Editions
Versions per orchestration	50
Active flows plus orchestrations	2,000
Total flows plus orchestrations	4,000

SEE ALSO:

Automate Your Business Processes : Flow Usage-Based Entitlements

Sales Productivity : Email Allocations per Edition

Platform Events Developer Guide: Platform Event Allocations

Considerations for Orchestrations

Keep these considerations in mind when designing and using orchestrations.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Entry and Exit Condition Requirements

Resources selected for a requirement for a step entry condition or a stage or step exit condition can contain orchestration resources or global variables. There are limitations for what can be included in a requirement.

- To use a record for the Resource or Value fields, you must select a field on the record.
- The referenced record must use fields from its object, not fields from a related record.

Record-Change-Triggered Flow Orchestration Events

A requirement for a step entry condition or a stage or step exit condition can contain a reference to a record. Changes to that record can trigger the orchestration to evaluate the status of the current stage and the outstanding steps within it. There are limitations for when the record can trigger condition evaluations.

- The referenced record's parent object must support change events.
- The referenced record fields aren't IsDeleted, SystemModeStamp, or any field that's derived from a related record or a formula.
- The referenced record is null or has an invalid ID.
- The referenced record is a global variable in an autolaunched orchestration.
- The referenced record is a global variable other than $Record in a record-triggered orchestration.

Input Values for Flows

If the combined input values for a flow called by an orchestration step is more than 32,768 characters, the orchestration fails. This error can be caused by passing one or more records to a flow called by a step. To avoid this error, pass a record ID to the referenced flow, and use a Get Records element in the flow with the passed ID. Using a passed ID with a Get Records element also means that you always have the latest version of the record.

Email Notifications

When a flow called by a step fails and causes an orchestration to fail, two email notifications are sent.

- A flow error notification
- An orchestration error notification

SEE ALSO:

Object Reference for the Salesforce Platform: StandardObjectNameChangeEvent

Considerations for Evaluation Flows

Keep these considerations in mind when using evaluation flows as entry or exit conditions.

An evaluation flow is a flow with a process type of Evaluation Flow. It's an autolaunched flow that contains a predefined Boolean output variable named `isOrchestrationConditionMet`.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

General Guidelines

Use an evaluation flow to pause an orchestration until a specific field update occurs.

Don't loop through records or make external callouts in evaluation flows.

To pass variables from the orchestration into an evaluation flow, use evaluation flow input variables.

Output Variable

An evaluation flow has one output variable named `isOrchestrationConditionMet`.

The `isOrchestrationConditionMet` output variable must be Boolean and initialized to false.

The values of all output variables other than `isOrchestrationConditionMet` are discarded and not used by the orchestration.

Security Considerations for Orchestrations

When designing orchestrations, keep these security considerations in mind.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Shield Platform Encryption

For enhanced security, enable Shield Platform Encryption for the `Screen Flow Inputs` field of the `Flow Orchestration Work Item` object.

SEE ALSO:

Salesforce Help: Strengthen Your Data's Security with Shield Platform Encryption

Flow Orchestration Entitlements

Flow Orchestration has usage-based entitlements. An orchestration *run* is a running instance of an orchestration. An *orchestration* is an application built by your admin that uses stages, steps, and decisions to organize a complex business process.

Flow Orchestration is automatically enabled for the editions listed in the Required Editions table.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

How To Get It	What's Included	Notes
Free orchestration runs	All orgs receive 600 orchestration runs per year at no charge. These runs reset every 12 months back to 600, regardless of how many were used in the previous year.	These runs aren't available if the org has one or more of the Workflow Orchestration Runs SKU.

How To Get It	What's Included	Notes
Workflow Orchestration Runs SKU	Provides 12 orchestration runs each year on an org-wide basis.	
Workflow Orchestration User SKU	Provides one user license with contractual restrictions for Flow Orchestrations. This license is for non-CRM users who are assigned steps in an orchestration.	Contractual restrictions apply. Contact your Salesforce account rep to learn more.

Flow Orchestration Reference

Bookmark this page for quick access to information about orchestration elements, resources, events, and more.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Flow Orchestration Resources

Each *resource* represents a value that you can reference throughout the orchestration.

Flow Orchestration Elements

Each *element* represents an action that the orchestration can execute. Orchestrations can contain Decision and Stage elements.

Flow Orchestration Connectors

Connectors determine the available paths that an orchestration can take at run time. On the canvas in Flow Builder, a connector looks like an arrow that points from one element to another.

Flow Orchestration Operators

In conditions and filters, operators let you evaluate information and narrow the scope of an orchestration operation.

Flow Orchestration Version Properties

An orchestration version's properties consist of its label and description. These values drive the field values that appear on the orchestration's detail page.

Flow Orchestration Resources

Each *resource* represents a value that you can reference throughout the orchestration.

In Flow Builder, the Manager tab shows the resources that are available in the orchestration.

You can create some resources by clicking **New Resource**. The system providers certain resources, such as global constants and global variables. Other resources are created when you add an element to an orchestration. For example, when you add a Decision element, a resource for each decision outcome is created.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Resource	Description	Creatable from the Resources Tab
Constant	Store a fixed value that you can use throughout an orchestration.	✓
Decision Outcome	When you add a Decision element to an orchestration, its outcomes are available as Boolean resources. If an outcome	

Resource	Description	Creatable from the Resources Tab
	path has already been executed in the running orchestration, the resource's value is `True`.	
`Element`	Any element that you add to an orchestration is available as a resource with the `was visited` operator in decision outcome criteria. An element is considered visited when it's executed in a running orchestration.	
Formula	Calculate a value when the formula is used in the orchestration.	
Flow Orchestration Resource: Global ConstantsGlobal Constant	Use fixed, system-provided values such as `EmptyString`, `True`, and `False`.	
Global Variable	Use system-provided variables that reference information about the Salesforce org or running user, such as the user's ID or the API session ID.	
Flow Orchestration Resource: Step	Organize the work done in an orchestration stage.	
Text Template	Store text that can be changed and used throughout the orchestration. To format the text, use HTML tags.	
Variable	Store a value that can be changed throughout the orchestration.	✓

Flow Orchestration Resource: Constant
Store a fixed value that you can use throughout an orchestration.

Flow Orchestration Resource: Formula
Calculate a value when the formula is used in the orchestration.

Flow Orchestration Resource: Global Constants
Use fixed, system-provided values such as `EmptyString`, `True`, and `False`.

Flow Orchestration Resource: Global Variables
Use system-provided variables that reference information about the Salesforce org or running user, such as the user's ID or the API session ID.

Flow Orchestration Resource: Step
Organize the work done in an orchestration stage.

Flow Orchestration Resource: Text Template
Store text that can be changed and used throughout the orchestration. To format the text, use HTML tags.

Flow Orchestration Resource: Variable
Store a value that can be changed throughout the orchestration.

Flow Orchestration Resource: Constant

Store a fixed value that you can use throughout an orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate the constant from other resources.
`Data Type`	Determines the type of value that the constant can store. You can't change the data type of a previously saved constant.
`Value`	The constant's value. This value doesn't change throughout the orchestration.

Flow Orchestration Resource: Formula

Calculate a value when the formula is used in the orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this formula from other resources.
`Data Type`	The data type for the value returned by the formula. You can't change the data type of a previously saved variable.
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your orchestration runs. Available only when the data type is Number or Currency.
`Formula`	The formula expression that the orchestration evaluates at run time. The returned value must be compatible with `Data Type`. Some formula functions aren't supported in Flow Builder.

Flow Orchestration Resource: Global Constants

Use fixed, system-provided values such as `EmptyString`, `True`, and `False`.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Global Constant	Supported Data Types
`{!$GlobalConstant.True}`	`Boolean`
`{!$GlobalConstant.False}`	`Boolean`
`{!$GlobalConstant.EmptyString}`	`Text`

Example: When you create a Boolean variable, `$GlobalConstant.True` and `$GlobalConstant.False` are supported. When you create a Currency variable, no global constants are supported.

Null Versus Empty String

At run time, `{!$GlobalConstant.EmptyString}` and `null` are treated as separate, distinct values. For example:

- When you leave a text field or resource value blank, the value is `null` at run time. If you want the value to be treated as an empty string, set it to `{!$GlobalConstant.EmptyString}`.
- For an orchestration condition, use the `is null` operator to check whether a value is `null`. If the condition compares two text variables, make sure that their default values are correctly set to `{!$GlobalConstant.EmptyString}` or left blank (`null`).

Flow Orchestration Resource: Global Variables

Use system-provided variables that reference information about the Salesforce org or running user, such as the user's ID or the API session ID.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Global Variable	Description
`$Api`	References an API URL or the session ID. These merge fields are available. • `Enterprise_Server_URL_`***`xxx`***—The Enterprise WSDL SOAP endpoint, where *`xxx`* represents the API version. • `Partner_Server_URL_`***`xxx`***—The Partner WSDL SOAP endpoint, where *`xxx`* represents the API version. • `Session_ID`
`$Label`	References custom labels. This global variable appears only if your org has custom labels. The returned value depends on the language setting of the contextual user. The value returned is one of the following, in order of precedence: • The local translation's text • The packaged translation's text • The primary label's text
`$Organization`	References information about your company.

Global Variable	Description
`$Permission`	References information about the current user's custom permission access.
`$Profile`	References information from the current user's profile, such as license type or name. • Use profile names to reference standard profiles in `$Profile` merge fields. • Users don't need access to their profile information to run a flow that references these merge fields.
`$Record`	Available only in record-triggered orchestrations. In a record-triggered orchestration, the orchestration makes a copy of the triggering record when it starts. If a step within that orchestration makes a field update on the triggering record, it's not reflected in the `$Record` global variable. To make field updates on the orchestration-triggering record called by one of the orchestration's steps, pass the record ID to the step's flow and use it in a Get Records element. You can reference `$Record` values throughout the orchestration by passing the variable as an input parameter to a flow in a step.
`$Record__Prior`	Available only in record-triggered orchestrations that are configured to run when a record is updated or when a record is created or updated. The `$Record__Prior` global variable contains the values that the triggering record had immediately before the orchestration started. You can't change these values in the orchestration or with any flows associated with its steps. When the orchestration is triggered by a newly created record, all `$Record__Prior` values are `null`.
`$Setup`	References custom settings of type `hierarchy`. This global variable appears only if your org has hierarchy custom settings. You can access custom settings of type `list` only in Apex. Hierarchical custom settings allow values at these levels. Salesforce determines the correct value for this custom setting field based on the running user's current context. • Organization—The default value for everyone • Profile—Overrides the Organization value • User—Overrides the Organization and Profile values
`$System`	`$System.OriginDateTime` represents the literal value of `1900-01-01 00:00:00`. Use this merge field to perform date/time offset calculations.
`$User`	References information about the user who's running the orchestration. Most of the time, an orchestration's running user is the automated process user. When the running user is the automated process user, $User doesn't provide useful information. `$User.UITheme` and `$User.UIThemeDisplayed` identify the look and feel that the running user sees on a Salesforce page. The difference between the two variables is that `$User.UITheme` returns the look and feel the user is supposed to see, while `$User.UIThemeDisplayed` returns the look and feel the user actually sees. For example, a user can have the preference and permissions to see the Lightning Experience look and feel, but if they're using a browser that doesn't support that look and feel, for example, older versions of Internet Explorer, `$User.UIThemeDisplayed` returns a different value. These merge fields return one of these values. • `Theme1`—Obsolete Salesforce theme • `Theme2`—Salesforce Classic 2005 user interface theme

Global Variable	Description
	- Theme3—Salesforce Classic 2010 user interface theme - Theme4d—Modern "Lightning Experience" Salesforce theme - Theme4t—Salesforce mobile app theme - Theme4u—Lightning Console theme - PortalDefault—Salesforce Customer Portal theme - Webstore—AppExchange theme
$UserRole	References information about the current user's role, such as the role name or ID.

Global Variable Considerations for Flows

- In a record-triggered orchestration, the $Record global variable doesn't contain the triggering record's values for fields whose values are derived from other records. Examples of derived fields include Contact.Name and User.MediumPhotoUrl.
- Multi-select picklist, time, and location global variables are available only in formulas.
- If a field in the database has no value, the corresponding merge field returns a blank value. For example, if no value is set for your org's Country field, {!$Organization.Country} returns no value.

Flow Orchestration Resource: $Flow Global Variables

A $Flow global variable provides information about the running orchestration. Some variables contain system-provided values. You can update the other variables throughout the orchestration by storing output values in the variables.

Flow Orchestration Resource: $Flow Global Variables

A $Flow global variable provides information about the running orchestration. Some variables contain system-provided values. You can update the other variables throughout the orchestration by storing output values in the variables.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Global Variable	Supported Resource Types	Description	Value Set By
$Flow.ActiveStages	Stage	A collection of stages that are relevant to the current path of the flow. This system variable references the flow Stage resource, not the orchestration Stage element. It can only be used in flows, including those flows called by an orchestration step, but it isn't supported for orchestrations.	Assignment
$Flow.CurrentDate	Text, Date, and Date/Time	Date when the flow interview executes the element that references the global variable.	System

Global Variable	Supported Resource Types	Description	Value Set By
`$Flow.CurrentRecord`	Text	ID of a related record. The value must be a single ID for a valid object. All custom objects and most standard objects are valid. When a user pauses the flow interview or the interview executes a Wait element, the interview is associated with this record by creating a FlowRecordRelation record. If the ID isn't valid, the interview fails to pause.	Assignment
`$Flow.CurrentStage`	Stage	The currently selected stage. This system variable references the flow Stage resource, not the orchestration Stage element. It can only be used in flows, including those flows called by an orchestration step, but it isn't supported for orchestrations.	Assignment
`$Flow.CurrentDateTime`	Text, Date, and Date/Time	Date and time when the flow interview executes the element that references the global variable.	System
`$Flow.FaultMessage`	Text	System fault message that can help flow administrators troubleshoot runtime issues.	System
`$Flow.InterviewGuid`	Text	Unique identifier for the interview.	System
`$Flow.InterviewStartTime`	Text, Date, and Date/Time	Date and time when the flow interview started. For a flow launched by a Subflow element, `$Flow.InterviewStartTime` indicates when the initial parent flow started.	System

Flow Orchestration Resource: Step

Organize the work done in an orchestration stage.

Orchestrations have background steps and interactive steps.

Note: The Step resource in Flow Orchestration isn't related to the discontinued Step element in Flow Builder.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Background Steps

Background steps call autolaunched flows and run without user interaction.

Field	Description
`Label`	Helps you identify the element on the canvas.
`API Name`	Automatically populated if empty when you fill out the `Label` field and press TAB.The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name,

Field	Description
	provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you remember what this resource does. When editing an element, appears after you click ✏.
`Condition`	Identifies the method used to determine whether a step is ready to start.
`Step Name`	Specifies a step that must be marked complete before the current step can start. Available when the entry condition is set to When another step is marked complete the step starts.
`Evaluation Flow`	Specifies the flow that determines if the step can start. Available when the entry condition is set to When the specified evaluation flow returns True, the step starts.
`Flow`	Specifies which autolaunched flow to run for a step.

Interactive Steps

Interactive steps call screen flows and require user interaction.

Field	Description
`Label`	Helps you identify the element on the canvas.
`API Name`	Automatically populated if empty when you fill out the `Label` field and press TAB.The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you remember what this resource does. When editing an element, appears after you click ✏.
`Condition`	Identifies the method used to determine whether a step is ready to start or can be considered complete.
`Step Name`	Specifies a step that must be marked complete before the current step can start. Available when the entry condition is set to When another step is marked complete the step starts.
`Evaluation Flow`	Specifies the flow that determines if the step can start or be marked complete. Available when the entry condition is set to When the specified evaluation flow returns True, the step starts. Also available when the exit condition is set to When the specified evaluation flow returns True, the step is marked Completed.
`Flow`	Specifies which screen flow to run for a step.
`Record ID`	Specifies the ID of the record where the Work Guide displays the screen flow to the assigned user.
`Username`	Specifies the user assigned to complete the screen flow.

Step Status

Step Status	Description
Not Started	The step hasn't met its entry condition.
In Progress	The step was started.
Completed	• The interactive step met its exit condition. • The background step's flow was completed. • The step was completed when its associated stage encountered an error.
Discontinued	• The step was in progress when its associated stage completed. • The step was in progress when the orchestration encountered an error. • The step was in progress when its associated stage encountered an error.
Error	• The step encountered an error. • The autolaunched flow associated with a background step encountered an error. • The screen flow associated with an interactive step encountered an error. • The MuleSoft action associated with a MuleSoft step encountered an error.

Flow Orchestration Resource: Text Template

Store text that can be changed and used throughout the orchestration. To format the text, use HTML tags.

Field	Description
`API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this text template from other resources.
Text Template	The text for the template. To reference information from other resources, use merge fields.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Field	Description
Rich Text	Control the text font, size, color, and alignment. Add HTML links, bullet points, or numbered lists. Rich text is on by default. To change to rich text, click ▼ .
Plain Text	Send email core actions use plain text. Some custom actions from AppExchange or built by Salesforce developers also expect plain text. To change to plain text, click ▼ .

Flow Orchestration Resource: Variable

Store a value that can be changed throughout the orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Field	Description
`Apex Class`	Defines fields for the Apex-defined data type. Only fields with the @AuraEnabled annotation are available in an orchestration.
`API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Description`	Helps you differentiate this variable from other resources.
`Data Type`	Determines the types of values that the variable can store. You can't change the data type of a previously saved variable. The Record data type can store multiple field values for one record. The Apex-defined data type can store multiple field values for one Apex class. Looking for sObject? In Flow Builder, that data type changed to Record.
`Allow multiple values (collection)`	When selected, the resource is a collection variable. You can store a list of values in collection variables. Collection variables can store only values that are compatible with their data type. When the data type is Record, the collection variable can only store values for the associated object's records. For example, store multiple email addresses in a collection variable, and reference the collection variable to send an email.
`Object`	The object whose field values you can store in the variable. You can't change the object of a previously saved variable. Available only when the data type is Record.

Field	Description
`Decimal Places`	Controls the number of digits to the right of the decimal point up to 17 places. If you leave this field blank or set it to zero, only whole numbers appear when your orchestration runs. Available only when the data type is Number or Currency.
`Availability Outside the Flow`	When a variable is available for input, it can be set at the start of the orchestration, such as when an orchestration is started from a Lightning page. Disabling input or output access for an existing variable can break the functionality of applications and pages that call the orchestration and access the variable. For example, you can access variables from URL parameters, processes, and other flows. This field doesn't affect how variables are assigned or used within the same orchestration.
`Default Value`	Determines the variable value when the orchestration starts. If you leave this field blank, the value is `null`. Not available for Picklist and Multi-Select Picklist variables.

Flow Orchestration Elements

Each *element* represents an action that the orchestration can execute. Orchestrations can contain Decision and Stage elements.

In Flow Builder, the Add Element menu shows the types of elements that you can add to the flow by selecting them. For a list of all elements already added to the orchestration, see the Elements section of the Manager tab in the Toolbox.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Flow Orchestration Element: Decision

Evaluate a set of conditions, and then route users through the orchestration based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

Flow Orchestration Element: Stage

Group a series of related steps in an orchestration.

Flow Orchestration Element: Decision

Evaluate a set of conditions, and then route users through the orchestration based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Outcomes

For each path that the orchestration can take, create an outcome. For each outcome, specify the conditions that must be met for the orchestration to take that path. To relabel the path that the flow takes if no outcome's conditions are met, click **Default Outcome**.

Field	Description
`Label`	Identifies the connector for this outcome on the canvas.

Field	Description
`Outcome API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Condition Requirements to Execute Outcome`	Determines whether the orchestration takes this outcome's path. Sets logic and conditions for each outcome that determine if the orchestration follows its path.
`When to Execute Outcome`	Available on record-triggered orchestrations. Determines whether this outcome's path is taken based on whether the triggering record is updated to meet the condition requirements. For example, the opportunity update that triggered the orchestration to run changed its stage to Closed Won from any value that isn't Closed Won.

Flow Orchestration Element: Stage

Group a series of related steps in an orchestration.

Stages run sequentially, one stage at a time, and contain steps.

Note: The Stage element in Flow Orchestration isn't related to the Stage resource in Flow Builder.

Field	Description
`Label`	Identifies the name for this stage on the canvas.
`API Name`	The requirement for uniqueness applies only to elements within the current orchestration. Two elements can have the same API name, provided they're used in different orchestrations. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores.
`Set Exit Condition`	Determines when a stage can be considered complete. **When all steps have been marked Complete, the stage is marked Complete** The stage is marked complete and the orchestration moves to the next element when every step in a stage is marked complete. **When the specified evaluation flow returns True, the stage is marked Complete** The orchestration runs a specified evaluation flow to determine if the stage can be marked complete. The orchestration doesn't mark the stage complete and move to the next element until the specified evaluation flow's `isOrchestrationConditionMet` output variable returns true.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

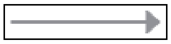
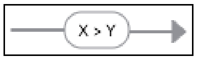
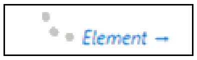
This feature is supported in Government Cloud and Government Cloud Plus.

Stage Status

Stage Status	Description
In Progress	The stage was started.
Completed	• The stage was completed, and the orchestration was completed. • The stage was completed, and the orchestration is in progress. • The stage was completed, and the orchestration was canceled.
Suspended	The stage was in-progress when the orchestration was manually suspended.
Canceled	The stage was in progress when the orchestration was canceled.
Discontinued	The stage was in progress when the orchestration encountered an error.
Error	• The stage encountered an error. • A background step within the stage encountered an error. • An autolaunched flow called by a background step within the stage encountered an error. • The stage was in progress when an interactive step within the stage encountered an error. • The stage was in progress when a screen flow associated with an interactive step within the stage encountered an error. • The stage was in progress when a MuleSoft step within the stage encountered an error. • The stage was in progress when a MuleSoft action associated with a MuleSoft step within the stage encountered an error.

Flow Orchestration Connectors

Connectors determine the available paths that an orchestration can take at run time. On the canvas in Flow Builder, a connector looks like an arrow that points from one element to another.

Label	Example	Description
Unlabeled		Identifies which element to execute next.
`Decision outcome`		Identifies which element to execute when the criteria of a Decision outcome are met.
Go To		Identifies which element to go to and execute next. Use to create loops in an orchestration.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Flow Orchestration Operators

In conditions and filters, operators let you evaluate information and narrow the scope of an orchestration operation.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Flow Orchestration Operators in Decision Elements

Use condition operators to verify the value of a selected resource. Conditions are used in Decision elements.

Flow Orchestration Operators in Decision Elements

Use condition operators to verify the value of a selected resource. Conditions are used in Decision elements.

Use this reference to understand the supported operators. The list is organized according to the data type that you select for Resource.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

- Apex-Defined
- Boolean
- Collection
- Currency
- Date
- Date/Time
- Multi-Select Picklist
- Number
- Picklist
- Record
- Text

Apex-Defined

To determine which operators are supported, match the *@AuraEnabled* attribute's Apex data type with the Flow Orchestration data type in this reference.

Apex Data Type	Flow Orchestration Data Type
Boolean	Boolean
Date	Date
DateTime	Date/Time
Decimal	Number
Double	Number

Apex Data Type	Flow Orchestration Data Type
Integer	Number
List	Collection
Long	Number
String	Text

Boolean

Check whether a Boolean resource's value matches another value or resource.

Operator	True if...	Supported Data Types
Does Not Equal	The value of the selected Resource doesn't match what you enter or select for Value.	Boolean
Equals	The value of the selected Resource matches what you enter or select for Value. • If the running orchestration took that outcome, a decision outcome resolves to `true`. • If executed without error, a stage element resolves to `true`. If a fault occurred, the element resolves to false. If the element wasn't executed, it resolves to `null`.	Boolean
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean
Was Visited	The selected Resource is a Decision element in the orchestration, and it has been visited while the orchestration is running.	Boolean

Collection

Check whether a Collection resource's value contains or matches another value or resource.

Operator	True If	Supported Data Types
Contains	An item in the collection that's selected for Resource contains the exact same value as Value.	Resource of the same data type. For record collection variables, only record resources with the same object type are supported.
Does Not Equal	The collection that's selected for Resource doesn't match the collection that's selected for Value. If two record collection variables include different fields or if the fields have different values, they're unequal.	Collection of the same data type. For record collection variables, only record collection variables with the same object type are supported.

Operator	True If	Supported Data Types
Equals	The collection that's selected for Resource matches the collection that's selected for Value. If two record collection variables include the same fields and those fields have the same values, they're equal.	Collection of the same data type. For record collection variables, only record collection variables with the same object type are supported.
Is Empty	An empty collection	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean

Currency and Number

Check whether a Currency or Number resource's value matches, is larger than, or is smaller than another value or resource.

Operator	True If	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value.	• Currency • Number
Equals	The value for Resource matches what's entered or selected for Value.	• Currency • Number
Greater Than	The value for Resource is larger than what's entered or selected for Value.	• Currency • Number
Greater Than or Equal	The value for Resource is larger than what's entered or selected for Value or is the same.	• Currency • Number
Less Than	The value for Resource is smaller than what's entered or selected for Value.	• Currency • Number
Less Than or Equal	The value for Resource is smaller than what's entered or selected for Value or is the same.	• Currency • Number
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean

Date and Date/Time

Check whether a Date or Date/Time resource's value matches, is before, or is after another value or resource.

Operator	True If	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value.	• Date • Date/Time
Equals	The value for Resource matches what's entered or selected for Value.	• Date • Date/Time
Greater Than	The value for Resource is a later date or time than what's entered or selected for Value.	• Date • Date/Time
Greater Than or Equal	The value for Resource is a later date or time than what's entered or selected for Value or is the same date or time.	• Date • Date/Time
Less Than	The value for Resource is an earlier date or time than what's entered or selected for Value.	• Date • Date/Time
Less Than or Equal	The value for Resource is an earlier date or time than what's entered or selected for Value or is the same date or time.	• Date • Date/Time
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean

Picklist

Check whether a Picklist resource's value matches or contains another value or resource.

Note: These operators treat the resource's value as a text value.

Operator	True If	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value. For example, if the value of `{!varPicklist}` is `yellow-green`, the condition `{!varPicklist}` **Contains** `green` evaluates to `TRUE`.	• Boolean • Currency • Date • Date/Time

Operator	True If	Supported Data Types
		Multi-Select Picklist Number Picklist Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value.	Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text
Equals	The value for Resource matches what's entered or selected for Value.	Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean

Multi-Select Picklist

Check whether a multi-select picklist resource's value matches or contains another value or resource.

Note: These operators treat the resource's value as a text value. If the resource's value includes multiple items, the operators treat the value as one string that happens to include semicolons. It doesn't treat each selection as a different value. For example, the operators treat `red; blue; green` as a single value rather than three separate values.

Operator	True If	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value. When you use this operator for a multi-select picklist resource, be aware of the values that a user can enter. If you want to check that a specific value is included and that value is also included as part of another value, create an orchestration formula resource that uses the INCLUDES() function. For example, your org has a Color multi-select picklist value. Among the possible values are "green" and "yellow-green". If both "green" and "yellow-green" are acceptable values, use the Contains operator in an orchestration condition. If only "green" is an acceptable value, create a formula that uses the INCLUDES() function.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value. Order matters. If you aren't sure which order the values that you're checking for appear in, use the INCLUDES() function in an orchestration formula. For example, if you compare "red; blue; green" to "blue; green; red" using the Does Not Equal operator, that condition resolves to true.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Equals	The value for Resource exactly matches what's entered or selected for Value. Order matters. If you aren't sure of the order that the values you're checking for appear, use the INCLUDES() function in an orchestration formula. For example, if you compare "red; blue; green" to "blue; green; red" using the Equals operator, that condition resolves to `false`.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean

Record

Check whether a record resource's value matches another value or resource.

Operator	True If	Supported Data Types
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value.	Record with the same object type
Equals	The value for Resource matches what's entered or selected for Value.	Record with the same object type
Is Blank	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value similar to Is Null. See Text on page 747.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean

Text

Check whether a Text resource's value matches, contains, ends with, or starts with another value or resource.

Operator	True If	Supported Data Types
Contains	The value for Resource contains what's entered or selected for Value.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Does Not Equal	The value for Resource doesn't match what's entered or selected for Value.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Equals	The value for Resource matches what's entered or selected for Value.	• Boolean

Operator	True If	Supported Data Types
		• Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Ends With	The end of the value for Resource matches what's entered or selected for Value.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Is Blank	The value for Resource is set to zero characters or only white space. Use to determine whether a field or variable value is set to zero characters or only white space.	Boolean
Is Null	The value for Resource is either not set or references no value. Use to determine whether a field or variable value is set to no value.	Boolean
Starts With	The beginning of the value for Resource matches what's entered or selected for Value.	• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text
Was Set	The value for Resource is a field in a record variable, and that field has been populated with a value at least one time in a flow called by a step in the orchestration.	Boolean

Flow Orchestration Version Properties

An orchestration version's properties consist of its label and description. These values drive the field values that appear on the orchestration's detail page.

Property	Description
`Orchestration Label`	The label for the orchestration version. The label appears in the orchestration detail page and list views. You can edit the label for inactive orchestrations and orchestration versions.
`Flow API Name`	The API name for the orchestration. The API name is used to differentiate this orchestration from other parts of Salesforce, such as in a URL or Lightning web component. An API name can include underscores and alphanumeric characters without spaces. It must begin with a letter and can't end with an underscore. It also can't have two consecutive underscores. The API name appears on the orchestration detail page. You can't edit the API name after saving the orchestration.
`Description`	Differentiates the orchestration version from other versions. The description appears in the orchestration detail page and list views. You can edit the description for inactive orchestrations and orchestration versions.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This feature is supported in Government Cloud and Government Cloud Plus.

Suggest Options to Users with Recommendation Strategies

Display the right recommendations to the right people at the right time with Einstein Next Best Action. Create and display offers and actions for your users that are tailored to meet your unique criteria. Develop a strategy that applies your business logic to refine those recommendations. Your strategy distills your recommendations into a few key suggestions, like a repair, a discount, or an add-on service. Display the final recommendations in your Lightning app or Experience Builder site.

Note: Where possible, we recommend building strategies in Flow Builder using the Recommendation Strategy flow type, but you can also create them in Strategy Builder.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Get Started with Einstein Next Best Action

Just getting started with Einstein Next Best Action? Follow these steps to complete each phase of the Next Best Action setup process, create personalized recommendations for your users, and put decisions into action.

Einstein Next Best Actions Considerations

Keep these considerations in mind when working with strategies and recommendations.

Einstein Next Best Action Entitlements

Einstein Next Best Action has usage-based entitlements. All orgs receive a free monthly allowance of Next Best Action requests. If your usage exceeds your allowance of free monthly requests or any entitlements that you purchase, Salesforce contacts you to discuss additions to your contract. To track your usage, from Setup, navigate to **Company Information**.

Create Recommendations

Create offers or actions to recommend to users using Einstein Next Best Action. Recommendations are standard Salesforce records, similar to accounts and contacts, that are processed by strategies and associated with flows. Strategies determine which recommendation records are surfaced using business rules, predictive models, and other data sources. The result of this process is context-specific recommendations that you present to your users.

Building a Strategy

A strategy determines when and how to present an Einstein Next Best Action recommendation on a Salesforce Lightning record page. For example, if you want to offer a discount to a subset of customers, create a strategy that collects the appropriate customer records and identifies the discount option to present. To create a strategy, you can use Flow Builder (recommended) or Strategy Builder.

Display Recommendations

After creating a strategy, choose a page to run your strategy and display your recommendations. You can use a Lightning record page, an app’s home page, an Experience Cloud site page, a Visualforce page, or an external site, depending on where you want recommendations to appear.

Report On and Track a Recommendation

Create a custom report type to report on and track recommendation data and strategy metrics. You can see the monthly total recommendations that a Salesforce org’s strategies served. And you can analyze which recommendations are accepted and rejected, who responds to them, and more.

SEE ALSO:

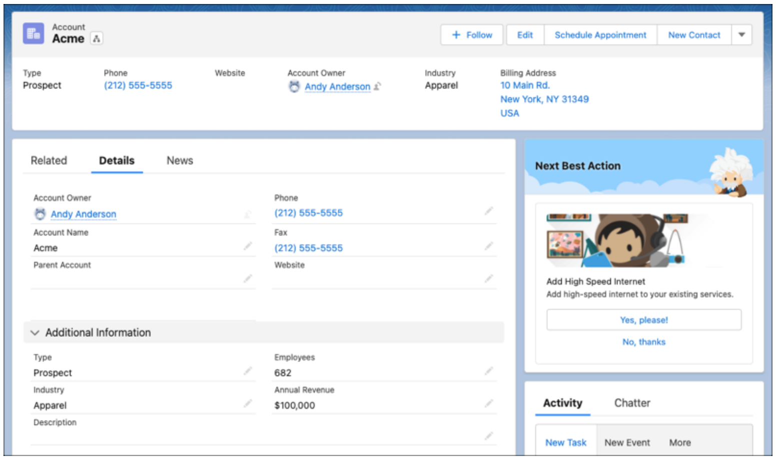
Connect REST API Developer Guide: Next Best Action Resources

Suggested Actions

Get Started with Einstein Next Best Action

Just getting started with Einstein Next Best Action? Follow these steps to complete each phase of the Next Best Action setup process, create personalized recommendations for your users, and put decisions into action.

Einstein Next Best Action is a solution that uses flows, strategies, and the Recommendation object to recommend actions to users. You can display these recommendations on many different types of pages, including Lightning pages in your Salesforce org, Experience Cloud sites, or external sites.



Account
Acme

Type
Prospect

Phone
(212) 555-5555

Website

Account Owner
Andy Anderson

Industry
Apparel

Billing Address
10 Main Rd.
New York, NY 31349
USA

Account Owner
Andy Anderson

Phone
(212) 555-5555

Account Name
Acme

Fax
(212) 555-5555

Parent Account

Website

Additional Information

Type
Prospect

Employees
682

Industry
Apparel

Annual Revenue
$100,000

Description

Add High Speed Internet
Add high-speed internet to your existing services.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Recommendations are displayed to users with the option to accept or reject the recommended action. Each recommendation contains an image, important text values such as button text and a description, and an assigned flow that runs when a user responds. They can be stored and referenced in the Recommendation standard object, or they can be manually assembled when building a strategy.

Strategies determine which recommendations to display to users, based on your data and business processes. When you set up Einstein Next Best Action on a page, you assign a strategy to that location, which then defines the recommendations that appear there.

You can control which recommendations are displayed in any situation, even if your org has a large number of recommendation records. Strategies can filter recommendations based on any available value, including recommendation fields, fields related to the running user, and fields related to the record that's currently displayed.

Important: In Flow Builder, you define which recommendations are displayed by making sure that they're in the outputRecommendations collection variable at the end of the flow. In Strategy Builder, you define which recommendations are displayed by making sure that they're not filtered out when they reach the Output element.

1. Plan Your Recommendations and Automation

   Decide where the recommendation appears, who it appears to, and the conditions in which it appears. Create a plan for the automation that you want to run when a user accepts the recommendation.

2. Build a Flow

   In Flow Builder, design and build the flow that runs when a user accepts or rejects the recommendation. You can assign only screen flows and autolaunched flows to a recommendation. If an inactive or invalid flow is assigned, the recommendation isn't displayed to users.

3. Create Recommendations

   Recommendations are standard Salesforce records, similar to accounts and contacts. To create recommendations, you can:

   - Create recommendation records on the Recommendation object.
   - Build recommendations from other data when creating your strategy. In Flow Builder, use the Recommendation Assignment element or a custom Apex invocable action.
   - Generate recommendations automatically through AI with Einstein Recommendation Builder.

4. Create a Strategy

   After you create a flow and make a plan for your recommendation records, use Flow Builder or Strategy Builder to create your strategy. Where possible, we recommend building strategies in Flow Builder using the Recommendation Strategy flow type, but you can also create them in Strategy Builder.

   Some features can be used only in strategies created in Strategy Builder.

   - Limiting repeated showings of some recommendations
   - Displaying recommendations on an Experience Cloud site or external site
   - Displaying AI-generated recommendations from Einstein Recommendation Builder

   To build a strategy in Flow Builder, follow these steps.

   a. Go to the Flows page in Setup, and click `New Flow`.

   b. Select **Use a Template**, and then click**Next**.

   c. Select the **Recommendation Strategy** flow type, and then click **Create**.

   d. To retrieve data from Salesforce records, such as the Recommendations object or an object related to the currently displayed record, add Get Records elements. To filter which recommendations are stored in the element's collection, use condition requirements in the Get Records element. Or you can build recommendations from other data with the Recommendation Assignment element or a custom Apex invocable action.

e. To limit the number of recommendations that users see, add logic elements. Use Collection Sort and Collection Filter elements to arrange and reduce the recommendations from the Get Records collection. If needed, you can also add other Flow elements such as Decision and Loop to create more complex, branching logic.

f. To set recommendations in the `outputRecommendations` collection, add the Assignment element. When running a strategy built in Flow Builder, Einstein Next Best Action displays only recommendation records in the `outputRecommendations` collection.

5. Display Next Best Actions

   After creating a strategy, choose a page to run your strategy and display your recommendations. You can use a Lightning record page, an app's home page, an Experience Cloud site page, a Visualforce page, or an external site, depending on where you want recommendations to appear.

   - Einstein Next Best Action Component

     Use the Einstein Next Best Action component to display recommendations to users on most Lightning pages within your Salesforce org, including record pages, home pages, and app pages.

   - Suggested Actions

     Use the Suggested Actions component to display recommendations on Experience Cloud sites. This component can run only strategies created in Strategy Builder.

SEE ALSO:

Build a Flow

Create Recommendations

Strategy Builder Strategies

Display Recommendations

Launch a Flow When a Recommendation Is Accepted or Rejected

Einstein Next Best Action Component

Suggested Actions

Einstein Next Best Action Examples

These examples walk you through the process of creating Einstein Next Best Action components.

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Offer a Gift Basket to Each Account

Use a Next Best Action component on the Lightning Account record page to offer a gift basket to each of your accounts. When a customer accepts the offer, a form opens to collect the recipient's name and shipping address. After the form is submitted, a request email is sent to the shipping department.

EDITIONS

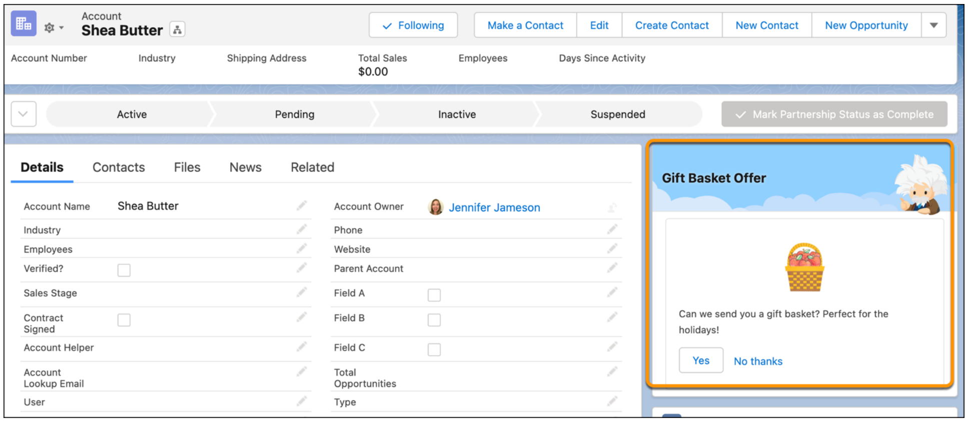
Available in: Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow



Account Number | Industry | Shipping Address | Total Sales $0.00 | Employees | Days Since Activity

Account Name: Shea Butter

Account Owner: Jennifer Jameson

Industry, Employees, Verified? □, Sales Stage, Contract Signed □, Account Helper, Account Lookup Email, User

Phone, Website, Parent Account, Field A □, Field B □, Field C □, Total Opportunities, Type

Can we send you a gift basket? Perfect for the holidays!

To configure this Einstein Next Best Action recommendation:

1. Create an action flow on page 753 that executes when the gift basket recommendation is accepted.
2. Create a recommendation on page 755 that specifies how to present the gift basket offer.
3. Create a recommendation strategy flow on page 756 that determines when and how the recommendation is presented.
4. Add a Next Best Action component on page 757 that displays the recommendation on the Account record page and executes the strategy.

Create an Action Flow

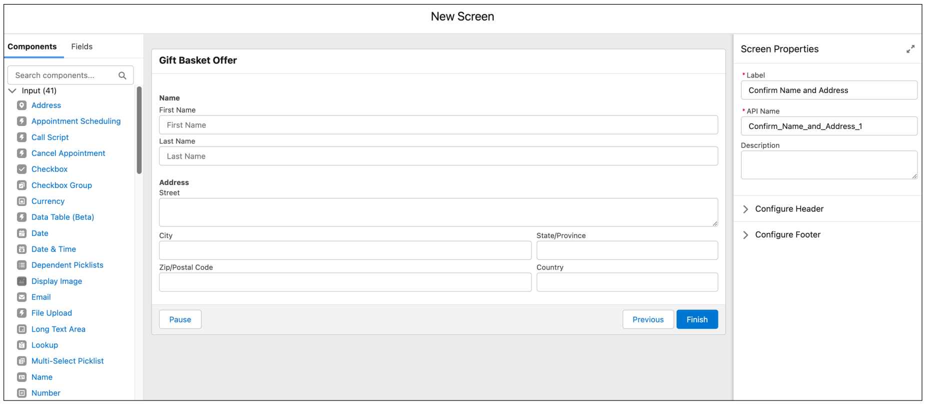
Create a flow that collects the recipient's name and address and sends an email to the shipping department.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. Select **Start From Scratch** and then click **Next**.
3. Select the **Screen** flow type and then click **Create**.
4. To collect the recipient's name and address, add a Screen element to the flow.

a. Enter a label and API name.

b. Drag the **Name** and **Address** components to the canvas and assign an API name to each.

c. Click **Done**.



New Screen

Gift Basket Offer

Name

First Name: First Name

Last Name: Last Name

Address

Street

City

State/Province

Zip/Postal Code

Country

Screen Properties

* Label: Confirm Name and Address

* API Name: Confirm_Name_and_Address_1

Description

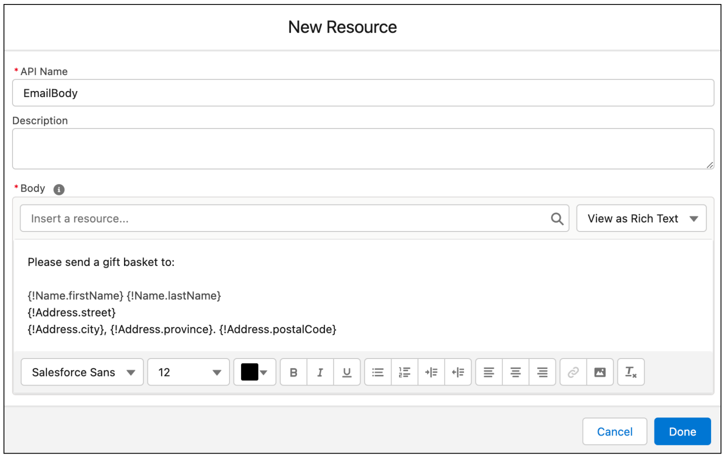
5. To create the text of the email message to send to the shipping department, click **New Resource** in the Flow Builder Toolbox. If the toolbox isn't visible, toggle the toolbox icon in the upper left corner of the Flow Builder canvas.

a. Add a Text Template resource type.

b. Enter `EmailBody` as the API name.

c. In the Body area, enter the email text, inserting the name and address resources.

d. Click **Done**.



New Resource

* API Name: EmailBody

Description

* Body

Please send a gift basket to:

{!Name.firstName} {!Name.lastName}
{!Address.street}
{!Address.city}, {!Address.province}. {!Address.postalCode}

6. To create a task for the shipping department, click ⊕ below the Screen element and add an Action element to the flow.

a. In the Action dropdown list, enter `Send Email` and select the **Send Email** action.

b. Enter a label and API name.

c. For Body, select the **EmailBody** text element.

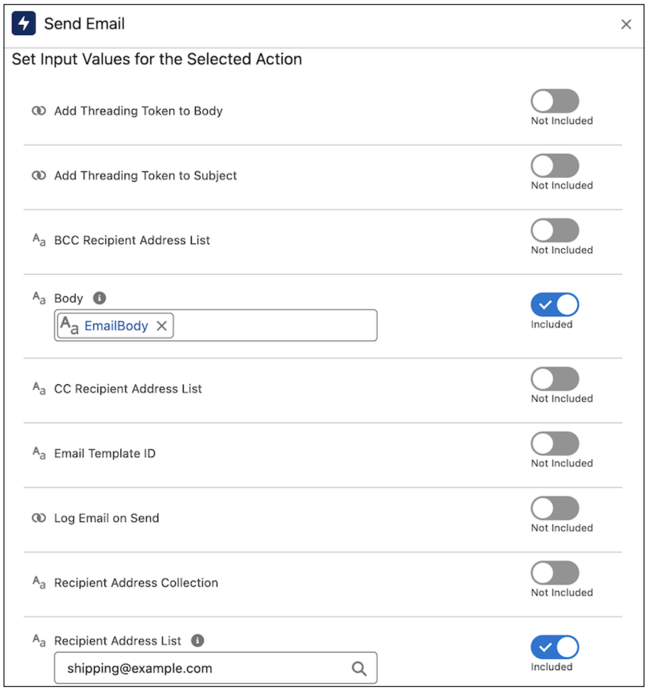
d. Enter a subject line.

e. For Recipient Email Addresses (comma-separated), select **Include** and add the email address of the shipping department.

f. To allow rich text formatting for the message, select Include and select the **True** global constant.

g. Set any other values as needed.

h. Click **Done**.



Send Email

Set Input Values for the Selected Action

Add Threading Token to Body	Not Included
Add Threading Token to Subject	Not Included
BCC Recipient Address List	Not Included
Body: EmailBody	Included
CC Recipient Address List	Not Included
Email Template ID	Not Included
Log Email on Send	Not Included
Recipient Address Collection	Not Included
Recipient Address List: shipping@example.com	Included

7. Save the flow and name it `Gift Basket Offer`.
8. Activate the flow.
9. To return to the Flows page, click **Back**.

Create a Recommendation Record

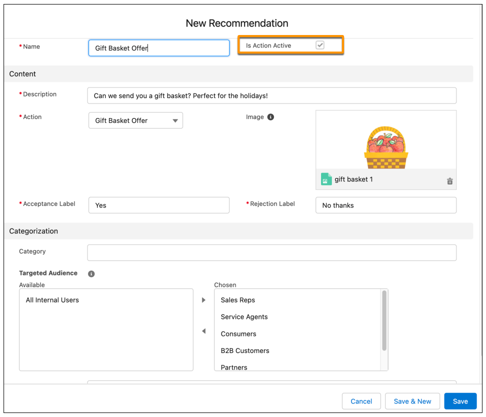
Create a recommendation that specifies how to present the gift basket offer.

1. From the App Launcher (⠿), in the Quick Find box, enter `Recommendations`, and select **Recommendations**.
2. Click **New**.
3. Enter a name and description for the recommendation.

   The description appears in the Next Best Action component on the Lightning record page.
4. For **Action**, select the action flow that you created.
5. To add an image (optional), click **Upload Image** and follow the instructions.

   For best results, use a 1000 px x 380 px image at 72 dpi or one with a similar ratio.
6. Enter text for the acceptance and rejection buttons.
7. Select the target audiences for the recommendation.
8. Click **Save**.

   The Is Action Active checkbox is automatically selected, which makes the recommendation available to Einstein Next Best Action.



New Recommendation

*Name: Gift Basket Offer

Is Action Active ☑

Content

*Description: Can we send you a gift basket? Perfect for the holidays!

*Action: Gift Basket Offer

Image: gift basket 1

*Acceptance Label: Yes

*Rejection Label: No thanks

Categorization

Category:

Targeted Audience

Available: All Internal Users

Chosen: Sales Reps, Service Agents, Consumers, B2B Customers, Partners

Create a Recommendation Strategy Flow

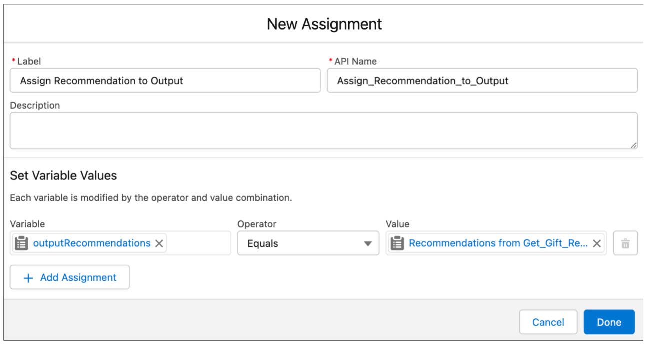
The recommendation strategy flow determines when and how the recommendation is presented.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. Select **Use a Template** and then click **Next**.
3. Select the **Recommendation Strategy** flow type and then click **Create**.
4. To specify which records to use for the recommendation, add a Get Records element to the flow.
   a. Enter a label and API name.
   b. Select the **Account** object.
   c. In the Filter section, add the condition `Id equals recordId`.
   d. Select the options to store all records and all fields.
   e. Click **Done**.
5. To load possible recommendations into the strategy, add a Get Records element.
   a. Enter the label `Get Gift Recommendation` and the API name. `Get_Gift_Recommendation`.
   b. Select the **Recommendation** object.
   c. In the Filter section, add the condition `Name contains Gift Basket`.
   d. Select the options to store all records and all fields.
   e. Click **Done**.

   In Flow Builder, you define which recommendations are displayed by making sure that they're in the outputRecommendations collection variable at the end of the strategy flow. The next step uses the Assignment element to add the recommendations to outputRecommendations. To learn how to use the Limit Repetition element to assign the outputRecommendation variable while also limiting the number of times that the user sees the recommendation, see Create Recommendations Based on Customer Satisfaction Scores on page 758.
6. To move the recommendation output out of this flow so it becomes available to Einstein Next Best Action, click **+** below the Recommendation Assignment element and add an Assignment element.
   a. Enter a label and API name.
   b. For Variable, select **outputRecommendations**.
   c. For Operator, select **Equals**.

d. For Value, select **Recommendations from Get Gift Recommendation**.

e. Click **Done**.



New Assignment

* Label: Assign Recommendation to Output

* API Name: Assign_Recommendation_to_Output

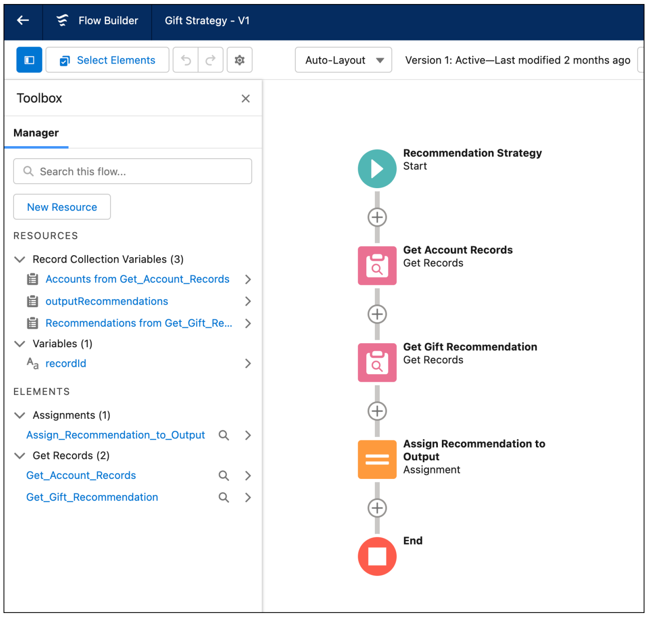
Description

Set Variable Values

Each variable is modified by the operator and value combination.

Variable	Operator	Value
outputRecommendations	Equals	Recommendations from Get_Gift_Re...

7. Save the flow and name it `Gift Strategy`.



8. Activate the flow.

9. To return to the Flows page, click **Back**.

Display the Next Best Action Recommendation

Display the Next Best Action recommendation on the Account record page.

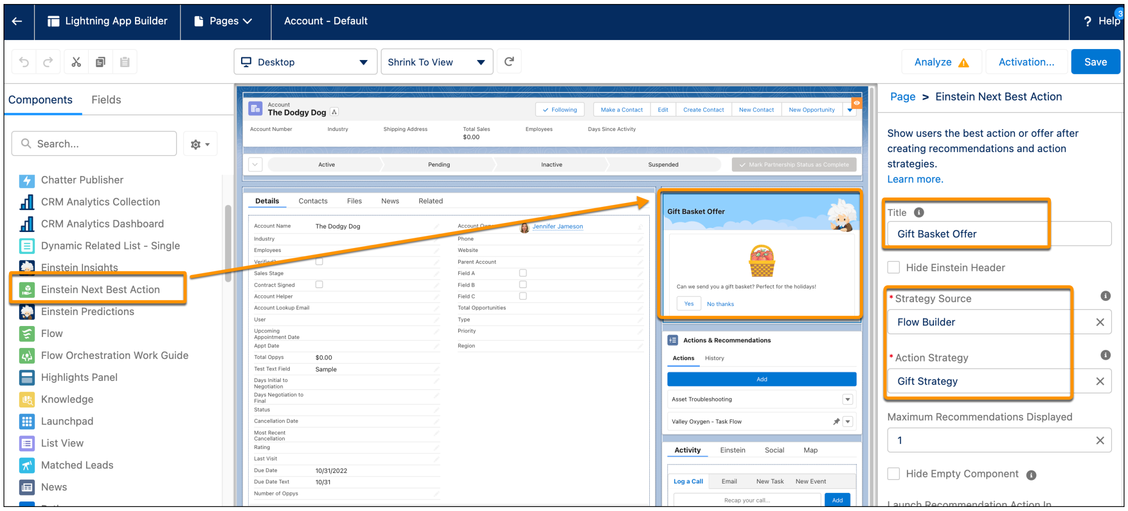
1. Open an Account record page.

2. Click the Setup icon (⚙), and select **Edit Page**.

3. Drag the Einstein Next Best Action component to the desired location on the page layout.

4. Add `Gift Basket Offer` as the component title.

5. For Strategy Source, select **Flow Builder** and then select the name of the recommendation strategy.



Show users the best action or offer after creating recommendations and action strategies. Learn more.

Title: Gift Basket Offer

□ Hide Einstein Header

* Strategy Source: Flow Builder

* Action Strategy: Gift Strategy

Maximum Recommendations Displayed: 1

□ Hide Empty Component

6. Save your changes.
7. Return to the Account record and refresh the page.

The recommendation is displayed. If the account rep clicks **Yes**, a form opens with entries for name and address. Completing the form generates an email request for the shipping department to fulfill the order.

Create Recommendations Based on Customer Satisfaction Scores

This example lets a customer service or account rep base a Next Best Action recommendation on whether a customer has a high or low customer satisfaction (CSAT) score. For customers with a low CSAT, a rep can offer the customer a discount on their service contract renewal. For customers with a high CSAT score, the rep can offer a new product preview.

Preparation

To record customer satisfaction scores and use them to determine which recommendation to display, this example includes two custom fields. To follow along with the example, set up these two fields before you begin.

Contact object custom field:

- Field Label: CSAT score
- API Label: CSAT_score
- Field type: Number (length 2, decimal places 0)

Recommendation object custom field:

- Field Label: Category
- API Label: category_c
- Field type: Text (length 18)

To set up these Next Best Action recommendations:

1. Create action flows on page 759 for the high and low CSAT recommendations.
2. Create recommendation records on page 759 for the high and low CSAT recommendations.
3. Create a strategy flow on page 760 that determines how the recommendations are presented to the customer service or account rep.

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

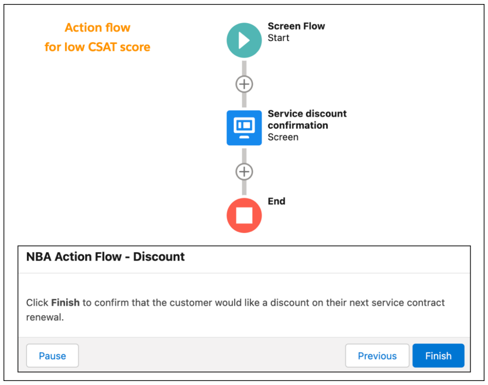
4. On the Contact record page, add the Next Best Strategy component on page 763 that displays the recommendations and executes the strategy.

Create Action Flows

Create two simple screen flows, one to execute an action for the low CSAT recommendation and one to execute an action for the high CSAT recommendation.

This example keeps things simple by displaying a different text message for each recommendation but not incorporating other automation. For a real-world application, you can add additional elements to implement the service contract discount and the new product preview. For an example of using an action flow to send an email request, see Offer a Gift Basket to Each of Your Accounts on page 753.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. Select **Start from Scratch**, and then click **Next**.
3. Select **Screen Flow**, and then click **Create**.
4. Add a Screen element to the flow.
5. Enter a label and API name.
6. Drag a **Display Text** component to the canvas.
7. Enter an API name for the component.
8. Add text for the high or low CSAT recommendation.



NBA Action Flow - Discount

Click **Finish** to confirm that the customer would like a discount on their next service contract renewal.

9. Click **Done**
10. Save the flow and name it `CSAT Action Flow - Discount` or `CSAT Action Flow - Product Preview`.
11. Activate the flow.
12. Repeat these steps to create the second action flow.

Create Recommendation Records

Create records for the low CSAT and high CSAT recommendations.

1. From the App Launcher (⠿), in the Quick Find box, enter `Recommendations`, and select **Recommendations**.
2. Click **New**.
3. Enter a name and description for the recommendation.

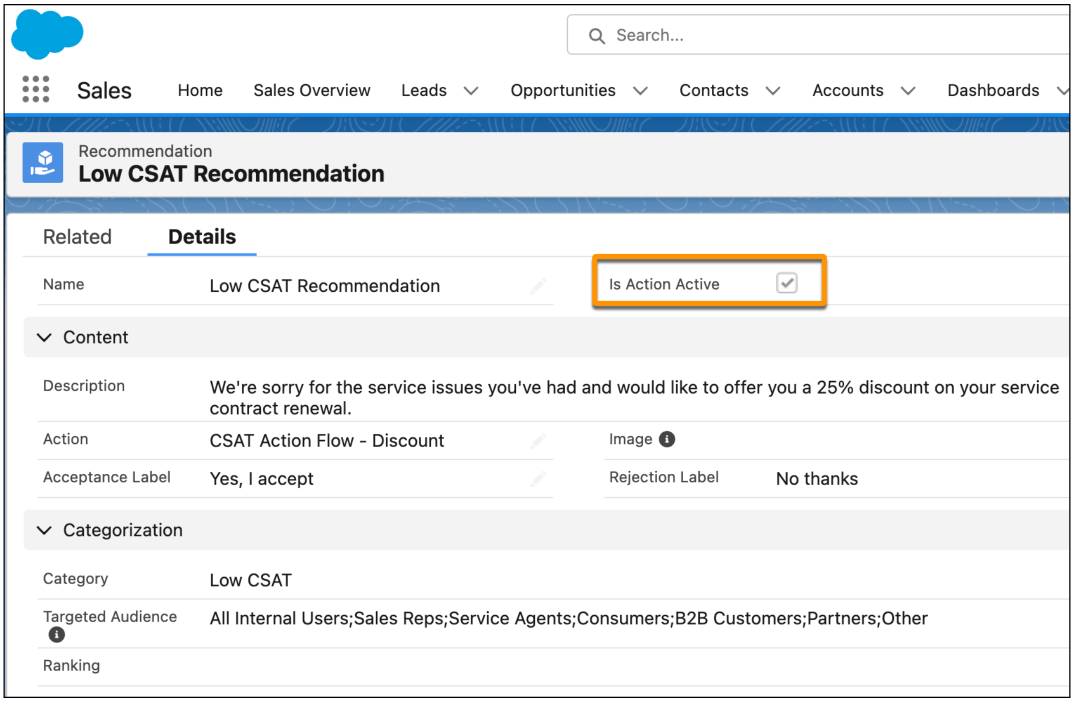
   The description appears in the Next Best Action component on the Lightning record page. Make the description specific to the particular recommendation (low CSAT or high CSAT).

4. For **Action**, select the low CSAT or high CSAT action flow.
5. To add an image (optional), click **Upload Image** and follow the instructions.

   For best results, use a 1000 px x 380 px image at 72 dpi or one with a similar ratio.

6. Enter text for the acceptance and rejection buttons.
7. Select the target audiences for the recommendation.
8. Click **Save**.

   The Is Action Active checkbox is automatically selected, which makes the recommendation available to Einstein Next Best Action.



Recommendation
Low CSAT Recommendation

Name: Low CSAT Recommendation

Is Action Active ☑

Content

Description: We're sorry for the service issues you've had and would like to offer you a 25% discount on your service contract renewal.

Action: CSAT Action Flow - Discount

Image

Acceptance Label: Yes, I accept

Rejection Label: No thanks

Categorization

Category: Low CSAT

Targeted Audience: All Internal Users;Sales Reps;Service Agents;Consumers;B2B Customers;Partners;Other

Ranking

9. Repeat these steps to create the second recommendation record.

Create a Recommendation Strategy Flow

The recommendation strategy flow specifies when and how the recommendations are presented on the Contact record page.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. Click **Use a Template**, and then click **Next**.
3. Click **Recommendation Strategy**, select a template, and then click **Create**.
4. Load the Contact records that you want to use for your recommendations by adding a Get Records element to the flow.
   a. Enter a label and API name.
   b. Select the **Contact** object.
   c. In the Filter section, add the condition `Id equals recordId`.
   d. Select the options to store all records and all fields.
   e. Click **Done**.
5. To accommodate different recommendations based on the customer's CSAT score, add a decision step after the Get Records step.

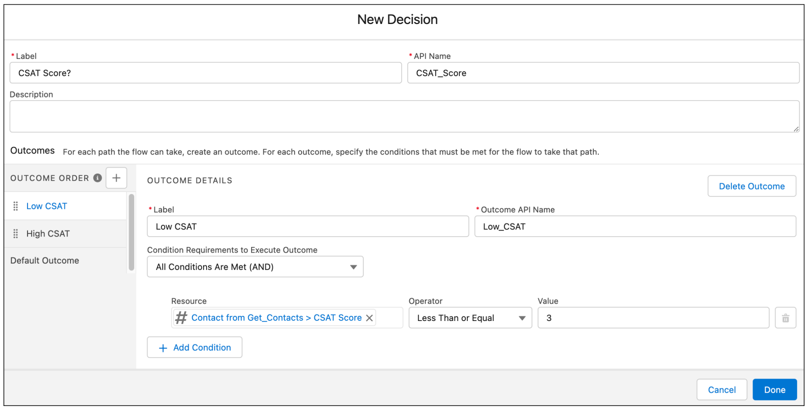
a. Enter the label `CSAT Score?` and the API name `CSAT_score`.

b. Create a `Low CSAT` outcome with the condition that the value of the CSAT Score field on the Contact record is 3 or lower.

c. Create a `High CSAT` outcome with the condition that the value of the CSAT Score field on the Contact record is 4 or higher.

d. Keep the Default outcome as-is for customers who don't have a CSAT score.

e. Click **Done**.



New Decision

* Label: CSAT Score?
* API Name: CSAT_Score
Description:

Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

* Label: Low CSAT
* Outcome API Name: Low_CSAT
Condition Requirements to Execute Outcome: All Conditions Are Met (AND)

Resource	Operator	Value
Contact from Get_Contacts > CSAT Score	Less Than or Equal	3

6. Bring in the appropriate recommendation for the low and high CSAT conditions by adding a Get Records element for each.

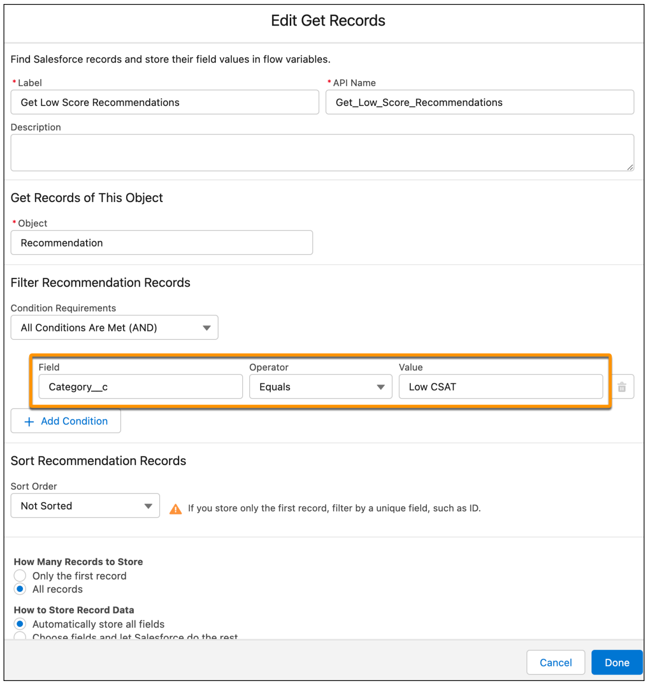
   a. Enter a label and API name.

   b. Select the **Recommendation** object.

   c. In the Filter section, add the appropriate condition by selecting the API name of the Category field in the Recommendation object and specifying the low or high condition.

   d. Select the options to store all records and all fields.

   e. Click **Done**.



Edit Get Records

Find Salesforce records and store their field values in flow variables.

* Label: Get Low Score Recommendations
* API Name: Get_Low_Score_Recommendations
Description:

Get Records of This Object

* Object: Recommendation

Filter Recommendation Records

Condition Requirements: All Conditions Are Met (AND)

Field	Operator	Value
Category__c	Equals	Low CSAT

Sort Recommendation Records

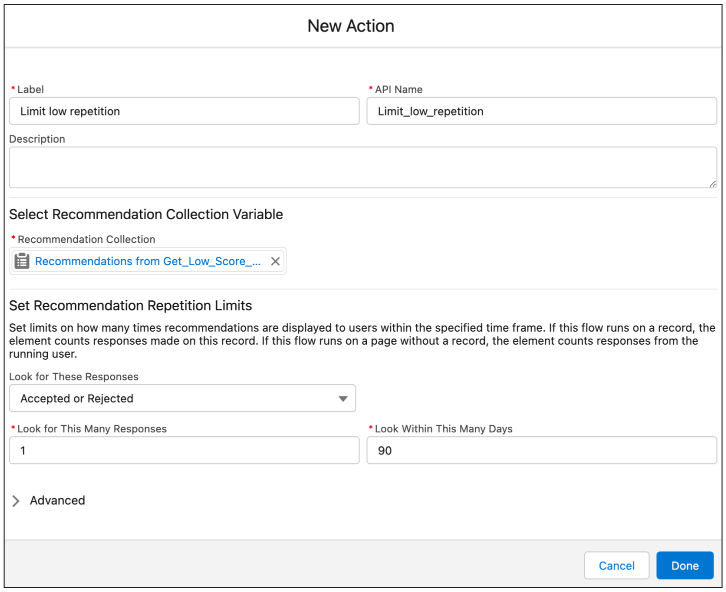
Sort Order: Not Sorted

If you store only the first record, filter by a unique field, such as ID.

How Many Records to Store
○ Only the first record
● All records

How to Store Record Data
● Automatically store all fields
○ Choose fields and let Salesforce do the rest

7. To show the recommendation only one time for each Account record and to assign the flow output, add a Limit Repetition element for the low and high score paths.
   a. Enter a label and API name.
   b. For Recommendation Collection, select the low score or high score recommendation.
   c. For Look for These Records, select **Accepted or Rejected**.
   d. For Look for This Many Messages, keep the default setting of `1`.
   e. To make the output from this path available to Next Best Action, click **Advanced**, select **Manually assign variables**, and then select **outputRecommendations**.
   f. Click **Done**.



New Action

*Label: Limit low repetition

*API Name: Limit_low_repetition

Description

Select Recommendation Collection Variable

*Recommendation Collection: Recommendations from Get_Low_Score_...

Set Recommendation Repetition Limits

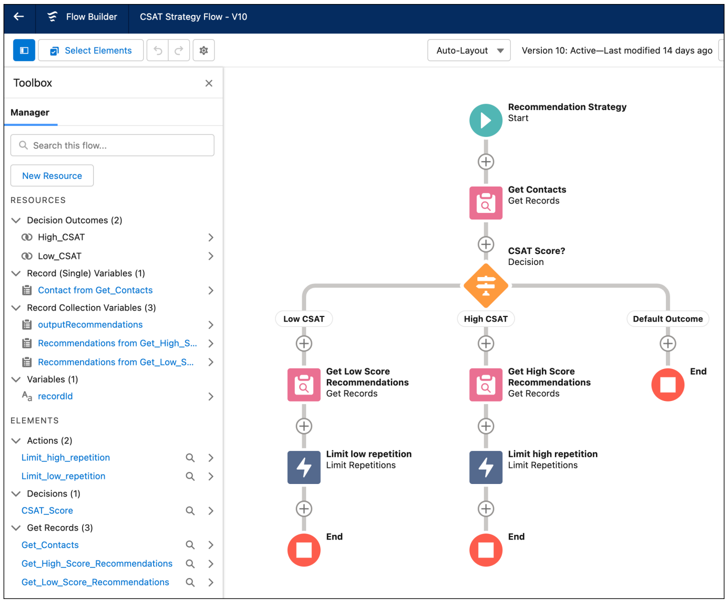
Set limits on how many times recommendations are displayed to users within the specified time frame. If this flow runs on a record, the element counts responses made on this record. If this flow runs on a page without a record, the element counts responses from the running user.

Look for These Responses: Accepted or Rejected

*Look for This Many Responses: 1

*Look Within This Many Days: 90

8. Save the flow and name it `CSAT Strategy Flow`.

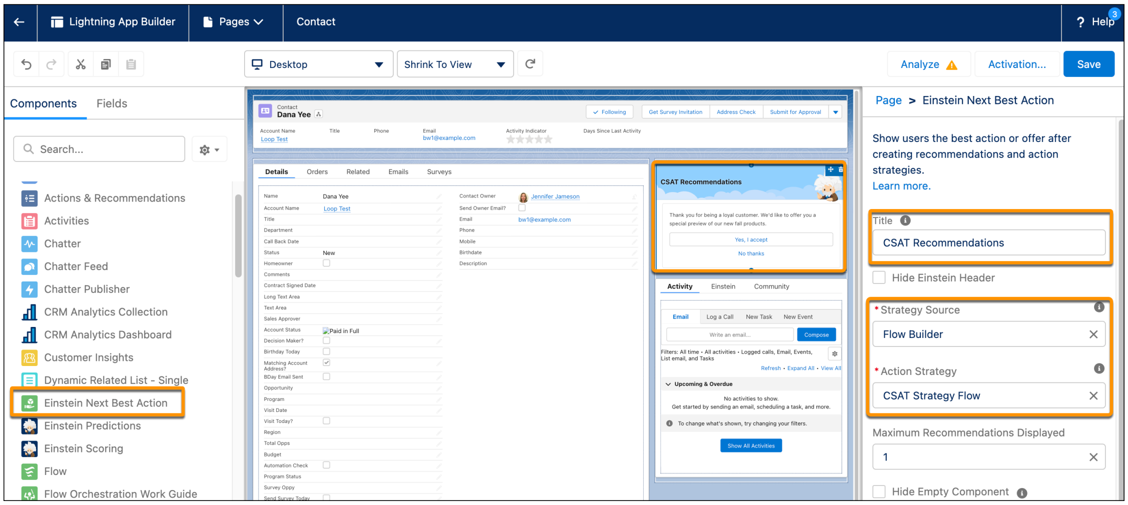


9. Activate the flow.
10. To return to the Flows page, click **Back**.

Display the Next Best Action Recommendations

To make the recommendations available to the customer service or account rep, display the Next Best Action component on the Contact record page.

1. Open a Contact record page.
2. Click the Setup icon (⚙), and select **Edit Page**.
3. Drag the Einstein Next Best Action component to the desired location on the page layout.
4. Add `CSAT Recommendations` as the component title.
5. For Strategy Source, select **Flow Builder** and then select the name of the recommendation strategy.



Show users the best action or offer after creating recommendations and action strategies. Learn more.

Title: CSAT Recommendations

□ Hide Einstein Header

* Strategy Source: Flow Builder

* Action Strategy: CSAT Strategy Flow

Maximum Recommendations Displayed: 1

□ Hide Empty Component

6. Save your changes.
7. Return to the Contact record and refresh the page.

Based on the contact's CSAT score, the correct recommendation is displayed. When the customer accepts the offer and the account rep clicks **Yes I Accept**, a form opens with the appropriate confirmation message.

Einstein Next Best Actions Considerations

Keep these considerations in mind when working with strategies and recommendations.

Einstein Next Best Action relies on flows, recommendations, strategies, and components, and has standard objects for reporting.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Flows

- All Recommendation objects reference a flow. If you don't have any flows, you can't surface a recommendation.
- Strategies only load recommendations with active flows.
- When a flow is executed via REST API, the flow runs in the context of the user who is authenticated via REST API. The running user's profile and permission sets determine the object

permissions and field-level access of the flow. We recommend that you create a profile and permission sets for users who run the flow.

Recommendations

- Consider adding a custom category field to the recommendation object and layout. A category field gives you more control when loading, sorting, and filtering recommendations and more options when creating flows.
- Create names, descriptions, acceptance labels, and rejection labels that are appropriate for your intended audience.
- Reusing a recommendation name creates a recommendation. It doesn’t overwrite an existing recommendation. Duplicated names can cause strategies to display duplicate recommendations to customers.
- All flows, both inactive and active, display in the Action dropdown list. After you save your recommendation, you can see if the flow is active.
- You can create a recommendation based on a flow that isn’t active, but no strategy loads it until the flow is activated.

Strategies

- All strategies require at least one recommendation.
- In Strategy Builder, you can load and filter the records of a Recommendation object. Or load and filter the records of any object, and convert them into recommendations at the end of the strategy using the Map element.
- Load elements require at least one criteria.
- Strategies only load recommendations that are based on active flows.
- The Limit Reoffer element in Strategy Builder lets you hide a recommendation from all users based on its responses. A recommendation is hidden if users respond more than a defined number of times within a defined number of days. For limit reoffers to work, recommendations must have a unique record ID. If you want to continue to test a recommendation as a flow-entry point, delete individual records from the Recommendations Reaction table with Rest API calls:

  ```
  GET /connect/recommendation-strategies/reactions
  { onBehalfOf: “005B00000018jK4IAI” }
  //Returns a list of reactions
  //For each result, if the reaction matches the strategyId of the strategy you’re testing:
  DELETE /connect/recommendation-strategies/reactions/${reactionId}
  ```

- Strategy Builder is available only in Lightning Experience.

Tracking and Reporting Reactions

- For strategies created in Flow Builder, create custom report types using the Recommendation Strategy Metrics and Recommendation Responses primary objects. For strategies created in Strategy Builder, create custom report types using the Recommendation Strategy Metrics and Recommendation Reactions primary objects.
- For reports created from the Recommendation Reactions primary object to correctly display the recommendation source name and ID for limit reoffers, recommendations must have a unique record ID.

```
Rights of ALBERT EINSTEIN are used with permission of The Hebrew University of Jerusalem.
 Represented exclusively by Greenlight.
```

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

Write a Strategy Builder Expression

Apex Reference Guide: NextBestAction Class

Einstein Next Best Action Entitlements

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Einstein Next Best Action has usage-based entitlements. All orgs receive a free monthly allowance of Next Best Action requests. If your usage exceeds your allowance of free monthly requests or any entitlements that you purchase, Salesforce contacts you to discuss additions to your contract. To track your usage, from Setup, navigate to **Company Information**.

Note: Next Best Action entitlement usage is based on a rolling 30-day period, beginning when the org is created. Entitlement usage listed on the Company Information page in Setup is based on the calendar month's usage, not the rolling 30-day usage.

Einstein Next Best Action is automatically enabled for the editions listed in the Required Editions table.

How To Get It	What's Included	What's Counted	Availability	Permission
Free org-wide requests	All orgs receive 5,000 Next Best Action strategy requests per month at no charge.	All Next Best Action and Einstein Recommendation Builder requests, including tests in Strategy Builder, display to customers, and repeat displays triggered by events.	Included in **Essentials**, **Professional**, **Enterprise**, **Performance**, and **Unlimited** Editions	None
Einstein Next Best Action Additional Requests SKU	Provides 10,000 additional Next Best Action requests each month on an org-wide basis. (Add-on)	All Next Best Action requests, including tests in Strategy Builder, display to customers, and repeat displays triggered by events.	For purchase in **Essentials**, **Professional**, **Enterprise**, **Performance**, and **Unlimited** Editions	None
Service Cloud Einstein SKU	Provides unlimited Next	Requests made by users with this	For purchase in **Enterprise**,	Unlimited Next Best Action

How To Get It	What's Included	What's Counted	Availability	Permission
	Best Action requests for a single user.	permission aren't counted against the monthly entitlement.	**Performance**, and **Unlimited** Editions	Strategy Executions
Lightning Platform Plus	Provides unlimited Next Best Action requests for a single user.	Requests made by users with this permission aren't counted against the monthly entitlement.	For purchase in **Enterprise**, **Performance**, and **Unlimited** Editions	Unlimited Next Best Action Strategy Executions

Next Best Action Request

A *request* is a call to the Next Best Action engine that causes a strategy to run and return recommendations.

Next Best Action Request

A *request* is a call to the Next Best Action engine that causes a strategy to run and return recommendations.

Each time a page with an Einstein Next Best Action component is loaded or refreshed in a browser, Salesforce generates a new request. For example, when a case status changes from New to In Progress, the data change on the page triggers a refresh. This action also applies to the Actions & Recommendations and Suggested Actions components.

Requests are also made when:

- A field is updated on a record detail page that includes the Next Best Action component.
- A user enters data in the Subject or Description field of a site contact support page that includes the Next Best Action component.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Another way to make a request is to call a Next Best Action REST API resource from your own web app. You can also call Next Best Action REST API resources from an iOS or an Android app. The app can make requests in response to a custom UI and return recommendations.

Paying customers can see the number of requests their org has made by navigating from Setup to **Company Information**, **Usage-based Entitlements**, **Maximum Next Best Action Requests available**.

SEE ALSO:

Display Recommendations

Einstein Next Best Action Entitlements

Create Recommendations

Create offers or actions to recommend to users using Einstein Next Best Action. Recommendations are standard Salesforce records, similar to accounts and contacts, that are processed by strategies and associated with flows. Strategies determine which recommendation records are surfaced using business rules, predictive models, and other data sources. The result of this process is context-specific recommendations that you present to your users.

Note:

- Salesforce has both a Recommendation object for Einstein Next Best Action (that's this page) and a Recommendation component for Experience Builder sites. The Recommendation component isn't related to Next Best Action.
- If you don't see Recommendations in the App Launcher, in Setup, select Default On in the Recommendations tab settings for your user profile or permission set.
- You can load and filter the records of a Recommendation object. Or load and filter the records of any object, and convert them into recommendations at the end of a strategy using the Map element.

Before creating recommendations, create the action flow that runs when a customer accepts the recommendation. For examples of action flows for Next Best Action, see Einstein Next Best Action Examples on page 752.

1. In the Recommendations tab, click **New Recommendation**.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

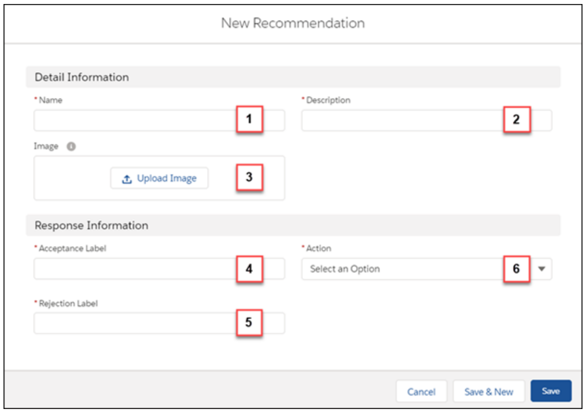
USER PERMISSIONS

To create or manage recommendations:

- Modify All Data

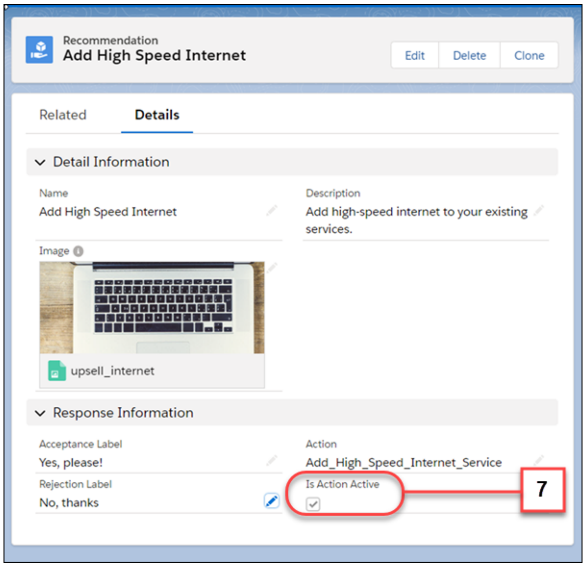
  OR

  Manage Next Best Action Recommendations



2. Enter a friendly name (1) and a brief description (2) for your recommendation. The description appears on the recommendation that is surfaced to users.
3. Optionally, click to upload an image (3) that you can display as a header for your recommendation. For best results, use a 1000 by 380 pixel image at 72 DPI, or an image with a similar ratio. You can choose whether the image displays using component properties. After it's uploaded, a thumbnail of your image displays on the Recommendations page. Customers can see the full image as a header for your recommendation in either the Lightning App Builder or Experience Builder component.
4. Enter an acceptance label (4) and a rejection label (5) for the buttons that customers click to, respectively, accept and reject the recommendation.
5. Create a flow. When a user accepts your recommendation, they're taken to the flow specified in Action (6).

6. Choose the flow that runs when a customer accepts the recommendation (6) and click **Save**. You can also choose a flow that runs when a customer accepts or rejects the recommendation. The Action list displays both active and inactive flows. Choosing a flow that isn't active hides the recommendation. When you've saved your recommendation, you can see if the flow is active from **Is Action Active** (7).



Recommendation
Add High Speed Internet

Detail Information

Name
Add High Speed Internet

Description
Add high-speed internet to your existing services.

Image
upsell_internet

Response Information

Acceptance Label
Yes, please!

Action
Add_High_Speed_Internet_Service

Rejection Label
No, thanks

Is Action Active
☑

7. Create a recommendation strategy in Strategy Builder that determines how your recommendations surface.

8. Optionally add a custom Category field to the Recommendation object and the Recommendation Layout. Adding a custom Category field can simplify loading, filtering, and sorting recommendations in Strategy Builder.

Recommendation Fields

Recommendations are suggested actions that users see and interact with through Einstein Next Best Action strategies. When creating a recommendation, use these fields to define its look and feel.

Launch a Flow When a Recommendation Is Accepted

Each recommendation is associated with a single flow. By default, Next Best Action launches a flow when a user accepts a recommendation. The flow then performs an action, such as updating a case or sending an email.

Launch a Flow When a Recommendation Is Accepted or Rejected

Each recommendation is associated with a single flow. By default, Next Best Action launches a flow when a user accepts a recommendation. The flow then performs an action, such as updating a case or sending an email. But you can also launch a flow when a user rejects a recommendation, which gives you more flexibility. For example, a flow could run an automated process, write to another system, or create a reminder email when a recommendation is rejected.

Add a Limit Repetitions Element to a Next Best Action Flow

You can add a Limit Repetitions element to your Recommendation Strategy flow to limit the number of times that the same recommendation or offer appears on the same record or for the same user during a time period.

SEE ALSO:

Build a Flow

Strategy Builder Strategies

Display Recommendations

View and Edit Tab Settings in Permission Sets and Profiles

Get Started with Einstein Next Best Action

Connect REST API Developer Guide: Next Best Action Resources

Recommendation Fields

Recommendations are suggested actions that users see and interact with through Einstein Next Best Action strategies. When creating a recommendation, use these fields to define its look and feel.

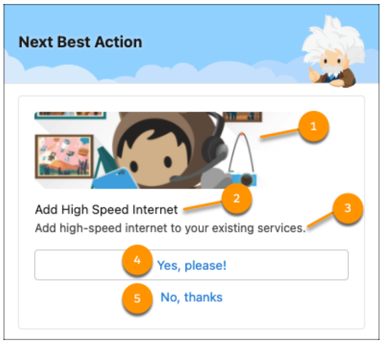
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

You can use these methods to create recommendations.

- Assemble recommendations as needed in Flow Builder or Strategy Builder.
- Create recommendations as standard Salesforce records, similar to accounts and contacts, in the Recommendation object. You can create recommendation records on the Recommendations tab in the App Launcher.
- Generate recommendations automatically through AI with Einstein Recommendation Builder.



- Image (1)—The image that is shown in the recommendation. To display this image with the Einstein Next Best Action Lightning page component, select `Show Image` when configuring the Lightning page component.
- Name (2)—The header text at the top of the recommendation. To display this text with the Einstein Next Best Action Lightning page component, select `Show Title` when configuring the Lightning page component.
- Description (3)—Additional descriptive text displayed in the recommendation. To display this text with the Einstein Next Best Action Lightning page component, select `Show Description` when configuring the Lightning page component.
- Acceptance Label (4)—The text of the button that accepts the recommendation. This option is always displayed.
- Rejection Label (5)—The text of the button that rejects the recommendation. To display this option with the Einstein Next Best Action Lightning page component, select `Show Reject Option` when configuring the Lightning page component.

Use these fields to define how the recommendation runs.

Action

The flow that runs when a user selects the Accept option. To run this flow when the user accepts or rejects the recommendation, select `Launch Flow on Rejection` when configuring the Einstein Next Best Action Lightning page component. If the referenced flow is inactive, invalid, or has an unsupported Flow Type, the recommendation isn't displayed to users. The supported flow types are screen flows and autolaunched flows.

SEE ALSO:

Create Recommendations

Get Started with Einstein Next Best Action

Launch a Flow When a Recommendation Is Accepted

Each recommendation is associated with a single flow. By default, Next Best Action launches a flow when a user accepts a recommendation. The flow then performs an action, such as updating a case or sending an email.

For example, on a case, display a recommendation to the service agent to upsell a premium service to the customer. When the agent accepts the recommendation, an autolaunched flow updates the case and the customer's order history and sends a receipt via email.

Or say that you have an autolaunched flow that sends a templated marketing campaign email to a customer. Your service agents have to determine whether your customers are eligible for this campaign. Doing so involves several clicks and complex calculations. Instead use Next Best Action to check the customer's eligibility and prompt the agent to accept the recommendation and launch the flow.

1. In Flow Builder, configure a flow that's associated with a recommendation. Be sure to activate the flow because Next Best Action can't call an inactive flow from a recommendation.
2. Add a flow action.
3. To add the flow, edit the recommendation.

SEE ALSO:

Launch a Flow When a Recommendation Is Accepted or Rejected

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and save Lightning pages in the Lightning App Builder:

- Customize Application

To run a recommendation strategy on a Lightning record page:

- Run Flows

  OR

  Flow User field enabled on the user detail page

Launch a Flow When a Recommendation Is Accepted or Rejected

Each recommendation is associated with a single flow. By default, Next Best Action launches a flow when a user accepts a recommendation. The flow then performs an action, such as updating a case or sending an email. But you can also launch a flow when a user rejects a recommendation, which gives you more flexibility. For example, a flow could run an automated process, write to another system, or create a reminder email when a recommendation is rejected.

For example, at a telecommunications company, the admin configures the Next Best Action component to display recommendations to its customer service representatives (CSRs). When a CSR accepts a recommendation for a customer who wants to purchase a discounted service, a flow is launched to calculate the discount. The admin analyzes the reactions to the recommendation, and is confused about why the CSRs are rejecting it. To help get answers, the admin uses Next Best Action to launch a questionnaire flow every time the recommendation is rejected.

This feature is available for:

- The Einstein Next Best Action component used with Lightning record pages
- The Suggested Actions component used in Experience Builder
- The Actions and Recommendations component used with Lightning console apps

To assign a flow that runs when a customer accepts or rejects the recommendation, create an input variable in the flow to accept the `isRecommendationAccepted` value. Then add a Decision element to the flow that's based on that value.

1. In Flow Builder, configure a flow that's associated with a recommendation. Be sure to activate the flow because Next Best Action can't call an inactive flow from a recommendation.
2. Create the Boolean `isRecommendationAccepted` input variable.
3. Create a Decision element and use the `isRecommendationAccepted` variable in your outcome conditions.
4. Create a decision outcome for what the flow does when the recommendation is accepted.
5. Create a decision outcome for what the flow does when the recommendation is rejected.

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

To create and save Lightning pages in the Lightning App Builder:

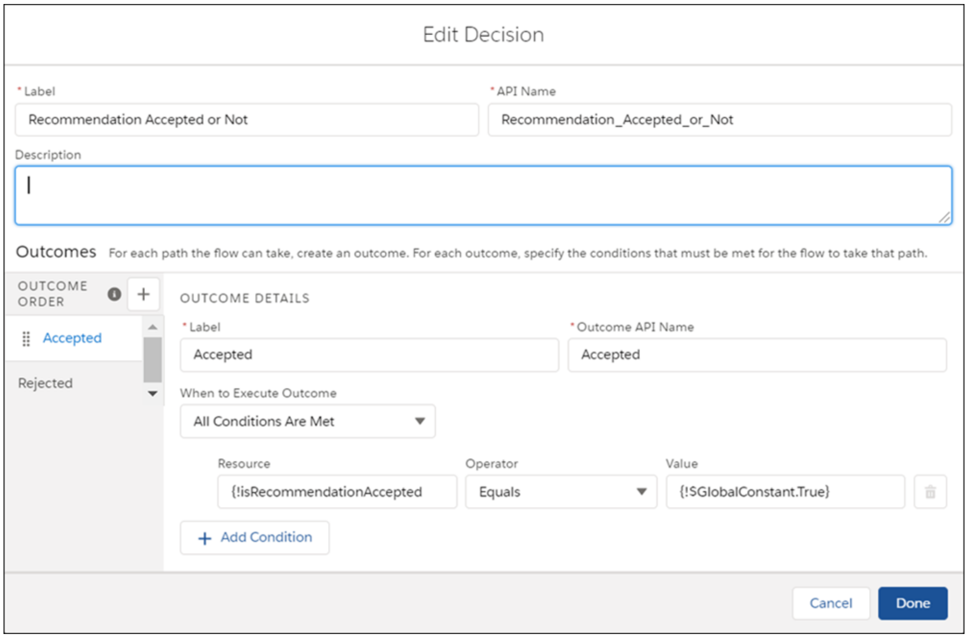
- Customize Application

To run a recommendation strategy on a Lightning record page:

- Run Flows

  OR

  Flow User field enabled on the user detail page



Edit Decision

* Label: Recommendation Accepted or Not

* API Name: Recommendation_Accepted_or_Not

Description:

Outcomes For each path the flow can take, create an outcome. For each outcome, specify the conditions that must be met for the flow to take that path.

OUTCOME ORDER

- Accepted
- Rejected

OUTCOME DETAILS

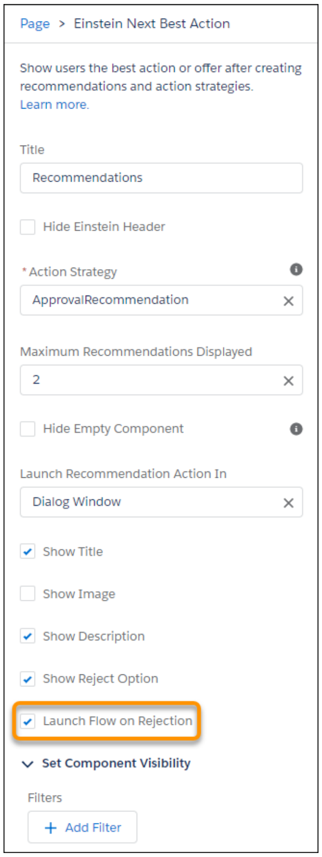
* Label: Accepted

* Outcome API Name: Accepted

When to Execute Outcome: All Conditions Are Met

Resource	Operator	Value
{!isRecommendationAccepted	Equals	{!$GlobalConstant.True}

6. Add any additional flow elements to handle each outcome path.
7. Add a flow action.
8. To add the flow, edit the recommendation.
9. When you add the Next Best Action component to a Lightning record page, select **Launch Flow on Rejection**.



Page > Einstein Next Best Action

Show users the best action or offer after creating recommendations and action strategies. Learn more.

Title: Recommendations

□ Hide Einstein Header

* Action Strategy: ApprovalRecommendation

Maximum Recommendations Displayed: 2

□ Hide Empty Component

Launch Recommendation Action In: Dialog Window

☑ Show Title

□ Show Image

☑ Show Description

☑ Show Reject Option

☑ Launch Flow on Rejection

Set Component Visibility

Filters

SEE ALSO:

- Launch a Flow When a Recommendation Is Accepted
- Flow Resource: Variable
- Flow Element: Decision
- Einstein Next Best Action Component

Add a Limit Repetitions Element to a Next Best Action Flow

You can add a Limit Repetitions element to your Recommendation Strategy flow to limit the number of times that the same recommendation or offer appears on the same record or for the same user during a time period.

- You must have a collection of recommendations that has a valid value in the ID or RecommendationKey fields. The RecommendationKey value must be a database ID or have the syntax *`DYNAMIC_<custom id>`*.
- If you include an Assignment element, from Actions, choose **Output from limit**. Or you can skip this step and add the output from the Limit Repetitions element.

1. From Setup, in the Quick Find box, enter *`Flows`*, and then select **Flows**.
2. Open or create a Recommendation Strategy.
3. After the collection of recommendations, add a Limit Repetitions element.
4. Enter a label and an API Name.
5. Add a description.
6. Search for and select the Recommendation Collection that you want to filter.
7. Select the responses that you want, and then enter the number of responses and days as whole numbers.

   Look Within This Many Days is based on days, not hours. If the number of days is set to 1 for an accepted response, and the user accepts the recommendation at any time on Monday, the recommendation doesn't display again until the start of Wednesday. So a one-day time period could be as few as 25 hours in duration or as many as 48 hours.
8. If you didn't include an Assignment element, you can search for and select the collection that includes the limit repetition output.
   a. In Advanced, select **Manually assign variables**.
   b. From the Store Output Variables field, search for and select the output variable.
9. Click **Done**.
10. Save your work.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

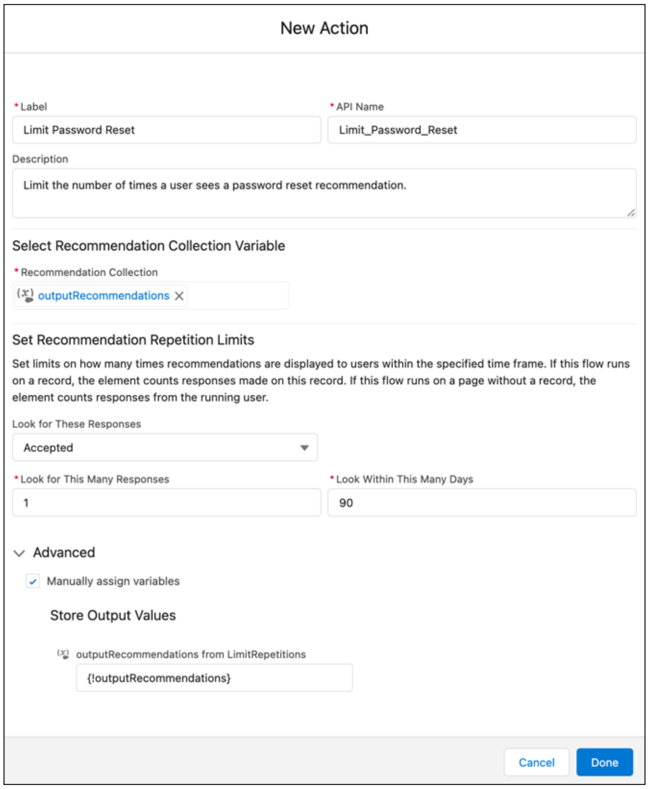
USER PERMISSIONS

To create or manage recommendations:

- Modify All Data

  OR

  Manage Next Best Action Recommendations



New Action

*Label: Limit Password Reset

*API Name: Limit_Password_Reset

Description: Limit the number of times a user sees a password reset recommendation.

Select Recommendation Collection Variable

*Recommendation Collection: outputRecommendations

Set Recommendation Repetition Limits

Set limits on how many times recommendations are displayed to users within the specified time frame. If this flow runs on a record, the element counts responses made on this record. If this flow runs on a page without a record, the element counts responses from the running user.

Look for These Responses: Accepted

*Look for This Many Responses: 1

*Look Within This Many Days: 90

Advanced

☑ Manually assign variables

Store Output Values

outputRecommendations from LimitRepetitions: {!outputRecommendations}

Example:

- If you want one accepted response over 90 days, such as a password reset recommendation, and the user accepts one time over 90 days, they don't see the message again for 90 days. But if the user rejects the recommendation, they see the message every time they reload the page until they accept it.
- If you want two accepted or rejected responses over 1 day, and a user accepts or rejects the recommendation only one time every day, they still see the recommendation.
- If you want two accepted or rejected responses over 1 day, and a user accepts or rejects the recommendation twice on day one, they don't see the recommendation on day two. They see the recommendation again on day three.

If you add an Assignment element after the Limit Repetitions element and change the label for accept or reject, you must update the limit repetitions output.

SEE ALSO:

Create Recommendations

Display Recommendations

Building a Strategy

A strategy determines when and how to present an Einstein Next Best Action recommendation on a Salesforce Lightning record page. For example, if you want to offer a discount to a subset of customers, create a strategy that collects the appropriate customer records and identifies the discount option to present. To create a strategy, you can use Flow Builder (recommended) or Strategy Builder.

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: When possible, we recommend building a strategy in Flow Builder using the Recommendation Strategy flow type.

Why Choose Flow Builder Instead of Strategy Builder?

Flow builder is a unified, feature-rich Salesforce tool for building business process automations and is the home for all future flow automation features and enhancements. Strategy Builder is a legacy tool and no updates are planned for it.

Strategy Builder Strategies

You can create strategies for Einstein Next Best Action using Strategy Builder or Flow Builder. Flow Builder is the recommended method.

Flow Builder Strategies

A Flow Builder strategy specifies business logic and generates output for an Einstein Next Best Action component on a Salesforce Lightning record page.

EDITIONS

Available in: Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

In a Flow Builder strategy, you generate recommendations in either of the following ways:

- Use predefined recommendations created in the Recommendations object on page 777. With this method, you create recommendations individually in the Recommendation object and then use them in one or more Next Best Action components. This method is best if you're creating a small number of recommendations.
- Create recommendations on the fly without using separate recommendation records on page 778. With this method, you create multiple recommendations dynamically in the strategy flow. This method is best if you're creating a large number of recommendations. For example, if you have an extensive product list, you can create a different upsell recommendation for each product in the list.

SEE ALSO:

Add and Edit Elements

Flow Element: Get Records

Flow Element: Assignment

Flow Element: Recommendation Assignment

Build a Strategy Flow Using Predefined Recommendations

Build a strategy flow based on predefined recommendations. This method works best if you have a small number of recommendations and want to make them available to multiple Einstein Next Best Action components. For example, you can create a recommendation that offers a discount to a customer. You can then use the same recommendation when creating a strategy for birthday discounts and for new customer discounts.

Before building your strategy flow, create recommendations in the Recommendations object on page 767.

Note: If you want to create a large number of recommendations dynamically at one time, you can build a strategy flow with on-the-fly recommendations on page 778 without creating separate records in the Recommendation object.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. On the **Alt + Templates** tab, select the **Recommendation Strategy** flow type, and click **Create**.
3. Load the records you want to use for your recommendation by adding a Get Records element to the flow.
   a. Enter a label and API name.
   b. Select the object to use for the recommendations, such as the Accounts, Cases, or Contacts object.
   c. In the Filter section, add conditions to limit which records from the object are used in your strategy.
4. Bring a predefined recommendation into the strategy by adding a Get Records element.
   a. Enter a label and API name.
   b. Select the **Recommendations** object.
   c. In the Filter section, use conditions to specify the recommendation that you want to use.
5. Add other flow elements as needed to define the strategy.
6. Make your recommendation available for use in an Einstein Next Best Action component.
   a. Add an Assignment element.
   b. For Variable, select **outputRecommendations**.
   c. For operator, select **Equals**.
   d. For Value, select the predefined recommendation.
7. Save your flow.
8. Activate your flow.

You're now ready to add a Next Best Action component to a Lightning record page. on page 807

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Build Strategy Using On-the-Fly Recommendations

Build a strategy flow with multiple recommendations that you create dynamically in bulk. For example, you can create a strategy that offers a different upsell recommendation for each product in your product list. With this method, you create recommendations directly in the strategy flow without using separate Recommendation records.

Note: If you want to reuse recommendations in multiple Einstein Next Best Action components, use pre-defined recommendations created in the Recommendations object on page 777.

1. From Setup, in the Quick Find box, enter `Flows`, select **Flows**, and then click **New Flow**.
2. On the **Alt + Templates** tab, select the **Recommendation Strategy** flow type, and click **Create**.
3. Load the records you want to use for your recommendations by adding a Get Records element to the flow.
   a. Enter a label and API name.
   b. Select the object to use for the recommendations, such as the Product object.
   c. In the Filter section, add conditions to limit which records from the object are used in your strategy.
4. Add a Recommendation Assignment element.
   a. Enter a label and API name.
   b. For Record Collection Variable, select the variable that you generated with Get Records.

      When you select the variable, the target fields are populated automatically.
   c. Set values for the target fields:
      - AcceptanceLabel—Button label to accept the offer.
      - RejectionLabel—Button label to reject the offer.
      - ActionFlow—API name of the flow that performs an action when the offer is accepted or rejected.
      - Description—Text that appears above the buttons in the Next Best Action component.
5. Make your recommendation available for use in an Einstein Next Best Action component.
   a. Add an Assignment element.
   b. For Variable, select **outputRecommendations**.
   c. For operator, select **Equals**.
   d. For Value, select the recommendation from the Recommendation Assignment step.
6. Save your flow.
7. Activate your flow.

You're now ready to add a Next Best Action component to a Lightning record page on page 807.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To open, edit, or create a flow in Flow Builder:

- Manage Flow

Strategy Builder Strategies

You can create strategies for Einstein Next Best Action using Strategy Builder or Flow Builder. Flow Builder is the recommended method.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Tour the Strategy Builder Interface

Before you start building your strategy, learn about the primary pieces of Strategy Builder and how they work together.

Create a Strategy with Strategy Builder

Once you've created flows and recommendation records, use Strategy Builder to funnel the correct recommendations to your users at the right time.

Manage Strategy Builder Action Strategies

Test, troubleshoot, and create strategies using Strategy Builder management tools.

Strategy Builder Elements

Use this page to quickly access a list of Strategy Builder elements and learn how they work together to create unique strategies.

Tour the Strategy Builder Interface

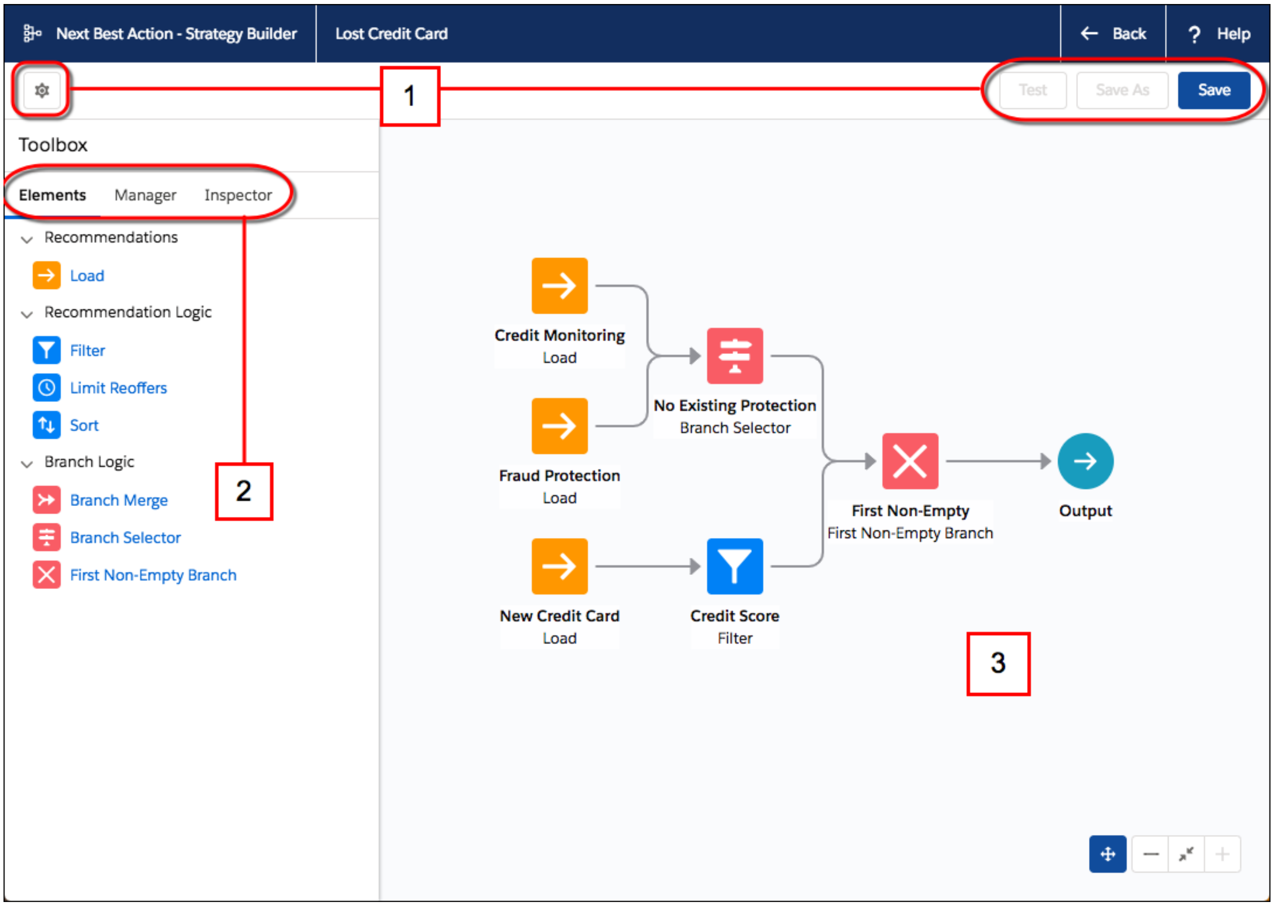
Before you start building your strategy, learn about the primary pieces of Strategy Builder and how they work together.

Find Strategy Builder in Setup by typing `Strategies` or `Next Best Action` in the Quick Find box. Select **Next Best Action**.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Button Bar (1)

Manage your strategies with basic functions like Test and Save.

- **Test** runs the most recently saved version of your strategy and displays the recommendations that are surfaced for your users. Testing your strategy allows you to determine if there are errors that must be fixed and confirms the recommendations that your users see.
- **Save** your strategies before you test them and before you leave Strategy Builder so you don't lose your work.
- **Save As** allows you to duplicate a strategy and your currently saved work.

Elements, Manager, and Inspector Tabs (2)

Use the Toolbox to create the substance of your strategy. Add elements, connect external sources, and troubleshoot errors in your strategy.

- From the **Elements** tab, drag new elements onto the canvas and create the building blocks of your strategy.
- From the **Manager** tab, add new connections from external sources or other Salesforce products.
- Use the **Inspector** tab to isolate specific elements and troubleshoot errors that appear during testing.

Canvas (3)

The canvas is a visual representation of your strategy. From here, you can rearrange elements and see how your recommendations are flowing from one branch to the next and finally into the output.

SEE ALSO:

Create a Strategy with Strategy Builder

Manage Strategy Builder Action Strategies

Inspect Strategy Builder Element Results

Create a Strategy with Strategy Builder

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:

- Modify All Data

  OR

  Manage Next Best Action Strategies

To run an action strategy:

- Run Flows

  OR

  Flow User field enabled on the user detail page

Once you've created flows and recommendation records, use Strategy Builder to funnel the correct recommendations to your users at the right time.

Before you start creating strategies, make sure that you create flows and recommendation records that you can use in your strategy.

1. Open Strategy Builder. From Setup, enter `Strategies` or `Next Best Action` in the Quick Find box, select **Next Best Action**, and click **New Strategy**.
2. Give your strategy a name and a description.
3. Select a context object from **Object Where Recommendations Display**.

   Note: The object that you choose here provides the context for your entire strategy.

   For example, if you plan to use this strategy on Case pages, select Case. When the strategy executes and resolves your expressions, the Next Best Action engine interprets the incoming recordId as a case object. The engine has to know to what type of object the pages belong to resolve expressions correctly. Linking your strategy to a specific object also enables Strategy Builder to provide intelligent assistance in other areas, such as the Test feature.

4. Drag the appropriate elements onto the canvas.

   Note: It's best to start by adding a Load element, as loading recommendations is the first step in any strategy.

5. Order your elements to make sure that recommendations are flowing through the correct branches.

   Note: Elements are divided into two main categories: Recommendation Logic and Branch Logic. Recommendation Logic elements act directly on the recommendations flowing into the element by filtering, sorting and limiting. Branch Logic elements act as gates, using context information, such as the recordId of the page the user is viewing, to decide which sets of recommendations to allow.

6. Save any changes to your strategy.
7. To make sure it's working as expected, test your strategy.

   Note: If your strategy isn't running properly or you see an unexpected error, try using the **Inspector** tab to find the problem.

8. Display your strategy using the Suggested Actions component in Experience Builder or the Einstein Next Best Action component in Lightning App Builder.

   Write a Strategy Builder Expression

   Create unique expressions using logic from the Salesforce expression builder to filter recommendations, select or deselect branches, and determine which recommendations are available for consideration in a strategy.

   Create a Strategy Builder Action Strategy Connection

   Use Apex actions to integrate external data sources and information from your Salesforce org into your strategies.

   Create a Custom Notification Flow for Next Best Action

   Create a trigger in Process Builder to receive direct notifications about errors occurring in your strategies. Launch a flow to send error information to your desired targets.

   Create, Package, and Distribute a Strategy Builder Template

   Enterprise developers can create and package strategy templates from Developer Edition orgs for use in multiple Salesforce orgs. Independent software vendors can also publish templates on AppExchange for distribution to their subscribers. Strategies not marked as templates in managed packages have intellectual property (IP) protection and can’t be edited or cloned. IP protection safeguards proprietary information in your strategies.

SEE ALSO:

Test Strategy Builder Action Strategies

Inspect Strategy Builder Element Results

Create a Strategy with Strategy Builder

Build a Flow

Write a Strategy Builder Expression

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Create unique expressions using logic from the Salesforce expression builder to filter recommendations, select or deselect branches, and determine which recommendations are available for consideration in a strategy.

Strategy Builder expressions, found on the Filter and Branch Selector elements, use standard Salesforce formula functions. To learn more about creating formulas in Salesforce, see Formula Operators and Functions by Context.

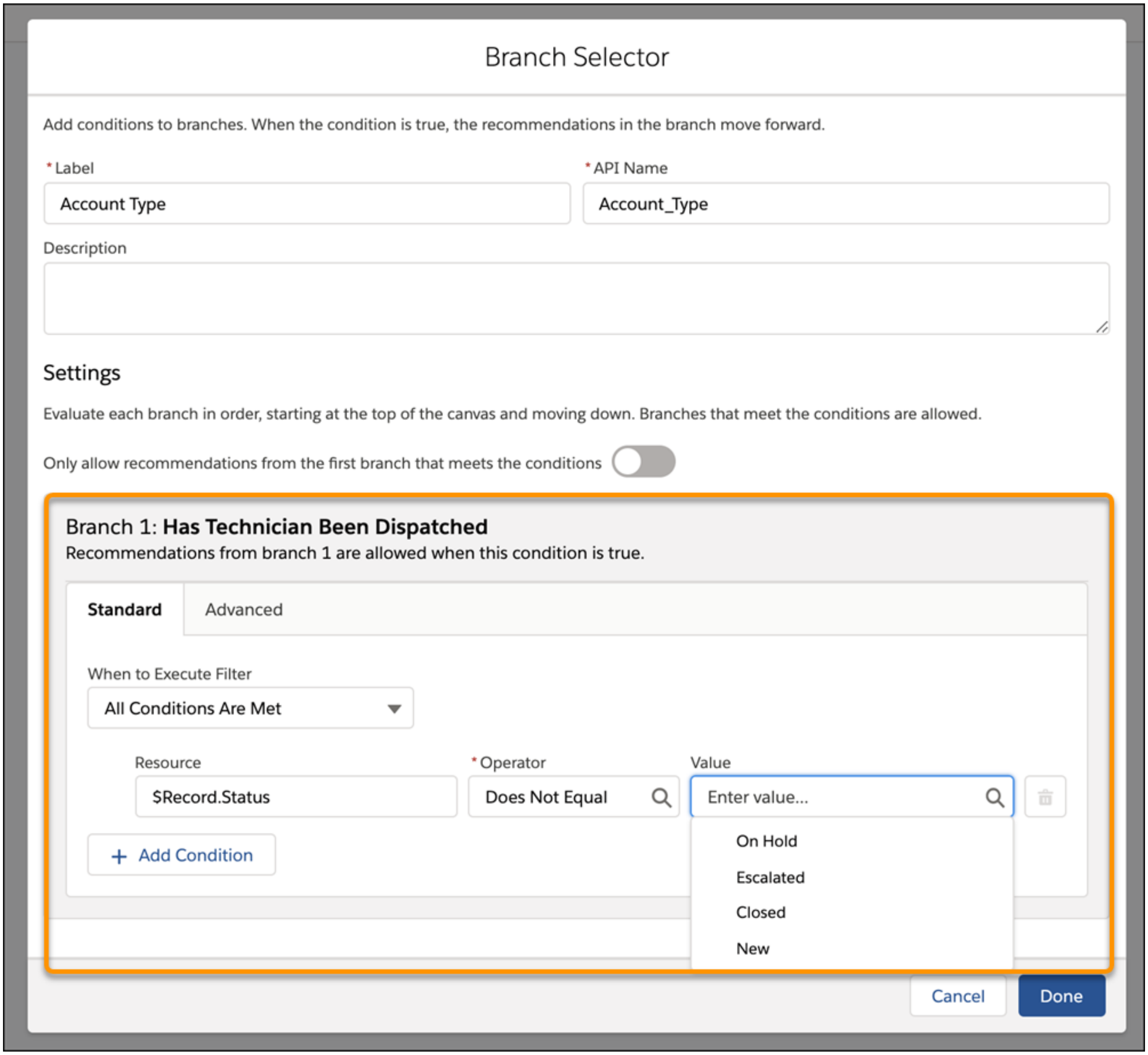
Strategies are designed to work with a particular object like Case or Contact. Strategy Builder elements use `$Record` as a placeholder for the actual record that gets passed in when a strategy runs.

1. Select the element you need for your strategy: **Filter** or **Branch Selector**.
2. Enter your expression. You can build expressions in two different modes: standard and advanced. Standard is declarative: search and select to build your formula. Use advanced mode for more complex expressions, when a given operator is unavailable in standard mode, or when you use concatenation.
3. In standard mode, set up conditions. At run time, the conditions are evaluated in the order you specify.

Column Header	Description
`Resource`	Recommendation resource whose value you want to evaluate, such as acceptance or rejection label, action, ID, name. For example, a strategy is associated with a Case. Your resource can be `$Record.Account.Type` or `$Record.Account.Contact.Name`.

Column Header	Description
`Operator`	Select an appropriate operator for that resource, for example `Equals`, `Does Not Equal`, `Starts With`, `Contains`, `Less Than Or Equal To`, and `Is Blank`. The available operators depend on the data types associated with that resource. Data types include text, number, Boolean, or picklist.
`Value`	Options: - Select a value that's appropriate for the recommendation resource and the operator. For example, if you enter `$Record.Status` as the resource and `Does Not Equal` as the operator, available values are `On Hold`, `Escalated`, `Closed`, and `New`. - Manually enter a literal value. `Resource` and `Value` in the same row must have compatible data types. When you add or subtract a number from a date value, the date adjusts in days, not hours.

Option	Behavior for Decision Outcomes
`All Conditions Are Met`	If one of the conditions is false, the recommendation evaluates the next outcome's conditions.
`Any Condition Is Met`	If one of the conditions is true, the recommendation immediately takes this outcome's path.



Branch Selector

Add conditions to branches. When the condition is true, the recommendations in the branch move forward.

*Label: Account Type

*API Name: Account_Type

Description:

Settings

Evaluate each branch in order, starting at the top of the canvas and moving down. Branches that meet the conditions are allowed.

Only allow recommendations from the first branch that meets the conditions

Branch 1: **Has Technician Been Dispatched**

Recommendations from branch 1 are allowed when this condition is true.

When to Execute Filter: All Conditions Are Met

Resource	*Operator	Value
$Record.Status	Does Not Equal	Enter value...

For example, say you create the custom field `Has_Mobile_Service__c` on the contact record. If you use `$Record.Contact.Has_Mobile_Service__c = false` in a Strategy Builder expression, and you're working with a case record, the recordID provided with the request replaces `$Record` when the expression resolves. The recordID replaces `$Record` because case records have a lookup relationship with contacts.

- Reference the context object in your formula using the *`$Record`* function.

For example, *`ISPICKVAL($Record.Account.Tier__c, 'Premium')`*

Note: The Context object is the object where you plan to surface your recommendations. Choose the Context object, or change it, by editing your strategy and choosing an object under **Object Where Recommendations Display**.

- Reference fields from the Recommendation object using the plain text label name. This option is available only in Filter and Load elements, not Branch Selector elements.

For example, `AcceptanceLabel = ='Yes, please'`

- Access fields returned from external connections using *$nameOfExternalConnection.dataFromExternalConnection* syntax. Manage your external connections through the **Manage** tab in the Toolbox.

For example, `$GetCreditScoreContext.output >= 760`

- Use *$Request* to access information the user types into forms and use that information to request specific recommendations. This option is available only on the Search and Contact Support pages in Experience Builder sites.

For example, `CONTAINS($Request.search, 'paperless billing') || CONTAINS($Request.search, 'order checks') || CONTAINS($Request.search, 'new address')`

For multi-select picklist fields, enter values like `Includes ($Record.CarType__c, 'Audi,' 'BMW')`

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

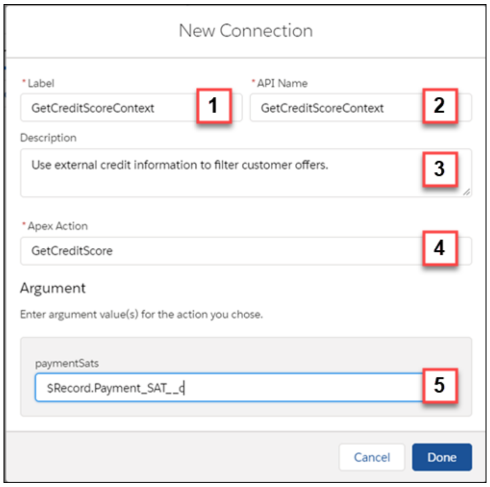
Formula Operators and Functions by Context

Create a Strategy Builder Action Strategy Connection

Use Apex actions to integrate external data sources and information from your Salesforce org into your strategies.

Use Apex invocable actions to pull sources of data into your strategy.

1. In Strategy Builder, click the **Manager** tab.
2. Click **New Connection**.



New Connection

*Label: GetCreditScoreContext

*API Name: GetCreditScoreContext

Description: Use external credit information to filter customer offers.

*Apex Action: GetCreditScore

Argument

Enter argument value(s) for the action you chose.

paymentSats: $Record.Payment_SAT__c

3. Enter a label to visually identify the connection (1).
4. Enter an API name. This name is used in Strategy Builder elements that require conditional statements, such as Branch Selector and Filter (2).
5. Enter a brief description for the connection (3).
6. Choose the action to use in logic elements' conditions (4).
7. Enter any parameters for the selected action (5) and click **Done**.

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:

- Modify All Data

  OR

  Manage Next Best Action Strategies

To run an action strategy:

- Run Flows

  OR

  Flow User field enabled on the user detail page

8. Click the connection label to edit its associated information.
9. Click the **>** to the right of the connection label to edit or view its details or to delete it.

Element	Description
Apex Action	Assigns the invocable action that runs when the connection is referenced in elements' conditions.
API Name	Specifies the connection name to use in logic elements' conditions. For example, `$GetCreditScoreContext.output >= 760`. Note: **API Name** is set to **Label** with underscores replacing spaces by default.
Argument	Specifies one or more parameters that the selected invocable action requires. This textbox appears only when the action has one or more arguments.
Description	Specifies the description shown in the connection details.
Label	Specifies the label displayed in the **Manager** pane for your connection.

SEE ALSO:

Connect REST API Developer Guide: Next Best Action Resources

Strategy Builder Strategies

Actions Developer Guide: Overview

Create a Custom Notification Flow for Next Best Action

Create a trigger in Process Builder to receive direct notifications about errors occurring in your strategies. Launch a flow to send error information to your desired targets.

A custom notification flow allows you to choose how you want to be informed when errors happen during Next Best Action strategy executions. It consists of two parts. First, a process created in Process Builder that subscribes to the Platform Status Alert Event, which is generated when the error occurs. Second, a notification flow that passes the information to your intended destination. Add input variables to your flow to receive the expected variables.

1. In Flow Builder, create a flow. You can direct your notifications to different places, including Chatter posts, SMS text messages, and emails. Make sure to define input variables for payload event fields that you want to use in your notifications. Input variables are flow variable resources, type text, with **Available for Inputs** checked.

   Note: A simple way to create your notification is to create a flow with the Send Email core action. From there, manually add the email address where you want the notification sent.

2. In Process Builder, create a process and for **The process starts when** select **A platform event occurs**.
3. Add a trigger. Under **Platform Event** select **Platform Status Alert Event**.
4. Select an object that allows you to define matching conditions that produce a single result.

   Note: For example, you could choose the User object and set **User ID** equal to the **Created By** ID in the event payload.

5. Add other criteria.
6. Add an immediate action and select **Flows**.
7. Name your action and select the flow you created in step one.
8. Add mappings to connect data from the payload of your event to flow inputs.
9. Save and activate your process.

SEE ALSO:

Automate Tasks with Flows

Configure the Process Trigger

Object Reference Developer Guide: PlatformStatusAlertEvent

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:
- Modify All Data

  OR

  Manage Next Best Action Strategies

To run an action strategy:
- Run Flows

  OR

  Flow User field enabled on the user detail page

To open, edit, or create a flow in Flow Builder:
- Manage Flow

To create, edit, or view processes in Process Builder:
- Manage Flow

  AND

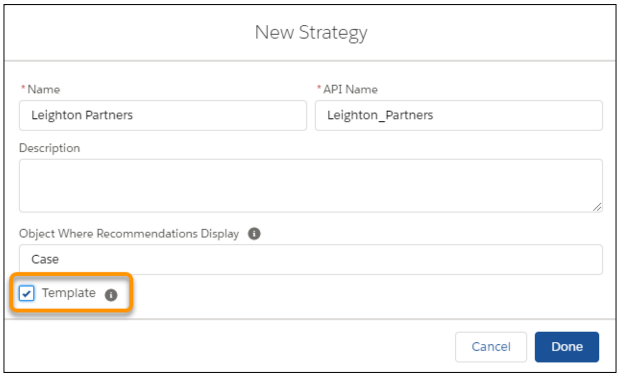
  View All Data

Create, Package, and Distribute a Strategy Builder Template

Enterprise developers can create and package strategy templates from Developer Edition orgs for use in multiple Salesforce orgs. Independent software vendors can also publish templates on AppExchange for distribution to their subscribers. Strategies not marked as templates in managed packages have intellectual property (IP) protection and can't be edited or cloned. IP protection safeguards proprietary information in your strategies.

You distribute changes to strategy templates via a managed package. Subscribers who install a strategy template can open it in Strategy Builder and clone it to customize it for their own use. When you publish updates to strategy templates via a package upgrade, template updates don't affect subscribers' copies.

1. In Strategy Builder, create the strategy that you want to make into a template.
2. Open the strategy's properties and select **Template**.



New Strategy

* Name: Leighton Partners

* API Name: Leighton_Partners

Description:

Object Where Recommendations Display: Case

☑ Template

3. If you must, create your managed package.
4. Distribute the strategy template in the managed package and let your subscribers know it's available.

Example: Suppose you build and package strategies for insurance companies. Because insurance laws and regulations can vary by location, your subscribers want the ability to modify your strategies when needed. They can do this using strategy templates you create.

SEE ALSO:

Create a Strategy with Strategy Builder

First-Generation Managed Packaging Developer Guide

Create a First-Generation Managed Package

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create a strategy:

- Modify All Data

  OR

  Manage Next Best Action Strategies

To create a managed package:

- Create AppExchange Packages

Manage Strategy Builder Action Strategies

Test, troubleshoot, and create strategies using Strategy Builder management tools.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Save Strategy Builder Action Strategies
Save your strategies or use Save As to create new a new strategy based on an existing one.

Test Strategy Builder Action Strategies
Test your strategy within Strategy Builder to see what recommendations display, given different inputs.

Troubleshoot Strategy Builder Action Strategies
Strategies can be complex, which means it's sometimes difficult to know where you went wrong when you encounter unexpected results. Use this page to determine the best tool for troubleshooting your strategy.

Inspect Strategy Builder Element Results
View the full details of each step of your strategy's execution from Strategy Builder's **Inspector** tab. Trace the path of recommendations through your strategy and identify problems in individual elements. Debug errors and see how your strategy is working behind the scenes.

Save Strategy Builder Action Strategies

Save your strategies or use Save As to create new a new strategy based on an existing one.

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:

- Modify All Data

  OR

  Manage Next Best Action Strategies

To run an action strategy:

- Run Flows

  OR

  Flow User field enabled on the user detail page

Save your new or updated action strategy by clicking **Save**. Create a strategy based on an existing one using **Save As**.

1. To save your action strategy click **Save**.
2. To create an action strategy based on an existing one, click **Save As**.
3. Replace the existing name in **Name**.
4. Replace the existing API name in **API Strategy**.

   You can have duplicate strategy names but we don't recommend it. The API name must be unique.

   To automatically generate a new API name, delete the existing API name after you rename the strategy. Click the **Name** textbox, and either tab over to or click the **API Name** textbox.
5. Optionally, replace the existing description.
6. If you want to base your strategy on a different object, click the **Object Where Recommendations Display** textbox and choose a new object.
7. Click **Done**.

SEE ALSO:

Strategy Builder Strategies

Suggest Options to Users with Recommendation Strategies

Test Strategy Builder Action Strategies

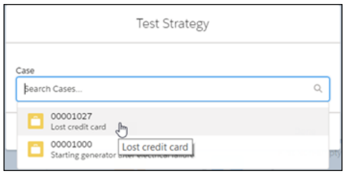
Test your strategy within Strategy Builder to see what recommendations display, given different inputs.

In Strategy Builder, you can test the strategies underlying your recommendations.

1. Create or edit a strategy.
2. To save your changes, click **Save**.

   Note: Always save before testing to test the most recent version of your strategy.

3. Click **Test**.
4. Select an object for the test.

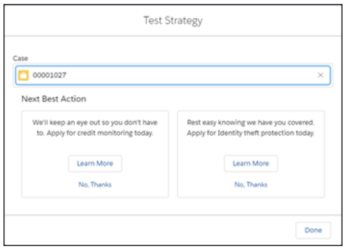
   

   00001027
   Lost credit card

   00001000
   Starting generator

   If you don't see the object that you want to test the strategy against, close the Test Strategy window. Select the properties wheel above the left pane. Change the object that the strategy is linked to by selecting an object from **Object Where Recommendations Display**. If you don't see an object listed, the strategy hasn't been linked to a specific object.

5. To test the underlying flow, choose a recommendation.

   00001027

   We'll keep an eye out so you don't have to. Apply for credit monitoring today.

   Rest easy knowing we have you covered. Apply for Identity theft protection today.

   Note: Images associated with recommendations aren't displayed when testing in Strategy Builder.

   The Test Strategy window doesn't show all possible error messages. Strategies are executed from right to left, starting at Output. If a particular Branch Selector expression results in a closed branch, the child elements of that branch (the elements to the left) are not executed. This process makes strategy evaluation faster, but it also means that any branches with false expressions could have errors that aren't exposed. The Test button shows what the user sees. To get a complete view of any errors occurring at run-time, use the Inspector tab in the Toolbox. Inspector highlights errors from all elements.

SEE ALSO:

[Inspect Strategy Builder Element Results]

[Strategy Builder Strategies]

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:

- Modify All Data

  OR

  Manage Next Best Action Strategies

To run an action strategy:

- Run Flows

  OR

  Flow User field enabled on the user detail page

Troubleshoot Strategy Builder Action Strategies

Strategies can be complex, which means it's sometimes difficult to know where you went wrong when you encounter unexpected results. Use this page to determine the best tool for troubleshooting your strategy.

If something goes wrong with your strategy, you have several troubleshooting options.

- Start by using the basic test function in Strategy Builder. After you create and save a strategy, click **Test** in the menu bar.
- For a more detailed view of your strategy execution, see the **Inspector** tab in the Strategy Builder Toolbox. The Inspector tab lists specific errors and gives you a detailed view of how your strategy executes.
- If you can't find the problem in the **Inspector** tab, or you want to troubleshoot for a specific user, try using the Apex debug log. Next Best Action has a specific debug log category.
- To receive full error reports sent directly to your email, a Chatter post, a text message, or other outlet, try creating a custom notification flow and a Process Builder trigger. Using a Platform Status Alert Event, you can subscribe to Next Best Action events and respond when errors occur.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Inspect Strategy Builder Element Results

Inspect Strategy Builder Element Results

View the full details of each step of your strategy's execution from Strategy Builder's **Inspector** tab. Trace the path of recommendations through your strategy and identify problems in individual elements. Debug errors and see how your strategy is working behind the scenes.

When testing your strategy doesn't return the recommendations you expect, investigate the execution details of the strategy or a selected element using the **Inspector** tab.

1. Click the **Inspector** tab.
2. Click **Test** and select an object.

   Note: Provide a sample `recordId` to test your strategy in the inspector. You can do so in either of the following ways:

   - While Inspector is open, click **Test** and select a record. The `recordId` of the selected record is pasted into the **Record ID** field of the inspector. Close the Test window and click **Run** in the inspector.
   - Copy a record ID from the URL of a record page and paste it into the RecordId field manually.

3. Click **Run**.

EDITIONS

Available in: Salesforce Classic

Essentials, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or manage action strategies:

- Modify All Data

  OR

  Manage Next Best Action Strategies

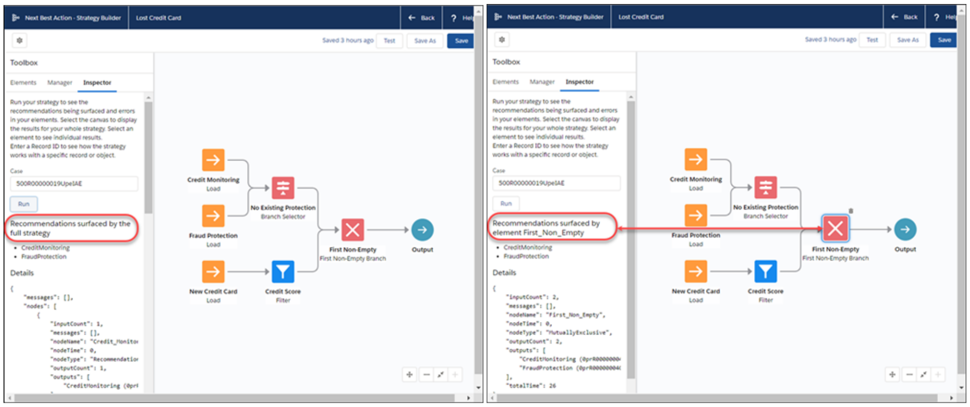
To run an action strategy:

- Run Flows

  OR

  Flow User field enabled on the user detail page

Note: Inspector can show a single element’s results, or the results for all elements in the strategy. If you select an element, you see recommendations surfaced by that element. If you have no elements selected, you see recommendations surfaced by the strategy.



Run your strategy to see the recommendations being surfaced and errors in your elements. Select the canvas to display the results for your whole strategy. Select an element to see individual results.
Enter a Record ID to see how the strategy works with a specific record or object.

Case: 500R00000019UpelAE

Recommendations surfaced by the full strategy

- CreditMonitoring
- FraudProtection

```
{
    "messages": [],
    "nodes": [
        {
            "inputCount": 1,
            "messages": [],
            "nodeName": "Credit_Monito
            "nodeTime": 0,
            "nodeType": "Recommendatio
            "outputCount": 1,
            "outputs": [
                "CreditMonitoring (0prl
```

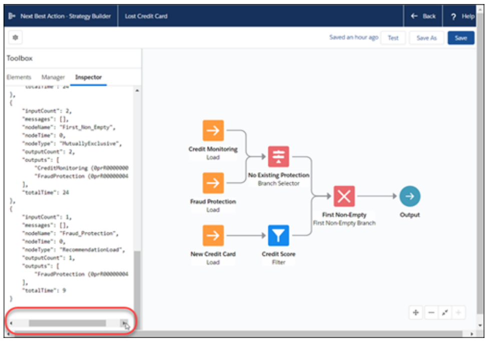
Recommendations surfaced by element First_Non_Empty

- CreditMonitoring
- FraudProtection

```
{
    "inputCount": 2,
    "messages": [],
    "nodeName": "First_Non_Empty",
    "nodeTime": 0,
    "nodeType": "MutuallyExclusive",
    "outputCount": 2,
    "outputs": [
        "CreditMonitoring (0prR0000000
        "FraudProtection (0prR00000004
    ],
    "totalTime": 26
```

Note: To see accurate results, you have to save your strategy before testing it. If you change the strategy or an element, **Run** becomes **Save and Run**.

4. To scroll right, use the horizontal scroll bar at the bottom of the Inspector pane.



```
},
{
    "inputCount": 2,
    "messages": [],
    "nodeName": "First_Non_Empty",
    "nodeTime": 0,
    "nodeType": "MutuallyExclusive",
    "outputCount": 2,
    "outputs": [
        "CreditMonitoring (0prR0000000
        "FraudProtection (0prR00000004
    ],
    "totalTime": 24
},
{
    "inputCount": 1,
    "messages": [],
    "nodeName": "Fraud_Protection",
    "nodeTime": 0,
    "nodeType": "RecommendationLoad",
    "outputCount": 1,
    "outputs": [
        "FraudProtection (0prR00000004
    ],
    "totalTime": 9
}
```

5. If you want to view recommendations for a different object, click **Test**, clear your current selection, and choose a new object from the dropdown. To update the recommendations in the inspector, click **Run**.

SEE ALSO:

Test Strategy Builder Action Strategies

Strategy Builder Strategies

Strategy Builder Elements

Use this page to quickly access a list of Strategy Builder elements and learn how they work together to create unique strategies.

Use elements to create your strategies by opening Strategy Builder and selecting Elements in the Toolbox. Drag elements onto the canvas to get started.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Strategy Builder Enhance Element

Get AI-driven predictions from services such as Einstein Discovery and Einstein Prediction Builder to enhance Next Best Action recommendations with additional information, such as propensity scores. The Enhance element allows you to modify a set of recommendations on the fly, every time a strategy is executed. These recommendations can be static and live as records in Salesforce, or dynamic and sourced from external data sources or other Salesforce objects.

Strategy Builder Generate Element

With the Generate element, you can dynamically generate personalized recommendations where a large number of possibilities makes it inconvenient to create recommendations manually. The Generate element allows you to create in-memory, on-the-fly recommendations, either from an external data source or from other Salesforce objects.

Strategy Builder Load Element

Load is the first element in a strategy branch. Load and filter the records of a Recommendation object. Or load and filter the records of any object, and convert them into recommendations at the end of the strategy using the Map element. Your load elements determine which of your recommendations are evaluated when your strategy executes.

Strategy Builder Filter Element

Create an expression that allows you to block or filter out undesirable recommendations, depending on the context. The expression is evaluated for every recommendation that passes through the branch.

Strategy Builder Limit Reoffers Element

Determine how often a user sees the same recommendation. You can decide how many times the user must react to a recommendation and how many days to wait before displaying the recommendation again.

Strategy Builder Map Element

The Map element lets you use formulas to create Recommendation fields and modify existing fields without Apex code. Instead, it relies on expressions and formulas. Use the Map element to pass data from a Recommendation field with one name to a Flow input with a different name. Or use it to modify current values for Description, Name, and other fields and personalize them with context-specific data.

Strategy Builder Sort Element

Choose how recommendations are ordered within a branch and reorder them using Recommendation fields.

Strategy Builder Branch Merge Element

Combine recommendations from multiple branches into a single branch.

Strategy Builder Branch Selector Element

Filter multiple branches through a branch selector and create unique expressions for each branch. If the expression is true, recommendations in the branch are allowed through and combined into a single branch.

Strategy Builder First Non-Empty Branch Element

The first non-empty branch element allows you to filter branches in the order they appear on the canvas. The first branch that contains recommendations is allowed through, all other branches are blocked.

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

Strategy Builder Enhance Element

Get AI-driven predictions from services such as Einstein Discovery and Einstein Prediction Builder to enhance Next Best Action recommendations with additional information, such as propensity scores. The Enhance element allows you to modify a set of recommendations on the fly, every time a strategy is executed. These recommendations can be static and live as records in Salesforce, or dynamic and sourced from external data sources or other Salesforce objects.

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.
`Apex Action`	Search or select an Apex action, which calls an Apex class. An Apex class must have a method marked as an invocable method to appear as an Apex action in declarative tools like Strategy Builder.
`Argument`	Specify one or more parameters for the selected Apex action.

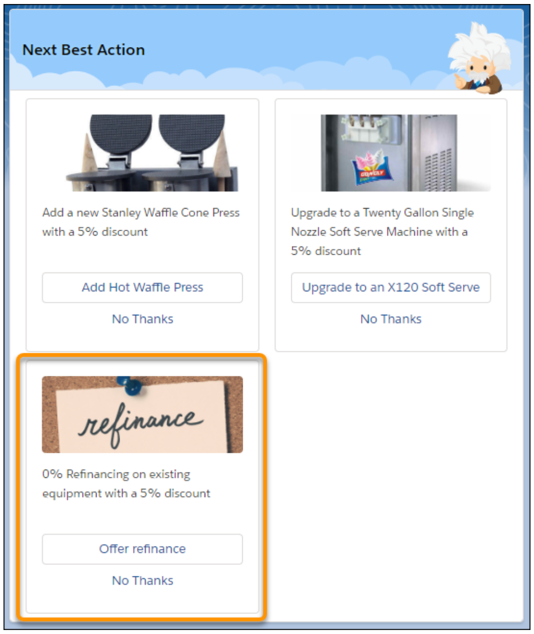
EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example: Assume that your company integrates separate data sources from the manufacturers of products your business sells. Those data sources include information about the current availability of each item (in stock, back ordered, or unavailable). You can connect an Enhance element to your strategy’s Load or Generate element to provide that information to users in the recommendation.

Example: You can use the Enhance element to calculate a discount percentage for your customers based on how long your company has managed their account. Or you can use it to A/B test two branches of recommendations.


Next Best Action

Add a new Stanley Waffle Cone Press with a 5% discount

Add Hot Waffle Press

No Thanks

Upgrade to a Twenty Gallon Single Nozzle Soft Serve Machine with a 5% discount

Upgrade to an X120 Soft Serve

No Thanks

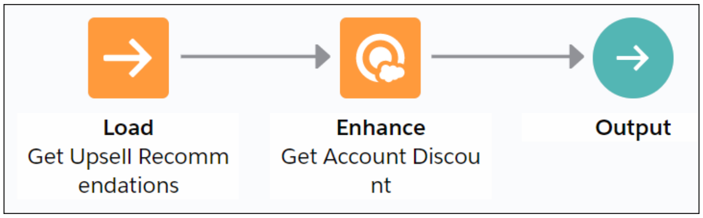
0% Refinancing on existing equipment with a 5% discount

Offer refinance

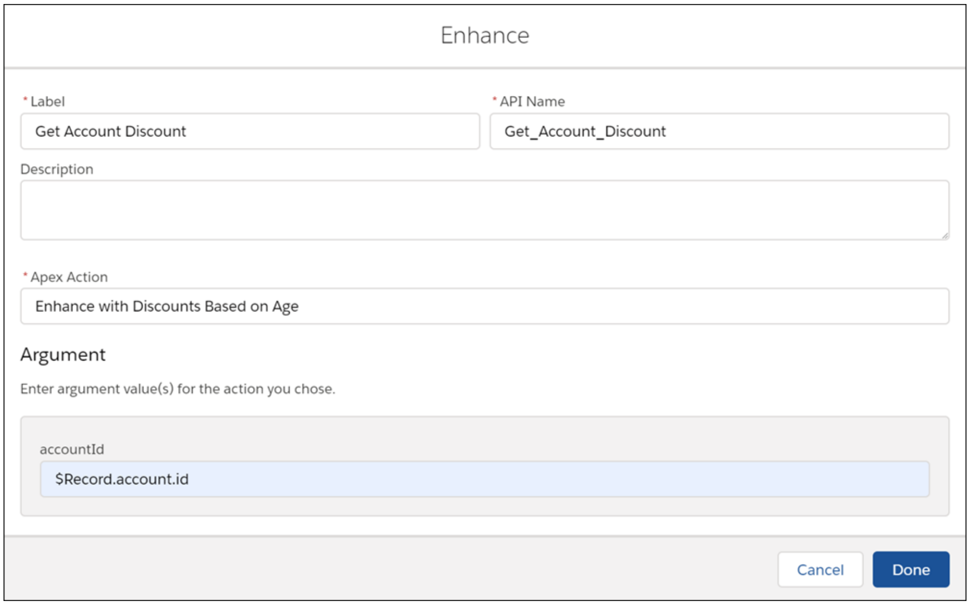
No Thanks

Example: Suppose you use Next Best Action to provide upsell recommendations. You want to add a 5% discount to your product recommendations for those customers who have been with your company for more than one year. Customers of more than two years get a 10% discount, customers of more than five years get a 20% discount, and so on. Use the Enhance element to call an Apex action that performs a SOQL query. The query retrieves the Account age and appends it to the description of all incoming recommendations.

The strategy used with an Enhance element can be as simple as Load -> Enhance -> Output. All recommendations the Load element retrieves or loads are passed as a list of recommendations to the underlying invocable method.



When configuring the Enhance element, select **Enhance with Discounts Based on Age** as the Apex action and specify **$Record.id** as the input parameter.


Enhance

*Label

Get Account Discount

*API Name

Get_Account_Discount

Description

*Apex Action

Enhance with Discounts Based on Age

Argument

Enter argument value(s) for the action you chose.

accountId

$Record.account.id

The Enhance element in turn calls the `getDiscounts` invocable method in the `Enhance_GetAccountDiscount` class. Notice how the description of each recommendation has a discount value appended to it `(r.Description + ‘ with a 5% discount’).`

```
global class Enhance_GetAccountDiscount {
    @InvocableMethod(label='Enhance with Discounts Based on Age' description='Returns
 an enhanced set of recommendations with appropriate discounts')
   global static List<List<Recommendation>> getDiscounts(List<DataContainer> inputData){


        List<Recommendation> recommendations = inputData[0].recommendations;
        List<List<Recommendation>> outputs = new List<List<Recommendation>>();

        Account[] accounts = [SELECT Name, Description,CreatedDate, id FROM Account
WHERE id = :inputData[0].accountId];
           Double ageAccountMonths =
accounts[0].CreatedDate.date().monthsBetween(date.today());
        Double ageAccount = ageAccountMonths/12;
        List<Recommendation> returnedRecommendations = new List<Recommendation>();
        for (Recommendation r:recommendations){
            if(ageAccount > 1){
                r.Description = r.Description + ' with a 5% discount';
            }
            else if (ageAccount > 2){
                r.Description = r.Description + ' with a 10% discount';
            }
            else if (ageAccount > 5){
                r.Description = r.Description + ' with a 20% discount';
            }
            returnedRecommendations.add(r);
        }
        outputs.add(returnedRecommendations);
        return outputs;
    }

}
```

Usage

The Enhance element requires an Apex action marked as an invocable method.

```
@InvocableMethod(
label='Enhance with Discounts Based on Age'
description='Returns an enhanced set of recommendations with appropriate discounts')
```

Use the Enhance element in combination with the Strategy Builder Load or Generate element.

The Enhance element can pass any number of inputs to the Apex action. The input parameter must be a list or a list of lists of a user-defined Apex object (for example, a custom class called `DataContainer`). The user-defined Apex object must include a `List<Recommendation>` variable. The `List<Recommendation>` variable is automatically defined with the recommendations that pass into the Enhance element.

```
global class DataContainer {
    @InvocableVariable
    public string accountId;

    @InvocableVariable
    public List<Recommendation> recommendations;
}

________
global static List<List<Recommendation>> invocableMethod(List<DataContainer> inputData)
```

The Enhance element returns a list of recommendations, `List<List<Recommendation>>`. These recommendation enhancements exist only in memory and don't persist after the strategy is executed.

```
global static List<List<Recommendation>> invocableMethod(List<DataContainer> inputData)
```

SEE ALSO:

- Strategy Builder Generate Element
- Strategy Builder Load Element
- Flow Element: Apex Action

Strategy Builder Generate Element

With the Generate element, you can dynamically generate personalized recommendations where a large number of possibilities makes it inconvenient to create recommendations manually. The Generate element allows you to create in-memory, on-the-fly recommendations, either from an external data source or from other Salesforce objects.

EDITIONS

Available in: Salesforce Classic

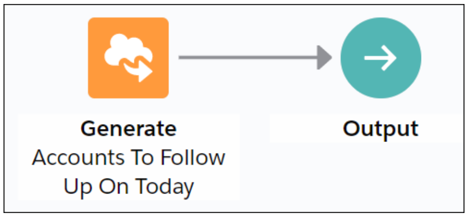
Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.

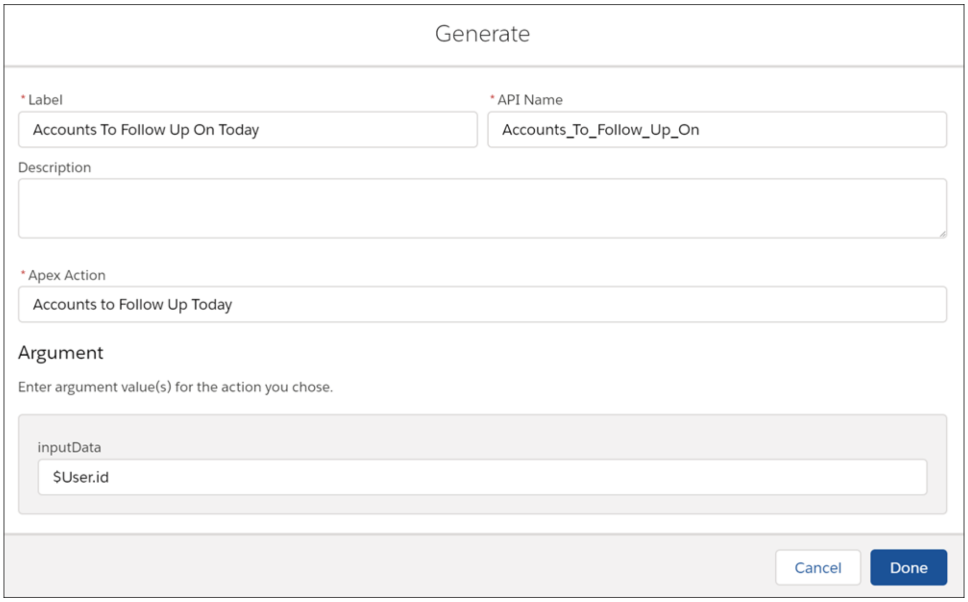
Field	Description
`Description`	Optional description of the element and how it works within the strategy.
`Apex Action`	Search or select an Apex action, which calls an Apex class. An Apex class must have a method marked as an invocable method in order to appear as an Apex action in declarative tools like Strategy Builder.
`Argument`	Specify one or more parameters for the selected Apex action.

Example: Assume that your company has a large catalog of products and you use a screen flow to recommend accessories to your customers based on their past product purchases. Instead of creating a single, static recommendation for each individual accessory, you can maintain that information in the Account or Product object in Salesforce. Or you can store information in external data sources like Commerce Cloud or a SQL database. Use a Generate element with an Apex invocable action to call the Apex class and generate accessory recommendations dynamically for your strategy.

Example: Suppose you want to show a service agent a list of key accounts to follow up with after a set number of days has passed since the previous contact. With the Generate element, you can call an Apex action that makes a SOQL query for Account where the Owner is the logged-in user (the agent). This query identifies the accounts who were last contacted more than, say, 90 days ago. Next Best Action returns the relevant accounts in the form of recommendations. The strategy can be as simple as the Generate element with an Output element.



When you configure the Generate element, select **Accounts to Follow Up Today** as the Apex action and specify **$User.id** as an input parameter.


Generate

* Label: Accounts To Follow Up On Today

* API Name: Accounts_To_Follow_Up_On

Description:

* Apex Action: Accounts to Follow Up Today

Argument

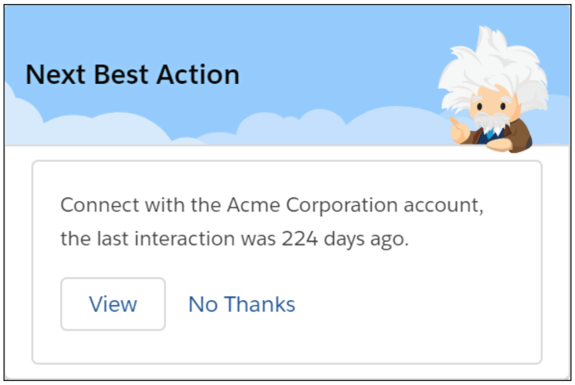
Enter argument value(s) for the action you chose.

inputData: $User.id

The Generate element calls the `getAccounts` invocable method in the `Generate_GetAccountsToFollowUp` Apex class. This method retrieves the relevant accounts and creates a list of recommendations. The recommendation description includes the name of the account (`account.Name`) and the number of days since the last contact (`daysSinceLastContact`).

```
global class Generate_GetAccountsToFollowUp {
    @InvocableMethod(label='Accounts to Follow Up Today'
                     description='Recommend accounts the current user should follow
up on today')
    global static List<List<Recommendation>> getAccounts(List<String> inputData){
        List<List<Recommendation>> outputs = new List<List<Recommendation>>();
        Integer daysSinceLastContact;
        Account[] accounts = [SELECT Name, Description, LastContactDate__c, OwnerId
FROM Account WHERE OwnerId = :inputData[0]];
        List<Recommendation> recs = new List<Recommendation>();
        for (Account account:accounts) {
            if (account.LastContactDate__c != null){
                daysSinceLastContact =
account.LastContactDate__c.daysBetween(date.today());
                if (daysSinceLastContact > 90){
                    Recommendation rec = new Recommendation(
                        Name = account.Name,
                        Description = 'Connect with the ' + account.Name + ' account,
 the last interaction was '+ daysSinceLastContact + ' days ago.',
                        //Pre-req: Create a screen flow with the name simpleFlow

                        ActionReference = 'simpleFlow',
                        AcceptanceLabel = 'View'
                    );
                    recs.add(rec);
                }
            }
        }
        outputs.add(recs);
        return outputs;
    }
}
```

When you execute the strategy, the resulting recommendation includes the name of the account and the number of days since the last contact with them.



Connect with the Acme Corporation account, the last interaction was 224 days ago.

Usage

The Generate element requires an Apex action marked as an invocable method.

```
@InvocableMethod(
label='Related Wikipedia Pages'
description='Recommend wikipages that are related to the named input wikipage')
```

The Generate element can pass any number of inputs to the Apex action, either as lists or a list of lists of primitives, sObjects, and user-defined Apex objects. To provide more than one input, the input parameter must be a list or a list of lists of a user-defined Apex object (for example, a custom class called `DataContainer`).

```
List<String> relatedTo
```

OR

```
global class DataContainer {
@InvocableVariable
public string accountId;
}
____
global static List<List<Recommendation>> invocableMethod(List<DataContainer> inputData)
```

The Generate element returns a list of recommendations. Invocable methods support returning either a list of an sObject type or a list of lists of an sObject type. Since the Enhance element operates not on a single recommendation but on a list of recommendations, the method must return a `List<List<Recommendation>>`.

```
global static List<List<Recommendation>> invocableMethod(List<DataContainer> inputData)
```

SEE ALSO:

Strategy Builder Enhance Element

Flow Element: Apex Action

Strategy Builder Load Element

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Load is the first element in a strategy branch. Load and filter the records of a Recommendation object. Or load and filter the records of any object, and convert them into recommendations at the end of the strategy using the Map element. Your load elements determine which of your recommendations are evaluated when your strategy executes.

Load recommendations from the records of any standard or custom object. You can use objects such as Recommendation, Account, Product, and Opportunity when you build a strategy. Choose criteria for when to load a recommendation. Filter out certain records from a strategy. Sort your records by selecting an object value.

A strategy treats another object the same as it does a Recommendation object until the end, when it converts it into a recommendation. If you choose an object other than Recommendation, add a Map element after the Load element. Use the Map element to map fields from the object's records to required fields on the Recommendation object.

In Strategy Builder, you can load up to 1,000 records in a strategy. A Strategy Builder strategy has a limit of 100 Load elements on the canvas. To load more than 1,000 records, create your recommendation strategy in Flow Builder, which has a load limit of 50,000 records.

Field	Description
Label	The name of the element as it appears on the canvas.
API Name	The API name of the element. The API name must be unique.
Description	Optional description of the element and how it works within the strategy.
Object	The object whose records are loaded, filtered, and converted into recommendations.
Condition Requirements	Determines the logic to evaluate conditions. To load the recommendation if it meets all the specified criteria, select **All Conditions are Met**. To load the recommendation if it meets any listed criteria, select **Any Condition is Met**.
Field	Choose a field from the Recommendation object to evaluate whether the recommendation is loaded into the strategy.
Operator	Choose an operator.
Value	Enter a value for your chosen field. Values can be simple numbers, string phrases, or formulas that use Salesforce formula support. Don't enclose string or number values with quotes. Picklists aren't supported.
Add Condition	Creates an extra set of conditions.

SEE ALSO:

- Suggest Options to Users with Recommendation Strategies
- Create Recommendations
- Display Recommendations
- Strategy Builder Enhance Element

Strategy Builder Filter Element

Create an expression that allows you to block or filter out undesirable recommendations, depending on the context. The expression is evaluated for every recommendation that passes through the branch.

Field	Description
Label	The name of the element as it appears on the canvas.
API Name	The API name of the element. The API name must be unique.
Description	Optional description of the element and how it works within the strategy.
Filter Expression	Create an expression that is evaluated for each recommendation that you load into your strategy. If the expression is true, the recommendation is allowed through. If false, the recommendation doesn't progress further through the strategy. Filter Expression accepts Standard Salesforce formulas. For more information, see Formula Operators and Functions by Context.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: Use `$Record` to reference fields from the context object. The context object is the object where you intend to surface your recommendations and can be changed by editing your strategy and choosing an object under **Object Where Recommendations Display**. Use plain text field labels to reference Recommendation object fields. Examples: `$Record.status != 'New'`, `RejectionLabel == 'No, thanks.'` For more information, see Write a Strategy Builder Expression.

Example: Suppose that you want to surface recommendations on the Case object so your service agents can suggest offers to your customers. If you want to suggest only credit card offers, create a `Category` field for the Recommendation object. Add a Credit Card Offer category to your field. Add a filter element and use the formula `Category_c = 'CreditCardOffer'` in **Filter Expression**.

Usage

Filter is the best way to remove certain recommendations from a strategy branch. Add the element to a branch and create an expression to evaluate every recommendation that passes through the branch.

SEE ALSO:

- Suggest Options to Users with Recommendation Strategies
- Create Recommendations
- Display Recommendations
- Formula Operators and Functions by Context

Strategy Builder Limit Reoffers Element

Determine how often a user sees the same recommendation. You can decide how many times the user must react to a recommendation and how many days to wait before displaying the recommendation again.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view a recommendation strategy with a Limit Reoffers element on a Lightning record page:

- Run Flows

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.
`User Reaction`	Choose a user reaction to base your limits on. For example, if you select **User Rejects the Recommendation**, your element only limits repeat offers after the recommendation is rejected.
`Number of Reactions`	Choose how many times you want the user to react before the recommendation is limited.
`Time Period in Days`	Choose how many days the system waits after the user has reacted the specified number of times before a repeat offer is shown to the same user. Time Period in Days is based on days, not hours. If the time period is set to 1, and the user accepts the recommendation at any time on Monday, the offer doesn’t display again until the start of Wednesday. So a one-day time period could be as few as 25 hours in duration or as many as 48 hours.

Example: Let's say you have a renewal offer that you want to surface at most one time per year. If a user has already accepted the offer and filled out the renewal, you don't want to show the same offer again. For this example, for User Reaction, select **User Accepts the Recommendation**. For Number of Reactions, select **1**, and set the time period for 365 days for an annual renewal.

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

Create Recommendations

Display Recommendations

Strategy Builder Map Element

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

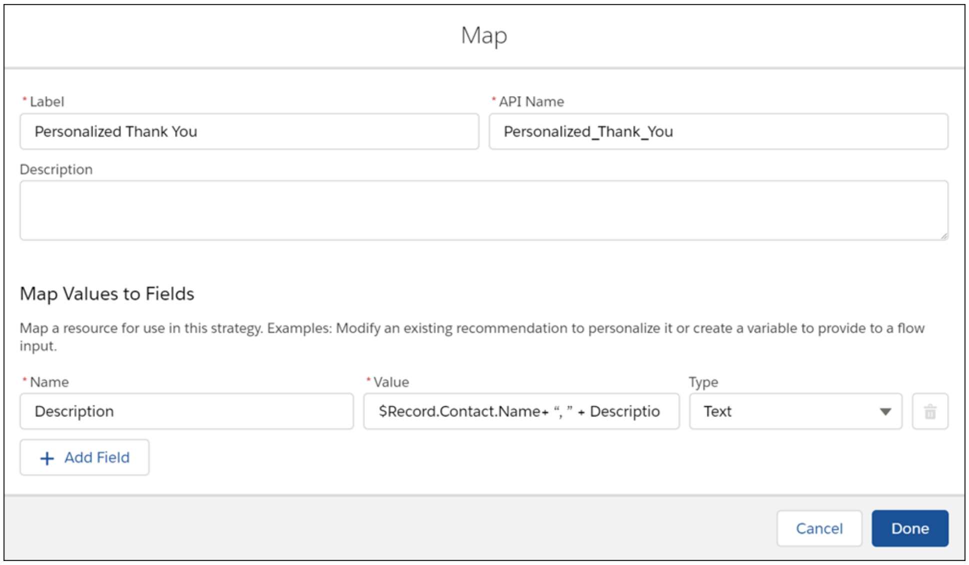
The Map element lets you use formulas to create Recommendation fields and modify existing fields without Apex code. Instead, it relies on expressions and formulas. Use the Map element to pass data from a Recommendation field with one name to a Flow input with a different name. Or use it to modify current values for Description, Name, and other fields and personalize them with context-specific data.

If you load an object other than Recommendation, add a Map element after the Load element and before the Output element in your strategy. Use the Map element to map fields from the records to required fields on the Recommendation object. For example, map the product Title field to the recommendation Name field. Mapping fields converts the filtered records into recommendations that are surfaced via the Next Best Action component and your own apps.

Example: You can include the name of a contact in a recommendation, and further personalize the recommendation with text. Suppose that you have a recommendation with the description, "Thank you for being a loyal customer. We truly appreciate your business!" Using the Map element, you can personalize the description. Add the name of the contact to the description, for example, "Lauren Boyle, Thank you for being a loyal customer. We truly appreciate your business!"

- Use a Load element to load all the recommendations you want to change. Or you can add a Generate element and pass in dynamically generated recommendations.
- Add a Map element. In the Name field, select **Description** and in the Value field, enter this expression:

  `$Record.Contact.Name+ “, ” + Description`. Leave the Type field as `Text`.



Map

* Label: Personalized Thank You

* API Name: Personalized_Thank_You

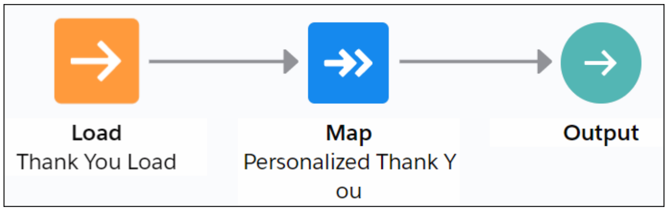
Description:

Map Values to Fields

Map a resource for use in this strategy. Examples: Modify an existing recommendation to personalize it or create a variable to provide to a flow input.

* Name	* Value	Type
Description	$Record.Contact.Name+ “, ” + Descriptio	Text

- Place the “Personalized Thank You” Map element after the Load element. It modifies the descriptions of all recommendations that pass through it.



- When you execute the strategy, your recommendations include the contact name for the current case.

Strategy Builder Sort Element

Choose how recommendations are ordered within a branch and reorder them using Recommendation fields.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.
`Recommendation Field`	Choose a field from the Recommendation object to sort on.
`Sort Direction`	Choose whether you want to sort your recommendations in an ascending or descending order.
`Sort Empty Values to Top`	Recommendations that don’t contain information in the field you chose in Recommendation Field are sorted to the top when selected.
`Maximum Recommendations`	Limits the number of recommendations allowed to pass through the element.

Usage

Sort the order of your recommendations in a branch by choosing a value from the Recommendation object to sort on. Choose whether you want to sort in an ascending or descending order, and decide how many recommendations to allow through.

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

Create Recommendations

Display Recommendations

Strategy Builder Branch Merge Element

Combine recommendations from multiple branches into a single branch.

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.
`Maximum Recommendations`	Determines the maximum number of recommendations allowed through the branch where the sort element is placed.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Usage

Merge multiple branches into a single branch and limit the number of recommendations allowed through the branch with the branch merge element.

SEE ALSO:

Suggest Options to Users with Recommendation Strategies

Create Recommendations

Display Recommendations

Strategy Builder Branch Selector Element

Filter multiple branches through a branch selector and create unique expressions for each branch. If the expression is true, recommendations in the branch are allowed through and combined into a single branch.

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.
`Condition`	Create an expression for each branch that flows through the element. If the expression is true, the recommendations in the branch are allowed through. If false, the recommendations in the branch don't progress any further through the strategy. Condition accepts standard Salesforce formula functions. For more information, see Formula Operators and Functions by Context.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: Use *`$Record`* to reference fields from the context object. The context object is the object where you intend to surface your recommendations and can be changed by editing your strategy and choosing an object under **Object Where Recommendations Display**. Example: *`ISPICKVAL($Record.status, 'New')`*. For more information, see Write a Strategy Builder Expression.

Example: Suppose that you want to surface recommendations on the Case object so your service agents can suggest offers to your customers. If a case has been escalated, you want to offer a special discount. To do so you, create a load element that loads the recommendations associated with your offer. Create a branch selector that only allows recommendations from the branch if the case has an escalated status. Make your offer load element a child of the branch selector element. In **Condition** on the branch selector element, use the following formula: *`ISPICKVAL($Record.status, 'Escalated')`*.

Usage

Branch selector is an important element when you want to weed out entire branches at once. Unlike a filter element, it can't filter based on individual recommendations.

SEE ALSO:

- Suggest Options to Users with Recommendation Strategies
- Create Recommendations
- Display Recommendations
- Formula Operators and Functions by Context

Strategy Builder First Non-Empty Branch Element

The first non-empty branch element allows you to filter branches in the order they appear on the canvas. The first branch that contains recommendations is allowed through, all other branches are blocked.

EDITIONS

Available in: Salesforce Classic

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Label`	The name of the element as it appears on the canvas.
`API Name`	The API name of the element. The API name must be unique.
`Description`	Optional description of the element and how it works within the strategy.

Example: Let's say you have five different types of credit card offers that could be surfaced to a single user. Although each offer type is unique and must have its own branch, you only want to surface one type. To do so, filter all of your branches that contain credit card offers through a first non-empty branch, in priority order from top to bottom. Your element only allows the first branch that contains recommendations.

Usage

Branches are filtered through the first non-empty branch element in the order that they appear on the canvas, moving from top to bottom. The element evaluates each branch until it finds one that contains recommendations. When the element recognizes that a branch contains recommendations, it allows those recommendations through and blocks recommendations from all other branches.

SEE ALSO:

- Suggest Options to Users with Recommendation Strategies
- Create Recommendations
- Display Recommendations

Display Recommendations

After creating a strategy, choose a page to run your strategy and display your recommendations. You can use a Lightning record page, an app's home page, an Experience Cloud site page, a Visualforce page, or an external site, depending on where you want recommendations to appear.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

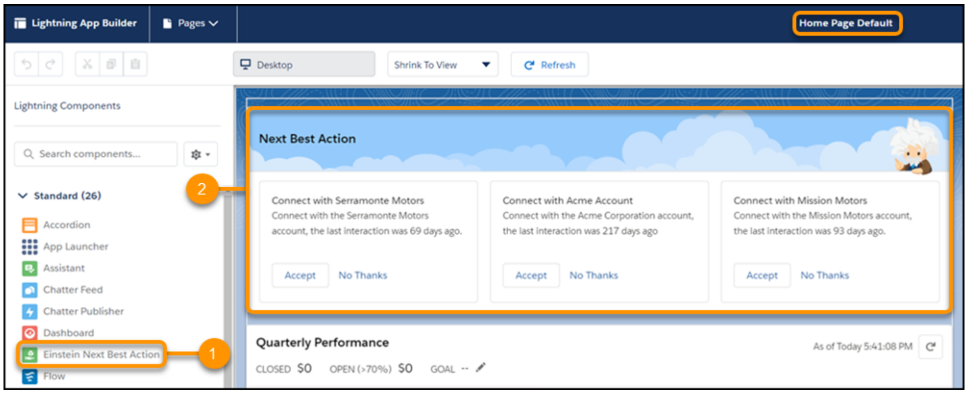
Lightning Page (Lightning App Builder)

- On a Lightning page in Lightning App Builder, create, edit, or clone a record page.
- Drag Einstein Next Best Action from the component list to the location on the page where you want to display it.
- Choose an action strategy and the number of recommendations that you want the component to display.

If you want to show users flows and quick actions in addition to recommendations, use the Actions & Recommendations component on your Lightning record page. You can create an Actions & Recommendations deployment that specifies action strategies and how you want your recommendations to appear.

App Home Page

- Create a strategy for the Next Best Action component. Use global variables such as $User.Id when you create the strategy. Use global variables because the home page isn't a record page and isn't associated with objects, like Case, Account, or Product.
- Navigate to your org's Home page.
- Click ⚙, and select Edit Page.
- From the list of Lightning components on the left (1), drag the Einstein Next Best Action component to the home page (2).



Connect with Serramonte Motors
Connect with the Serramonte Motors account, the last interaction was 69 days ago.

Connect with Acme Account
Connect with the Acme Corporation account, the last interaction was 217 days ago

Connect with Mission Motors
Connect with the Mission Motors account, the last interaction was 93 days ago.

Experience Builder Site Page (Experience Builder)

- In Experience Builder, create or edit a site page.
- Drag Suggested Actions from the component list to the location on the page where you want to display it.

Visualforce Page: Use Lightning Out to add the lightning:nextBestAction component.

Custom Apps: Add Einstein Next Best Action functionality into your app with the global lightning:nextBestAction component.

Einstein Next Best Action Component

Einstein Next Best Action uses strategies that apply your org's business rules to display context-sensitive suggested offers and actions on your Lightning record pages.

SEE ALSO:

Suggested Actions

Flow Builder for Service and the Actions & Recommendations Component

Lightning Aura Components Developer Guide: Add Aura Components to Any App with Lightning Out (Beta)

Einstein Next Best Actions Considerations

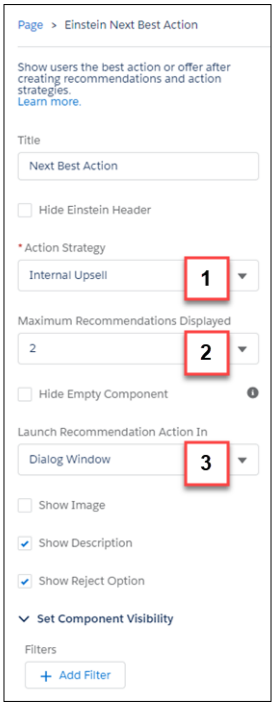
Einstein Next Best Action Component

Einstein Next Best Action uses strategies that apply your org's business rules to display context-sensitive suggested offers and actions on your Lightning record pages.

1. Create a recommendation strategy in Strategy Builder.
2. Drag the Einstein Next Best Action component onto your record page.

   Note: In Experience Builder, the component is called Suggested Actions.

3. In the property editor, select the strategy you want to display (1). Enter the maximum number of recommendations to display (2) and choose where recommendations open when accepted (3).



Page > Einstein Next Best Action

Show users the best action or offer after creating recommendations and action strategies.
Learn more.

Title
Next Best Action

□ Hide Einstein Header

* Action Strategy
Internal Upsell

Maximum Recommendations Displayed
2

□ Hide Empty Component

Launch Recommendation Action In
Dialog Window

□ Show Image

☑ Show Description

☑ Show Reject Option

Set Component Visibility

Filters

+ Add Filter

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create and save Lightning pages in the Lightning App Builder:

- Customize Application

To run a recommendation strategy on a Lightning record page:

- Run Flows

  OR

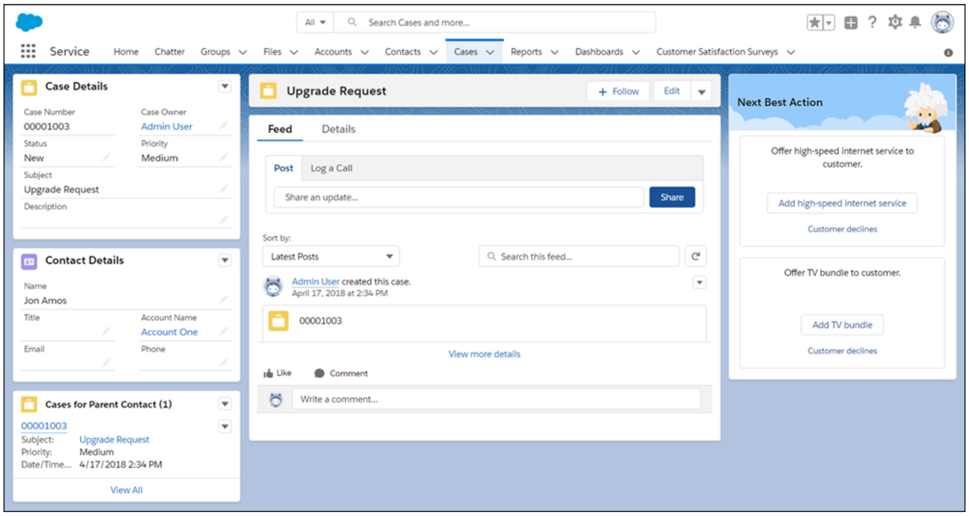
  Flow User field enabled on the user detail page

  OR

  For Experience Cloud, the FlowSites perm provides org-wide access. To restrict access to users by profile or permission set, add a component visibility filter to the Suggested Actions component.

Component	Description
Title	Displays the title for the component on the Record page.
Hide Einstein Header	Hides the Einstein Recommendations graphical header.
Strategy	Displays all available strategies created in Strategy Builder.
Maximum Recommendations Displayed	Displays up to four recommendations.
Hide Empty Component	Displays the component only when there are recommendations available initially.
Launch Recommended Action In	Specifies whether recommendations open in a display window or a new browser window.
Show Image	Shows images associated with each displayed recommendation. If there isn't an image, a placeholder displays.
Show Description	Displays the recommendation descriptions.
Show Reject Option	Displays the reject option.
Set Component Visibility	Allows Dynamic Lightning Pages by adding filter conditions and logic to the component properties in the Lightning App Builder.

Here's how a strategy looks with the Einstein header and no images in the Service console:



Case Details

Case Number: 00001003
Case Owner: Admin User
Status: New
Priority: Medium
Subject: Upgrade Request
Description:

Contact Details

Name: Jon Amos
Title:
Account Name: Account One
Email:
Phone:

Cases for Parent Contact (1)

00001003
Subject: Upgrade Request
Priority: Medium
Date/Time... 4/17/2018 2:34 PM

Upgrade Request

Admin User created this case.
April 17, 2018 at 2:34 PM
00001003

Next Best Action

Offer high-speed internet service to customer.

Offer TV bundle to customer.

Report On and Track a Recommendation

Create a custom report type to report on and track recommendation data and strategy metrics. You can see the monthly total recommendations that a Salesforce org's strategies served. And you can analyze which recommendations are accepted and rejected, who responds to them, and more.

Salesforce updates recommendation strategy metrics each time a strategy is executed or a recommendation is accepted or rejected. Analyze usage metrics to better understand how your strategies are performing. Use this knowledge to improve your strategies' logic and increase their effectiveness.

For example, run A/B tests on two different strategies and compare their relative performance. If your service agents accept more recommendations served from Strategy B, use metrics to discover why.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

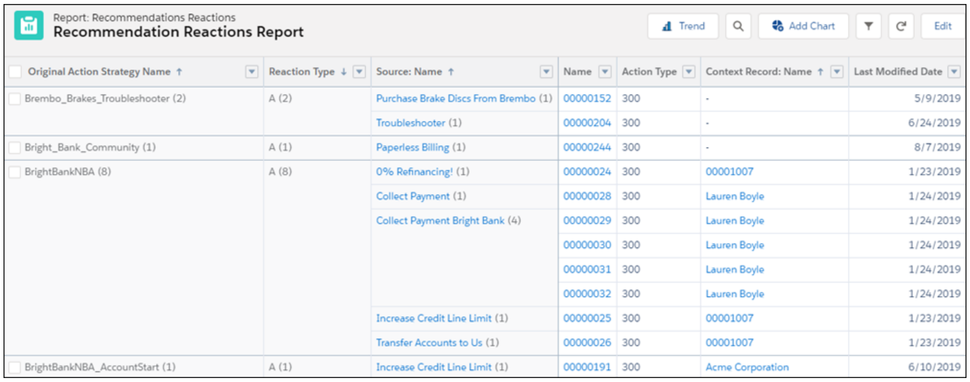
To create, edit and delete custom report types:

- Manage Custom Report Types

To view recommendation metrics data:

- Modify All Data or Manage Next Best Action Strategies

1. For complete instructions on creating custom report types, search for Create a Custom Report Type in Salesforce Help.
2. For strategy-level data that's aggregated for each calendar month, use the Recommendation Strategy Metrics primary object. For recommendation-level details, use the Recommendation Reactions primary object instead.
3. Using the Recommendation Strategy Metrics primary object, combine fields from it (like Recommendation Source ID) and the related strategy (like Context Record Type). Using the Recommendation Reactions primary object, include fields to report on, such as Context Record ID, Created Date, Last Modified Date, Recommendation Score, and Source ID.
4. To analyze a strategy's performance, group your strategy executions by recommendation source ID, and the number of times a recommendation was served, accepted, and rejected. To compare performance between two different strategies, group your strategy executions by recommendation source ID. Add useful metadata to your report, such as recommendation description and create date.
5. Deploy the report types you want to make available to users.
6. Let users know that they can create reports using these custom report types.

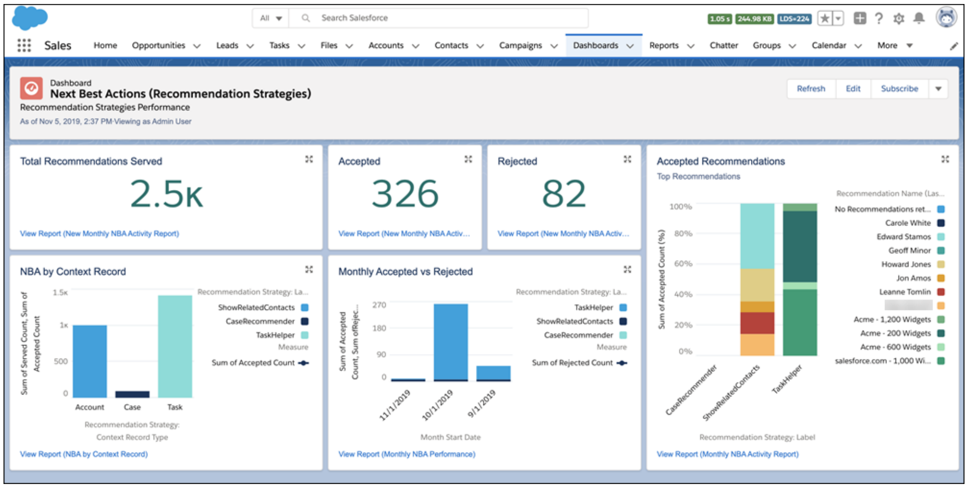


Report: Recommendations Reactions

Recommendation Reactions Report

Original Action Strategy Name ↑	Reaction Type ↓	Source: Name ↑	Name	Action Type	Context Record: Name ↑	Last Modified Date
Brembo_Brakes_Troubleshooter (2)	A (2)	Purchase Brake Discs From Brembo (1)	00000152	300	-	5/9/2019
		Troubleshooter (1)	00000204	300	-	6/24/2019
Bright_Bank_Community (1)	A (1)	Paperless Billing (1)	00000244	300	-	8/7/2019
BrightBankNBA (8)	A (8)	0% Refinancing! (1)	00000024	300	00001007	1/23/2019
		Collect Payment (1)	00000028	300	Lauren Boyle	1/24/2019
		Collect Payment Bright Bank (4)	00000029	300	Lauren Boyle	1/24/2019
			00000030	300	Lauren Boyle	1/24/2019
			00000031	300	Lauren Boyle	1/24/2019
			00000032	300	Lauren Boyle	1/24/2019
		Increase Credit Line Limit (1)	00000025	300	00001007	1/23/2019
		Transfer Accounts to Us (1)	00000026	300	00001007	1/23/2019
BrightBankNBA_AccountStart (1)	A (1)	Increase Credit Line Limit (1)	00000191	300	Acme Corporation	6/10/2019

7. Users can also create dashboards from the custom report type.



Dashboard
Next Best Actions (Recommendation Strategies)
Recommendation Strategies Performance
As of Nov 5, 2019, 2:37 PM·Viewing as Admin User

Total Recommendations Served: 2.5K
View Report (New Monthly NBA Activity Report)

Accepted: 326
View Report (New Monthly NBA Activ...

Rejected: 82
View Report (New Monthly NBA Activ...

SEE ALSO:

Create a Custom Report Type

Connect REST API Developer Guide: Recommendation Reaction

Automated Actions

An automated action is a reusable component that performs some sort of action behind the scenes—like updating a field or sending an email. After you create an automated action, add it to a process, milestone, or other automated process.

Action Type	Supported In...				
	Workflow Rule	Process	Flow	Approval Process	Entitlement Process
Email Alert	✓	✓	✓	✓	✓
Field Update	✓			✓	✓
Flow Trigger (Pilot)	✓ (immediate only)				
Outbound Message	✓			✓	✓
Task	✓			✓	✓

EDITIONS

Available in: both Lightning Experience and Salesforce Classic

Flow triggers are available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Outbound messages are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Email alerts are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Automated Actions

Before you start working with automated actions, familiarize yourself with relevant limits and special behaviors.

Manage Automated Actions in Workflow Rules

Task Actions

Task actions determine the details of an assignment given to a specified user by an automated process. You can associate task actions with workflow rules, approval processes, or entitlement processes.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

From Setup, enter `Tasks` in the `Quick Find` box, and select **Tasks**. Then use these settings to configure your task.

Field	Description
Object	Select an object for your task. Remember, tasks can only be associated with workflow rules or approval processes for the same object type. Choose an object that tracks activities. Tasks aren't available for article types.
Assigned to	Select an assignee for your task. An assignee can be in the form of a user, role, record owner, record creator, opportunity team role, or account team role, depending on the type of record you chose. • If the assignee of a task is set to the record owner and the owner of a lead or case is a queue, the task is assigned to the person who triggered the rule. • If a custom object has a master-detail relationship with a standard object, the owner of the custom object record is automatically set to the owner of the main standard object record. For example, if a custom object called "Expenses" has a master-detail relationship with Accounts, the owner of each expense record is the owner of the account for the expense. • If the assignee of a workflow task is a role and more than one user belongs to that role, the record owner becomes the task assignee, regardless of their role. We recommend that you don't assign tasks to roles with multiple users. Assigning tasks to roles with one user allows you to easily change the user in that role without modifying the workflow rule. If the assignee of a workflow task is a role and that role is empty, the record owner becomes the task assignee, regardless of their role. • When a lead is converted by someone who isn't the lead owner, all workflow tasks associated with the lead that are assigned to that user, except email alerts, are reassigned to the lead owner. Workflow tasks assigned to users other than the lead owner and lead converter aren't changed.
Subject	Enter a subject for the task. Distinguish automated tasks from user-created ones by starting the subject with a specific notation, such as adding `(Automated)` at the end.
Unique Name	Enter a unique name to refer to this component in the API. The requirement for uniqueness is only within the selected object type. You can have actions of the same type with the same unique name, provided they're defined for different objects.

Field	Description
	The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Due Date	Choose a due date for the task. Due dates appear in the time zone of the assignee. Configuring a task's **Due Date** to "Rule Trigger Date" sets time triggers and workflow task due dates based on the date that the workflow time trigger's action is executed. For example, if the task due date is "Rule Trigger Date plus 10 days" and the time trigger is executed on January 1, Salesforce sets the task due date to January 11.
Status	Choose a status for the task.
Priority	Choose a priority for the task.
Comments	Enter comments for the task.

Notice that all your tasks include a **Created By** field. For tasks, this field contains the name of the person who saved the record that triggered the rule to assign the task.

Tasks don't trigger task-based workflow rules if they're created automatically, such as by clicking the **Send An Email** button or by using the Email to Salesforce BCC address field.

SEE ALSO:

Associate Actions with Workflow Rules or Approval Processes

Email Alert Actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

An email alert is an email generated by an automated process and sent to the designated recipients. The action consists of the standard text and the list of recipients. You can use an email alert in an automation, such as a flow, approval process, or entitlement process. Legacy workflow rules and processes built in Process Builder or through the Invocable Actions REST API endpoint also use email alerts.

From Setup, enter `Email Alerts` in the Quick Find box, and select **Email Alerts**. Then use these settings to configure your email alert.

Tip: Create a standardized letterhead to use for all email templates that you use for email alert actions.

Field	Description
Description	Enter a description.
Unique Name	Enter a unique name to refer to this component in the API. The requirement for uniqueness is only within the selected object type. You can have actions of the same type with the same unique name, provided they're defined in different objects.The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Field	Description
Object	If available, choose an object for this email alert. Salesforce uses this object when generating merge field values for email templates. Also, you can define the recipients of this email alert using contact and user lookup fields that are relevant to that object. For example, if you select Contract, you can define the contract signer as a recipient. The object is read-only if the new email alert is associated with an automation for a particular object.
Email Template	Choose an email template. Insert merge fields to reference specific information based on the record that triggers the email alert. For example, insert a merge field for the opportunity. **Classic Email Templates** Except for `{!ApprovalRequest.Comments}`, approval merge fields named `{!ApprovalRequest.field_name}` in email templates return values only in approval assignment emails and email alerts for approval processes. When used in other emails—including email alerts for automations—the approval merge fields return null. The `{!ApprovalRequest.Comments}` merge field returns only the most recently entered comment in emails for an approval step that requires unanimous approval from multiple approvers. If available, select **Protected Component** to mark the alert as protected if it's part of a Managed - Released package. The Recipient merge field isn't supported in either Classic or Lightning email templates used for automations. **Lightning Email Templates** In Lightning email templates, merge fields are resolved for activity-enabled objects only, except Contact and Lead merge fields. Activity-enabled objects appear in the Related Entity Type field on the email template record home. Cross-object merge fields aren't supported. The object specified in the Related Entity Type field must be the object used in the template's merge fields. For a workaround, see Cross-object Merge Fields Do Not Work in Email Templates. There isn't equivalent support for special merge fields in Classic email templates, such as `{!this}` or `{!$Setup.LabelName}`. UTF-8 encoding is used on all emails sent through email alerts.
Recipient Type	Select who receives this email alert.
Recipients	Select who receives this email alert in the **Available Recipients** list and click **Add**. If you change the object after selecting recipients, Salesforce clears the **Selected Recipients** list. If your email recipient is a role and that role contains multiple people, Salesforce emails each person in that role. If your email recipient is a record owner and the owner of the record is a queue, the queue email receives the email alert. If the queue is set up so that email is sent to all members, queue members are notified as well. If no queue email is specified, only queue members are notified.
Additional Emails	Enter up to five more email addresses for recipients who aren't Salesforce users, leads, or contacts.
From Email Address	Either the default workflow user or a previously configured and verified organization-wide address. This field lets you use a standard email address for your organization (such as support@company.com) instead of the default **From** field, which is the email address of the person who updates the record. If the default workflow user is the guest user, you must have a verified organization-wide address in this field. If you select **Make this the default From email address for this object's email alerts**, this email address overrides the From Email Address for all email alerts associated with the object. You can still customize individual email alerts

Field	Description
	to use a different From Email Address. The From Email Address in an email alert changes to the current user when the email alert is installed using a managed or unmanaged package. The From Email Address doesn't change when using other types of deployment, such as Metadata API or change sets.

The daily allocation for emails sent through email alerts is 1,000 per standard Salesforce license per org—except for free Developer Edition and trial orgs, where the daily email allocation is 15. The overall org allocation is 2,000,000. This allocation applies to emails sent through email alerts in automations or REST API. Single emails sent to external email addresses are also limited, and how those limits are enforced depends on when your org was created.

SEE ALSO:

Recipient Types for Email Alerts

Daily Allocations for Email Alerts

Recipient Types for Email Alerts

When you configure an email alert, you identify who receives the email. The options available vary based on your Salesforce settings and the object that you selected.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Recipient Type	Description
Account Owner	The user listed as the owner of the account itself or the account associated with the record. This option works only for email alerts on accounts, opportunities, cases, contacts, contracts, and any custom object that is a child of the account object. The associated account must also specify an account owner. If you select another object type or the associated account doesn't have an account owner, Salesforce sends the email alert to the record owner instead.
Account Team	All users assigned to a particular account team role. The Account Team option is always available. However, emails are sent only when the rule is associated with the account object or its immediate child objects.
Case Team	All users assigned to a particular case team role.
Creator	The user who created the record.
Customer Portal User	All users associated with a Customer Portal.
Email Field	An email address field on the selected object, such as the Email field on lead records or custom email fields. When creating email alerts for campaign members, Email Field refers to the email field on the lead or contact that the campaign member is based on.

Recipient Type	Description
Opportunity Team	All users assigned to a particular opportunity team role. This option appears only when team selling is enabled. The Opportunity Team option works only for email alerts configured for opportunities. It doesn't work for email alerts configured for child objects of opportunities.
Owner	The record owner.
Partner User	All users associated with a partner portal.
Portal Role	All users assigned to a particular portal role.
Portal Role and Subordinates	All users assigned to a particular portal role, plus all users in roles below that role.
Public Groups	The users in a particular public group.
Related Contact	An associated contact on the record. For example, you can select the Customer Signed By field for contracts that contain the name of the contract signer.
Related Lead or Contact Owner	A campaign member's lead or contact owner.
Related User	An associated user on the record. For example, contract records have an Activated By field that contains the name of the user that activated the contract.
Role	All users assigned a particular role.
Role and Internal Subordinates	All users assigned a particular role, plus all users in roles below that role, excluding partner portal and Customer Portal users.
Role and Subordinates	All users assigned a particular role, plus all users in roles below that role.
User	A particular user.

The Recipient merge field isn't supported in either Classic or Lightning email templates used for automations.

Field Update Actions

Field update actions let you automatically update a field value. You can associate field updates with workflow rules, approval processes, or entitlement processes.

Important: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

From Setup, enter *Field Updates* in the `Update` box, and select **Field Updates**. Then use these settings to configure your field update.

Before you begin, check the type of the field you want to update. Read-only fields like formula or auto-number fields aren't available for field updates.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
Name	Enter a name for this field update.
Unique Name	Enter a unique name to refer to this component in the API. The requirement for uniqueness is only within the selected object type. You can have field updates of the same type with the same unique name, provided they're defined in different objects. The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Enter a description for the field update.
Object	Select the object whose field you want to update.
Field to Update	Select the field to update. Fields are shown only for the object that you selected. You can select a field on a related object in a master-detail relationship. You can use field updates on encrypted custom fields, but the encrypted field isn't available in the formula editor. Avoid associating more than one field update with a rule or approval process that applies different values to the same field.
Re-evaluate Workflow Rules After Field Change	Select if you want workflow rules on this object to be reevaluated after the field value is updated. If you select this option, Salesforce reevaluates all workflow rules on the object if the field update results in a change to the value of the field, triggering any workflow rules whose criteria are met. For more information, see Field Updates That Reevaluate Workflow Rules on page 826.
Specify New Field Value	The value that the field can be updated with. The available options depend on the type of field you're updating. For more information, see Value Options for Field Update Actions on page 817 .

SEE ALSO:

Associate Actions with Workflow Rules or Approval Processes

Cross-Object Field Updates

Considerations for Field Update Actions

Value Options for Field Update Actions

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When you create a field update action, specify the new value of the field.

Available field update options depend on the type of field you're updating.

- Choose **A specific value**, and enter the value in the space provided.
- Choose **A blank value (null)** if you want Salesforce to remove any existing value and leave the field blank. This option isn't available for required fields, checkboxes, and some other types of fields.
- For record owners, choose a user to assign to the record. For case, lead, and custom object records, you can also choose a queue for this field. To send an email to the new record owner, select `Notify Assignee`. (This option is unavailable when user control over task assignment notifications is enabled.)
- For checkboxes, choose `True` to select the checkbox and `False` to deselect it.

- For picklists, select a specific value from the dropdown list, or select the value above or below the current value based on the sorting specified in the picklist definition. If you sort values alphabetically, the values above or below the current value can be different for users in other languages.
- To calculate the value based on an expression, merge fields, or other values, select **Use a formula to set the new value**. For more information about using formulas in Salesforce, see Calculate Field Values with Formulas.

Outbound Message Actions

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

An outbound message sends information to a designated endpoint, like an external service. You configure outbound messages from Setup. You must configure the external endpoint and create a listener for the messages using SOAP API. You can associate outbound messages with flows, workflow rules, approval processes, or entitlement processes.

Note: Previously, outbound messages were available in Professional Edition with the purchase of an add-on. The add-on is no longer available for Professional Edition.

For example, automatically initiate the reimbursement process for an approved expense report by triggering an outbound API message to an external HR system.

From Setup, in the Quick Find box, enter `Outbound Messages`, and then select **Outbound Messages**. Then use these settings to configure your outbound message.

Field	Description
Object	Choose the object that has the information you want included in the outbound message
Name	Enter a name for this outbound message.
Unique Name	Enter a unique name to refer to this component in the API. The requirement for uniqueness is only within the selected object type. You can have outbound messages with the same unique name, provided they're defined for different objects. The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Enter a description that makes it easy for other users to tell what the outbound message does.
Endpoint URL	Enter an endpoint URL for the recipient of the message. Salesforce sends a SOAP message to this endpoint.
User to send as	Select the Salesforce user to use when sending the message. The chosen user controls data visibility for the message that is sent to the endpoint
Protected Component	If present, select **Protected Component** to mark the outbound message as protected in a managed package. Only available in Developer Edition.
Send Session ID	To include the Salesforce session ID in the outbound message, select **Send Session ID**. Include the session ID in your message if you intend to make API calls and you don't want to include a username and password. Never send a username and password in an unencrypted message, especially in a production environment. It isn't secure. When you select **Send Session ID**, only HTTPS is supported for the endpoint URL to ensure secure transmission of the session ID. For managed and unmanaged packages created before Spring '19 with this option but without an HTTPS endpoint, subscribers can still install them. Starting in Spring '19, you can't create packages with insecure outbound message options.

Field	Description
`Object` fields to send	Select the fields to include in the outbound message, and click **Add**.

If your endpoint URL uses a client certificate, see Import a Client Certificate for Your Endpoint URL on page 821.

SEE ALSO:

- Track the Delivery Status of an Outbound Message
- Considerations for Outbound Messages
- SOAP API Developer Guide
- Associate Actions with Workflow Rules or Approval Processes
- Considerations for Outbound Messages

Outbound Message Notifications

You can request that up to 5 users receive a notification listing all outbound messages that have failed for at least 24 hours. A fresh notification is sent every 24 hours until you cancel the request. Failed messages are deleted from the failed outbound messages related list after 7 days. Before they're removed, you can delete them yourself or request that they be retried again.

Note: Previously, outbound messages were available in Professional Edition with the purchase of an add-on. The add-on is no longer available for Professional Edition. If outbound messages are available in your Salesforce edition but you don't see the Outbound Message Notifications page, your org doesn't have notifications for outbound messages enabled. Contact Salesforce to enable notifications for outbound messages.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Create an Outbound Message Notification

You can request that up to five users receive a notification listing all outbound messages that have failed for at least 24 hours. A fresh notification is sent every 24 hours until you cancel the request.

View an Outbound Message Notification Request

View or edit outbound message notification requests.

Create an Outbound Message Notification

You can request that up to five users receive a notification listing all outbound messages that have failed for at least 24 hours. A fresh notification is sent every 24 hours until you cancel the request.

Note: If you don't see the Outbound Message Notifications page, your org doesn't have notifications for outbound messages enabled. Contact Salesforce to enable notifications for outbound messages.

1. From Setup, enter `Outbound Message Notifications` in the Quick Find box, then select **Outbound Message Notifications**.
2. Click **New**.
3. Enter a full username, or click the icon to select it from a list of usernames.
4. Save the request.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create an outbound message notification:

- Modify All Data

View an Outbound Message Notification Request

View or edit outbound message notification requests.

From the detail page of an outbound message notification request:

- To change the username for a notification request, click **Edit**. It's simpler than deleting the request and then creating a one.
- To delete the notification request, click **Delete**.
- To create a notification request with the same username, click **Clone**.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view or edit outbound message notification request:

- Modify All Data

Track the Delivery Status of an Outbound Message

To track the status of an outbound message, from Setup, enter `Outbound Messages` in the `Quick Find` box, then select **Outbound Messages**.

- *Next items for delivery* are awaiting delivery.
- *Oldest failures* haven't yet been deleted because they haven't been delivered and aren't 24 hours old.
- *Failed outbound messages* failed to be delivered and are no longer being retried. Messages are listed here only if you configure the message when you create it by selecting `Add failures to failed outbound message related list`. If you don't see this related list, it hasn't been enabled for your organization.

You can perform several tasks here.

- To view the action that triggered it, click any workflow or approval process action ID.
- To change the **Next Attempt** date to now, click **Retry**. This option causes the message delivery to be immediately retried. If you select **Retry** in the **Failed outbound messages** related list, the outbound message moves to the **Next items for delivery** related list and is retried for another 24 hours.
- To permanently remove the outbound message from the queue, click **Del**.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To track outbound messages:

- Modify All Data

Note: If you don't have this option, your org doesn't have outbound messages enabled. Contact Salesforce to enable outbound messages.

Import a Client Certificate for Your Endpoint URL

If the endpoint URL of your outbound message uses a client certificate, import it to put your outbound message into action.

1. From Setup, enter `API` in the `Quick Find` box, then select **API**
2. Click **Generate Client Certificate**.
3. Save the certificate to the appropriate location.
4. Import the downloaded certificate into your application server and configure your application server to request the client certificate.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Automated Actions

Before you start working with automated actions, familiarize yourself with relevant limits and special behaviors.

Considerations for Field Update Actions

Learn how to use field update actions to their full potential in workflow.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Outbound Messages

Review the considerations for using outbound message actions before implementing them in your workflows.

SEE ALSO:

Daily Allocations for Email Alerts

Considerations for Field Update Actions

Learn how to use field update actions to their full potential in workflow.

[other]: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

When creating field updates for workflow rules or approval processes, consider the following:

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field Update Processing

- Field updates occur before email alerts, tasks, and outbound messages.
- Field updates occur after case assignment, lead assignment, and auto-response rules.
- Field updates function independently of field-level security. Therefore, a workflow rule can update fields even though they're hidden on the user's page layout.
- The result of a field update is unpredictable when a single workflow rule includes multiple field updates that apply different values to the same field.
- Field updates can affect the information in a related list. For example, if a field such as the `Amount` or `Close Date` of an opportunity is set to be updated, it affects the Stage History related list on opportunities.
- If a user gets a field update error when saving a record, you can use the debug log to see which field update failed. The debug log stops when a failure occurs.
- For reminder fields on tasks and events:
  - Field updates can set the reminder for a task or event but they can't use the due date of a task or the scheduled time of an event.
  - Formulas for date/time values are calculated in days. Divide the value by 1440—the number of minutes in a day—to express the value in minutes. For example, the formula `Now()-7` means seven *days* ago, while `Now()-7/1440` means seven *minutes* ago.
- If your organization uses multiple currencies, currency fields are updated using the record's currency. If you choose to update a field based on a formula, any values in your formula are interpreted in the currency of the record.
- Field updates are tracked in the History related list if you have set history tracking on those fields.
- Workflow rules and some processes can invalidate previously valid fields. Invalidation occurs because updates to records based on workflow rules and also on process-scheduled actions don't trigger validation rules.
- If you have person accounts enabled, you can use the `Is Person Account` field as part of the evaluation criteria for workflow rules. However, because the `Is Person Account` field is read-only, any field updates set up to modify it fails.

Tip: Salesforce processes rules in the following order:

- Validation rules
- Assignment rules
- Auto-response rules
- Workflow rules (with immediate actions)

- Escalation rules

Notes on Cross-Object Field Updates

- For all custom objects and some standard objects, you can create workflow and approval actions where a change to a detail record updates a field on the related main record. Cross-object field updates work for custom-to-custom master-detail relationships, custom-to-standard master-detail relationships, and a few standard-to-standard master-detail relationships. For more information, see Cross-Object Field Updates on page 11.
- Approval processes can't use cross-object field update actions.
- An approval process can specify a field update action that reevaluates workflow rules for the updated object. If, however, the reevaluated workflow rules include a cross-object field update, those cross-object field updates are ignored.
- To create workflow rules so that case comments or emails automatically update fields on associated cases, select **Case Comment** or **Email Message** in the Object dropdown list when creating a workflow rule and select **Case** in the Field to Update list. Email-to-Case or On-Demand Email-to-Case must be enabled for your organization to use the Email Message in a workflow rule.

  When cases are updated by an email-triggered workflow rule, the updated case can trigger:

  - Workflow rules
  - Validation rules
  - Updates to roll-up summary fields
  - Escalation rules
  - Apex triggers
  - Entitlement processes

  The updated case can't trigger:

  - Assignment rules
  - Auto-response rules

Field Update Actions and Custom Fields

- Before changing a custom field’s type, make sure it isn’t the target of a workflow field update or referenced in a field update formula that’s invalidated by the new type.
- You can't delete a custom field that is referenced by a field update.
- You can use field updates on encrypted custom fields, but if you try to use a formula to set the new value, the encrypted field isn't available in the formula editor.

Field Update Actions on Opportunities and Contracts

- You can define field updates for the `Stage` field on opportunities, but be aware of how this field affects the `Type` and `Forecast Category` fields.
- You can define field updates using the `Amount` field on opportunities but it only applies to those opportunities that don't have products. Adding products to an opportunity changes the `Amount` field to a read-only field that is automatically calculated and not affected by that field update.
- You can define field updates for the `Status` field on contracts. However, the value of this field can affect the value of the `Status Category` field as well.
- Avoid creating a field update for contracts or orders that changes the `Status` field to any value other than Approved.

Field Update Action Limitations

- The results of a field update can't trigger additional rules such as validation, assignment, auto-response, or escalation rules.
- The results of a field update can trigger additional workflow rules if you've flagged the field update to do so. For more information, see Field Updates That Reevaluate Workflow Rules on page 826.
- Field updates that are executed as approval actions don't trigger workflow rules or entitlement processes.
- These fields aren't available for field update actions:
  - Read-only fields like formula or auto-number fields
  - The `Language` picklist field on multilingual solutions
  - Some activity fields, such as `Related To` and `Private`
- Email message workflow rules can only be associated with field updates.
- If a field update references a specific user, you can't deactivate that user. For example, if your field update is designed to change the owner of a record to Bob Smith, change the field update before deactivating Bob Smith.
- You can update long text area fields, but the option to insert `A specific value` restricts you to entering up to the maximum number of characters allowed in the destination field.
- You can't make a field universally required if it's used by a field update that sets the field to a blank value.
- Workflow rules that update owners *don't* also transfer associated items. To ensure transfer, click **Change** next to the owner's name in a record and make your transfer selections.

Cross-Object Field Updates

For all custom objects and some standard objects, you can create actions where a change to a detail record updates a field on the related main record. Cross-object field updates work for custom-to-custom master-detail relationships, custom-to-standard master-detail relationships, and a few standard-to-standard master-detail relationships.

Field Updates That Reevaluate Workflow Rules

If `Re-evaluate Workflow Rules After Field Change` is enabled for a field update action, and a field update results in a change to the value of the field, Salesforce reevaluates all workflow rules on the object.

SEE ALSO:

Cross-Object Field Updates

Cross-Object Field Updates

For all custom objects and some standard objects, you can create actions where a change to a detail record updates a field on the related main record. Cross-object field updates work for custom-to-custom master-detail relationships, custom-to-standard master-detail relationships, and a few standard-to-standard master-detail relationships.

[other]: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

For example, in a custom recruiting application, create a workflow rule that sets the status of an application (the main object) to "Closed" when a candidate (the detail object) accepts the job. Or, for standard objects, create a rule to change the status of a case from "Awaiting Customer Response" to "In Progress" when a customer adds a case comment.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Custom Object to Custom Object

Cross-object field updates are supported for all custom objects that are children of custom objects in a master-detail relationship.

Custom Object to Standard Object

Cross-object field updates are supported for custom objects that are children of certain standard objects in a master-detail relationship. The standard objects that support cross-object field updates from custom objects are:

- Account
- Asset
- Campaign
- Case
- Contact
- Contract
- Contract Line Item
- Entitlement
- Opportunity
- Order
- Question
- Quote
- Service Contract
- Solution

Standard Object to Standard Object

Cross-object field updates are supported for standard objects that are children of standard objects in a master-detail relationship. However, only these standard-to-standard relationships are supported.

Note: If you have workflow rules on converted leads and want to use cross-object field updates on the resulting accounts and opportunities, you must enable the lead setting `Require Validation for Converted Leads`.

- Case Comments updating Case
- Email updating Case

  Tip: To create workflow rules so that case comments or emails automatically update fields on associated cases, select **Case Comment** or **Email Message** in the Object dropdown list when creating a workflow rule and select **Case**in the Field to Update list. Email-to-Case or On-Demand Email-to-Case must be enabled for your organization to use the Email Message in a workflow rule.

- Opportunity Product updating Opportunity

  Note: Cross-object field updates to a parent opportunity's `Amount` and `Quantity` fields only work if the opportunity has no opportunity products associated with it.

- Opportunity updating Account—Supported for both business accounts and person accounts.

Standard-to-standard cross-object field update actions:

- Can’t be used in, or assigned to, approval processes.
- Update a parent record even if the user doesn’t have edit access to it.

Note: If you have Apex code that updates parent fields in the same relationships as a cross-object field update action, consider replacing your code with cross-object field updates. Otherwise, both will fire, and since workflow rules run after Apex triggers, the workflow field update will override any change made by your Apex code.

SEE ALSO:

Considerations for Field Update Actions

Object Relationships Overview

Field Updates That Reevaluate Workflow Rules

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

If `Re-evaluate Workflow Rules After Field Change` is enabled for a field update action, and a field update results in a change to the value of the field, Salesforce reevaluates all workflow rules on the object.

- If the field update changes the field's value, all workflow rules on the associated object are reevaluated. Any workflow rules whose criteria are met as a result of the field update are triggered.
- If any of the triggered workflow rules result in another field update that's also enabled for workflow rule reevaluation, a domino effect occurs, and more workflow rules can be reevaluated as a result of the newly triggered field update. This cascade of workflow rule reevaluation and triggering can happen up to five times after the initial field update that started it.
- Make sure that your workflow rules aren't set up to create recursive loops. For example, if a field update for Rule1 triggers Rule2, and a field update for Rule2 triggers Rule1, the recursive triggers can cause your organization to exceed its limit for workflow time triggers per hour.
- In a batch update, workflow is only retriggered on the entities where there's a change.
- Only workflow rules on the same object as the initial field update are reevaluated and triggered.
- Only workflow rules that didn't fire before are retriggered.
- Cross-object workflow rules aren't candidates for reevaluation.
- Cross-object field updates that cause a field value to change don't trigger workflow rule reevaluation on the associated object.
- An approval process can specify a field update action that reevaluates workflow rules for the updated object. If, however, the reevaluated workflow rules include a cross-object field update, those cross-object field updates are ignored.
- Time-dependent actions aren't executed for a reevaluated workflow rule in the following situations:
  - The reevaluated workflow rule's immediate actions cause the record to no longer meet the workflow rule criteria.
  - An Apex `after` trigger that is executed as a result of a workflow or approvals action causes the record to no longer meet the workflow rule criteria.

SEE ALSO:

Considerations for Field Update Actions

Considerations for Outbound Messages

Review the considerations for using outbound message actions before implementing them in your workflows.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When creating outbound messages for workflow rules or approval processes, keep these considerations in mind.

- A single SOAP message can include up to 100 notifications. Each notification contains an ID that uniquely identifies a record, and a reference to the data in the record. If the information in the record changes after the notification is sent, but before the notification is delivered, only the updated information is delivered. If the record is deleted before the notification is delivered, the notification contains no data.
- Messages are queued until they're sent, to preserve message reliability.
- If the endpoint is unavailable, messages stay in the queue until sent successfully or until they're 24 hours old. After 24 hours, messages are dropped from the queue.
- If a message can't be delivered, the interval between retries increases exponentially, up to a maximum of two hours between retries.
- Messages are retried independent of their order in the queue, which can result in messages being delivered out of order.
- A message can be delayed by other, long-running messages in the queue. The queue can also contain messages that originate from other Salesforce orgs that are hosted on the same Salesforce instance. The system attempts to optimize the execution of messages that historically have fast run times so that they aren't delayed by slow-running messages. To get the best performance, make sure that the message endpoint runs efficiently. For slow-running messages, consider using asynchronous processes, such as platform events or Apex future methods.
- You can't build an audit trail using outbound messages. While each message is delivered at least one time, it can be delivered more than one time. Also, if delivery can't be done within 24 hours, the message doesn't get delivered at all. Finally, as noted above, the source object can change after a notification is sent but before it's delivered, so the endpoint will only receive the latest data, not any intermediate changes.

SEE ALSO:

Platform Events Developers Guide

Apex Developer Guide: Future Annotation

Approval Processes

It's likely that you're familiar with process automation in the form of workflow rules. Approval processes take automation one step further, letting you specify a sequence of steps that are required to approve a record.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

An approval process automates how records are approved in Salesforce. An approval process specifies each step of approval, including from whom to request approval and what to do at each point of the process.

Example: Your org has a three-tier process for approving expenses. This approval process automatically assigns each request to right person in your org, based on the amount requested.

If an expense record is submitted for approval, lock the record so that users can't edit it and change the status to Submitted.

If the amount is $50 or less, approve the request. If the amount is greater than $50, send an approval request to the direct manager. If the amount is greater than $5,000 and the first approval request is approved, send an approval request to the vice president.

If all approval requests are approved, change the status to Approved and unlock the record. If any approval requests are rejected, change the status to Rejected and unlock the record.

Set Up an Approval Process
If Approvals is the right automation tool for your business process, follow these high-level steps to create one for your org.

Prepare Your Org for Approvals
Make sure that your users can submit their records for approval, and consider how you can make it easy for approvers to respond to approval requests.

Limits and Considerations for Approvals
Before you automate something with an approval process, be aware of the limits and considerations.

Sample Approval Processes
Review samples of common approval processes to help you get started creating your own.

Approval History Reports
If you create a custom report type for approval process instances, users can view the historical details of completed and in-progress approval processes and their individual steps.

Manage Multiple Approval Requests
Transfer multiple approval requests from one user to another or remove multiple approval requests from the approval process.

Approval Requests for Users
Your admin can set up approval processes that let you and other users submit records for approval, which results in *approval requests*.

Approval Process Terminology
Salesforce uses this terminology for approval processes.

Set Up an Approval Process

If Approvals is the right automation tool for your business process, follow these high-level steps to create one for your org.

1. Prepare to Create an Approval Process
   Plan each approval process carefully to ensure a successful implementation.
2. Choose the Right Wizard to Create an Approval Process
   Before you create an approval process, determine which wizard is best for your needs.
3. Add an Approval Step to an Approval Process
   Approval steps define the chain of approval for a particular approval process. Each step determines which records can advance to that step, who to assign approval requests to, and whether to let each approver's delegate respond to the requests. The first step specifies what to do if a record doesn't advance to that step. Later steps specify what happens if an approver rejects the request.
4. Add Automated Actions to an Approval Process
   You can associate actions to approval steps, initial submission, final approval, final rejection, or recall. Approval processes support four automated actions.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval processes:
- Customize Application

5. Activate an Approval Process
   After you've created at least one step for the approval process, activate the process.

SEE ALSO:

Approval Process Terminology

Sample Approval Processes

Prepare Your Org for Approvals

Prepare to Create an Approval Process

Plan each approval process carefully to ensure a successful implementation.

Review the following checklist before creating your approval process.

- Prepare an approval request email template.
- Prepare an approval request post template.
- Determine the approval request sender.
- Determine the assigned approver.
- Determine the delegated approver.
- Decide if your approval process needs a filter.
- Design initial submission actions.
- Decide if users can approve requests from a wireless device.
- Determine if users can edit records that are awaiting approval.
- Decide if records should be auto-approved or rejected.
- Determine how many levels your process has.
- Determine the actions when an approval request is approved or rejected.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Which email template do you want to use for approval requests?

The email template you specify on an approval process is used when notifying users that an approval request is assigned to them. You can use the Salesforce default email template or create your own template. Include the appropriate approval process merge fields to link directly to the approval request. Does your org have email approval response enabled? If so, the default email template includes instructions for replying to an approval request. Type `approve`, `approved`, `yes`, `reject`, `rejected`, or `no` in the first line of the email body and add comments in the second line.

Which Chatter post template do you want to use for approval requests?

If your org has Approvals in Chatter enabled, specify an approval post template to use when notifying a user via Chatter about an assigned approval request. You can use the Salesforce default post template or create your own.

Who is the sender of approval requests?

Approval request notifications are sent from the user who submitted the record for approval. When you configure an email alert, you can add a different return email address for these notifications. You can choose the email address of the default workflow user or a previously configured and verified org-wide address. Determine which email address to use.

Who can approve requests?

Any of the following can approve or reject a request.

- A user or queue that the approval request submitter chooses.
- A queue specified by the administrator.
- A user listed in the `Manager` standard field on the submitter's user detail page.
- A user listed in a custom hierarchy field on the submitter's user detail page.
- Any combination of users and related users (users listed in a standard or custom field on the submitted record) specified by the administrator.

Do you want approval requests delegated to another user for approval?

An approver can designate a delegate to approve requests, but you can disable this option. To assign delegates, populate the `Delegated Approver` field for each user's detail page.

Note: Internal Salesforce users are listed by and can be added using the Delegated Approver lookup field. Use Data Loader and a comma-delineated (CSV) file to add users with communities licenses as Delegated Approvers. The CSV uses the `CommunityUserId` rather than the `UserId` for `DelegatedApproverId`. Communities licenses are used with Experience Cloud sites and legacy portals.

Which records are included in this process?

Determine what attributes a record must have to be included in your approval process. If necessary, create the custom fields to store this information so that you can use it in your filter criteria. For example, if you want to include expense records from your headquarters office only, create a custom picklist field called `Office Location` that has two options: "HQ" and "Field." Then, you would specify in your filter criteria that records must have "HQ" in the `Office Location` field to be included in the approval process.

What happens when a record is first submitted for approval?

When users submit a record for approval, Salesforce automatically locks the record so that other users can't change it while it's awaiting approval. You can still add campaign members to campaigns locked for approval.

Decide if you want other workflow actions to happen when a record is first submitted, such as email alerts, tasks, field updates, and outbound messages. These actions become your initial submission actions.

Can users approve requests from a mobile device?

Determine if you want to require users to log in to Salesforce to approve requests. You can also set up your approval process to allow users to approve requests remotely using a mobile browser.

Who can edit records that are awaiting approval?

Records submitted for approval are locked. Users with the "Modify All" object-level permission for the given object or the "Modify All Data" permission can always unlock a record and edit it. You can also specify that the currently assigned approver can edit the record. You can still add campaign members to campaigns locked for approval.

Can records be automatically approved, rejected, or skipped based on certain criteria?

You can set entry criteria for each step of your process. Configure Salesforce to approve, reject, or skip the process if a record doesn't meet the criteria. For example, all expenses submitted with an `Amount` less than $15 are automatically approved.

How many people have to approve these requests?

An approval process can have several layers of approvals. Determine how many users have to approve requests and in what order.

What happens when a request is approved or rejected?

When a request is recalled, approved, or rejected, Salesforce can perform up to 10 instances of each of the following types of actions—up to 40 actions total. You can also configure up to 40 actions to occur when a record has received all necessary approvals or is rejected.

SEE ALSO:

- Set Up an Approval Process
- Limits and Considerations for Approvals
- Sample Approval Processes

Choose the Right Wizard to Create an Approval Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Before you create an approval process, determine which wizard is best for your needs.

Create an Approval Process with the Jump Start Wizard
For approval processes that use a single step, use the jump start wizard. This wizard chooses some default options for you.

Default Selections for the Approval Process Jump Start Wizard
To make it easier for you to get started with a simple approval process, the jump start wizard automatically chooses some default options for you.

Create an Approval Process with the Standard Wizard
When your approval process is more complex and you want to define specific steps, use the standard wizard.

SEE ALSO:

- Set Up an Approval Process

Create an Approval Process with the Jump Start Wizard

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval processes:
- Customize Application

For approval processes that use a single step, use the jump start wizard. This wizard chooses some default options for you.

1. From Setup, enter `Approval Processes` in the `Quick Find` box, then select **Approval Processes**.
2. Select an object.
3. Select **Create New Approval Process** > **Use Jump Start Wizard**.
4. Configure the approval process by following the wizard.
   a. Default Selections for the Approval Process Jump Start Wizard
   b. Choose Approval Request Notification Templates
   c. Design the Approval Request Page
   d. Control Which Records Apply to an Approval Process

e. Identify Assigned Approvers for an Approval Step

SEE ALSO:

Default Selections for the Approval Process Jump Start Wizard

Considerations for Configuring Approvals

Considerations for Setting Approvers

Set Up an Approval Process

Choose the Right Wizard to Create an Approval Process

Default Selections for the Approval Process Jump Start Wizard

To make it easier for you to get started with a simple approval process, the jump start wizard automatically chooses some default options for you.

After creating an approval process using the jump start wizard, you can modify these default options and add more steps from the approval process detail page. Exception: you can't modify the Record Lock action on the Initial Submission Actions list.

- To edit records awaiting approval in the approval process, users must have the "Modify All" permission for the given object or the Modify All Data permission.
- The page layout for the approval request includes the record name (or number), owner, date created, and approval history.
- The security settings require approvers to log in to Salesforce to view the approval page.
- Only the owner of the record can submit the record for approval.
- Records are locked when submitted for approval.
- Records remain locked until approved or rejected.
- Rejected records are unlocked.
- Only admins can recall a record after it's submitted.
- There are no auto-approve or auto-reject actions.
- No email notification is sent upon approval or rejection.
- No field values are automatically updated during the approval process.
- An approver can't automatically delegate another user to approve the approval requests.
- The **Allow submitters to recall approval requests** option isn't selected.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Create an Approval Process with the Jump Start Wizard

Choose the Right Wizard to Create an Approval Process

Create an Approval Process with the Standard Wizard

When your approval process is more complex and you want to define specific steps, use the standard wizard.

From Setup, enter `Approval Processes` in the `Quick Find` box, then select **Approval Processes**.

Select an object, and then select **Create New Approval Process** > **Use Standard Setup Wizard**. Configure the approval process.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval processes:

- Customize Application

1. Control Which Records Apply to an Approval Process

   Narrow down the list of records that can be part of the approval process by specifying criteria. You can either use filters or write a formula.
2. Choose Approval Request Notification Templates

   When an approval process assigns an approval request to a user, Salesforce sends the user an approval request email. If Approvals in Chatter is enabled, Salesforce also posts the approval request to Chatter. Choose templates for each of these notifications.
3. Choose an Automated Approver Throughout an Approval Process

   Associate a hierarchy field—such as the user's manager—with an approval process. When selected, the field is available as an assigned approver option for approval steps. You can always select a hierarchy field here but not use it for any approval steps.
4. Specify Who Can Edit Locked Records

   When a record is submitted for approval, it's locked to prevent users from editing it during the approval process. Use the record editability properties to determine who can edit records that are locked in this approval process.
5. Design the Approval Request Page

   The approval page is where an approver responds to an approval request. Customize which fields appear on that page and in which order. This page is used only for this approval process.
6. Specify Who Can Submit Records to an Approval Process

   Only specified individuals or roles can submit a record for approval. You can also let submitters recall an approval request.

SEE ALSO:

Set Up an Approval Process

Limits and Considerations for Approvals

Control Which Records Apply to an Approval Process

Narrow down the list of records that can be part of the approval process by specifying criteria. You can either use filters or write a formula.

If you want all records to pass through the approval process, click Next. If only certain types of records are considered, use one of the following options.

Option	To enter the approval process...
criteria are met	The record must meet the filter criteria.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Option	To enter the approval process...
formula evaluates to true	The formula must return `True`. Some functions aren't available in approval process formulas. For information on which functions you can use in approval process formulas, see Formula Operators and Functions by Context.

Example: This filter lets an expense report enter this approval process only if the employee who submitted the report is at headquarters.

```
Current User: Office Location Equals Headquarters
```

This formula lets a record enter this approval process only if its discount approval cutoff date is less than 30 days away.

```
(Discount_Approval_CutoffDate__c < (CloseDate - 30)
```

SEE ALSO:

Considerations for Configuring Approvals

Formula Operators and Functions by Context

Choose Approval Request Notification Templates

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When an approval process assigns an approval request to a user, Salesforce sends the user an approval request email. If Approvals in Chatter is enabled, Salesforce also posts the approval request to Chatter. Choose templates for each of these notifications.

These fields are available from both the jump-start and standard wizards.

Field	Description
Approval Assignment Email Template	Choose a custom email template to use when notifying approvers that an approval request is assigned to them. Or leave blank to use the default email template. The approval process uses the same template for every assignment email—no matter which approval step it's for.
Approval Post Template	Available only when Approvals in Chatter is enabled. Choose an approval post template to use when notifying approvers via a post in their Chatter feed. Leave blank to use the default post template for this object or, if there isn't one, the system default template.

Note: If email approval response is enabled, be sure that the email template you use describes how to correctly use both response options: clicking the link and replying by email. If the user doesn't respond correctly (for example, if the user misspells `approve` or types it on the wrong line), Salesforce doesn't register the user's response.

SEE ALSO:

- Chatter Post Templates for Approval Requests
- Email Templates in Salesforce Classic
- Merge Fields for Approvals

Choose an Automated Approver Throughout an Approval Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Associate a hierarchy field—such as the user's manager—with an approval process. When selected, the field is available as an assigned approver option for approval steps. You can always select a hierarchy field here but not use it for any approval steps.

Set **Next Automated Approver Determined By** with one of the following options.

Option	Description
--None--	Approval requests aren't automatically assigned based on a field. Instead, you manually specify a user to approve all approval requests.
`Field`	Approval requests are assigned to an approver from the specified field. You can select only a hierarchical relationship field, such as `Manager`.
Use Approver Field of *`Object`* Owner	Available only when *`Field`* is selected. If selected, the first executed approval step sets the approver to the value of *`Field`* on the record owner's user record—instead of the submitter's user record. All remaining steps use *`Field`* in the user record of the preceding step's approver.

Example: If you select the `Manager` field, you can configure any step in this process to route approval requests to the submitting user's manager.

If you select **Use Approver Field of *`Object`* Owner**, the first step that isn't skipped is routed to the owner's manager. All other steps are routed to the previous approver's manager.

SEE ALSO:

- Custom Field Types
- Considerations for Setting Approvers

Specify Who Can Edit Locked Records

When a record is submitted for approval, it's locked to prevent users from editing it during the approval process. Use the record editability properties to determine who can edit records that are locked in this approval process.

Option	Description
Administrators ONLY...	Default. Lets users edit the record that's pending approval only if they have: - The "Modify All" object-level permission for the given object, or - The "Modify All Data" permission
Administrators OR...	Lets the assigned approver and admins edit the record.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note:

- Even when a campaign is locked for approval, users can add campaign members to it.
- In Lightning Experience, you can't unlock Knowledge articles during an approval process.

Design the Approval Request Page

The approval page is where an approver responds to an approval request. Customize which fields appear on that page and in which order. This page is used only for this approval process.

Option	Description
Selected Fields	Specifies which fields to display on the approval request page. Keep in mind that approvers could view this page on a mobile device. Select only the fields necessary for users to decide whether to approve or reject records.
Display approval history information...	If selected, the approval request page displays the approval history of the associated record.
Security Settings	Controls whether users have to log in to Salesforce to see the approval request. **Allow approvers to access the approval page only from within the application (Recommended)** Default. Users log in to Salesforce to view the approval page. **Allow approvers to access the approval page only from within the application, or externally from a wireless-enabled mobile device** Users can access an external version of the approval page from a browser, including browsers on mobile devices, without logging in to Salesforce. If selected, you can't add approval steps that let users manually select the next approver.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Specify Who Can Submit Records to an Approval Process

Only specified individuals or roles can submit a record for approval. You can also let submitters recall an approval request.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Initial Submitters

Submitter Type	Select a type or search to populate the Available Submitters list.
Allowed Submitters	If the user who submits a record for approval isn't included in this list, the record doesn't enter this approval process—even if the record meets the entry criteria.

Page Layout Settings

Add the Submit for Approval button and Approval History related list to all `Object` **page layouts**	When selected, Salesforce adds the Submit for Approval button to the Standard Buttons and Approval History related list to the Related Lists for all page layouts for the object. This setting is available only when you create an approval process. If the Standard Buttons for a layout haven't been customized before, the Submit for Approval button isn't added.

Submission Settings

Allow submitters to recall approval requests	If selected, submitters can recall their approval requests. If unselected, only admins can recall requests. This option is useful for situations where things can change on the submitter's side while waiting for an approval. For example, an opportunity could be lost after the user submits it for approval.

Add an Approval Step to an Approval Process

Approval steps define the chain of approval for a particular approval process. Each step determines which records can advance to that step, who to assign approval requests to, and whether to let each approver's delegate respond to the requests. The first step specifies what to do if a record doesn't advance to that step. Later steps specify what happens if an approver rejects the request.

You can add steps to an approval process only if it's inactive.

From the approval process, click **New Approval Step**, and follow the wizard.

Steps are executed in the order specified.

1. Control Which Records Apply to an Approval Step
   Control which records are part of the approval step by setting the step's criteria. You can also specify what happens to records that don't meet the step's criteria.
2. Identify Assigned Approvers for an Approval Step
   Specify who to send an approval request for this step to.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval steps:
- Customize Application

3. Specify Rejection Behavior for an Approval Step
   Configure what happens if an approver rejects a request. The final rejection actions for the approval process determine the first step's rejection behavior.

SEE ALSO:

Set Up an Approval Process

Enable Email Approval Response

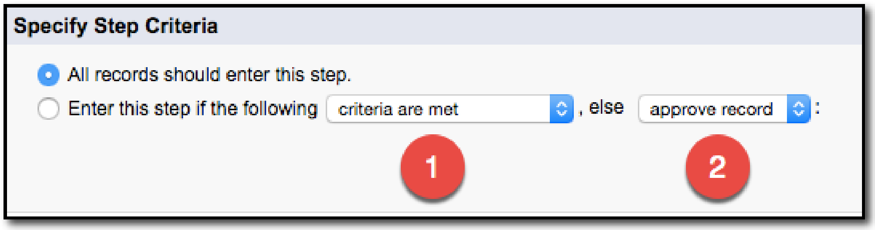
Control Which Records Apply to an Approval Step

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Control which records are part of the approval step by setting the step's criteria. You can also specify what happens to records that don't meet the step's criteria.



Specify Step Criteria

● All records should enter this step.

○ Enter this step if the following criteria are met, else approve record :

Criteria Options

If all records go through this approval step, leave **All records should enter this step** selected.

If only certain types of records are supposed to enter this process, select **Enter this step if the following...** and choose the appropriate option (1). For details on the options, see Control Which Records Apply to an Approval Process.

(2) Else Options for Approval Step Criteria

If you specified filter criteria or entered a formula, choose what happens to records that don't meet the criteria or if the formula doesn't return `True`.

Note: You can't change your selection after the approval process has been activated, even if you deactivate the approval process.

Option	Description
approve record	Approves the request and performs all final approval actions.
reject record	Rejects the request and performs all final rejection actions. This option is available only for the first step in the approval process.
go to next step	Skips this step and goes to the next step. Available only when there's a later step. When you apply this option in the first step, keep in mind: • If the record doesn't meet the criteria for any subsequent steps, the record is rejected. • If you delete all later steps, Salesforce rejects the record. When you apply this option in another step, keep in mind:

Option	Description
	• If you delete all later steps, Salesforce ends the process.

SEE ALSO:

Set Up an Approval Process

Enable Email Approval Response

Identify Assigned Approvers for an Approval Step

Specify who to send an approval request for this step to.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Select Approver	Specify who to assign the approval to. **Let the submitter choose the approver manually. (default)** Prompts the user to select the next approver. **Automatically assign an approver using a standard or custom hierarchy field.** Assigns the approval request to the user in the field displayed next to this option. You select this field when you configure the approval process. **Automatically assign to a queue.** Available only for objects that support queues. Assigns approval requests to a queue. **Automatically assign to approver(s).** Assigns the approval request to one or more specific users, specific queues, or users related to the submitted record. You can add up to 25 per step.
When multiple approvers are selected:	Available only when `Automatically assign to approver(s)` is selected. **Approve or reject based on the first response.** The first response to the approval request determines whether the record is approved or rejected. **Require unanimous approval from all selected approvers.** The record is approved only if everyone approves the request. If any approvers reject the request, the approval request is rejected.
The approver's delegate may also approve this request	Users can identify a delegate in their approval settings. Delegated approvers can't reassign approval requests; they can only approve or reject them. Internal Salesforce users are listed by and can be added using the Delegated Approver lookup field. Use Data Loader and a comma-delineated (CSV) file to add users with communities licenses as Delegated Approvers. The CSV uses the `CommunityUserId` rather than the `UserId` for

	`DelegatedApproverId`. Communities licenses are used with Experience Cloud sites and legacy portals.

SEE ALSO:

Considerations for Setting Approvers

Specify Rejection Behavior for an Approval Step

Configure what happens if an approver rejects a request. The final rejection actions for the approval process determine the first step's rejection behavior.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Option	Description
Perform all rejection actions...	Rejects the request even if previous steps were approved. Salesforce performs all rejection actions specified for this step and all final rejection actions.
Perform ONLY the rejection actions for this step...	Rejects the request, and returns the approval request to the previous approver. Salesforce performs all rejection actions specified for this step.

Add Automated Actions to an Approval Process

You can associate actions to approval steps, initial submission, final approval, final rejection, or recall. Approval processes support four automated actions.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Action Type	Description
Task	Assigns a task to a user who you specify. You can specify the subject, status, priority, and due date of the task.
Email Alert	Sends an email to a designated recipient using a specified email template.
Field Update	Changes the value of a selected field. You can specify a value or create a formula for the new value.
Outbound Message	Sends a message to a designated endpoint. You can also specify a username and the data to include in the message. Not supported for approval processes on junction objects.

Example: When expenses are approved, you want to print checks for payment. To do so, you add an outbound message, which sends the appropriate information to your Oracle accounting service, as a Final Approval action.

Groups of Automated Actions in an Approval Process

Each approval process is organized into groups of actions based on when the actions occur, such as initial submission. To add an automated action to your approval process, determine which group of actions to add it to.

Add an Automated Action to Your Approval Process

If you didn't create an automated action before configuring your approval process, you can create one directly from the approval process.

Add an Existing Automated Action to Your Approval Process

If you've already created an automated action, you can add it to your approval process.

SEE ALSO:

Set Up an Approval Process

Automated Actions

Considerations for Automated Actions

Groups of Automated Actions in an Approval Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Each approval process is organized into groups of actions based on when the actions occur, such as initial submission. To add an automated action to your approval process, determine which group of actions to add it to.

Group	Occurs When...	Default Actions
Initial Submission	A user first submits a record for approval.	Record Lock (locks)
Approval Step Approval	All required approvals for this step have been given for a record.	None
Approval Step Rejection	An approver rejects this request for this step.	None
Final Approval	All required approvals have been given for a record	Record Lock (locks)
Final Rejection	An approver rejects the request, and it goes to the final rejection state.	Record Lock (unlocks)
Recall	A submitted approval request is recalled.	Record Lock (unlocks

SEE ALSO:

Considerations for Automated Actions

Add an Automated Action to Your Approval Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval actions:
- Customize Application

If you didn't create an automated action before configuring your approval process, you can create one directly from the approval process.

1. Open the approval process that you want to add an action to.
2. From the appropriate related list, click **Add New**. For an approval step where the Approval Actions and Rejection Actions are hidden, click **Show Actions**.
3. Choose the type of action.

   The list of available actions differs depending on your settings and whether you've reached the limit for a type of action.

4. Configure the action.

SEE ALSO:

Set Up an Approval Process

Considerations for Automated Actions

Groups of Automated Actions in an Approval Process

Add an Existing Automated Action to Your Approval Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To select approval actions:
- Customize Application

If you've already created an automated action, you can add it to your approval process.

1. Open the approval process that you want to add an action to.
2. From the appropriate related list, click **Add Existing**. If that button is hidden, click **Show Actions**.
3. Choose the type of action.
4. Move the action from Available Actions to Selected Actions.
5. Save your changes.

SEE ALSO:

Groups of Automated Actions in an Approval Process

Considerations for Automated Actions

Activate an Approval Process

After you've created at least one step for the approval process, activate the process.

1. Open the approval process.
2. Make sure that it's configured correctly.
3. Click **Activate**.

SEE ALSO:

Prepare Your Org for Approvals

Considerations for Managing Approvals

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To activate approval processes:

- Customize Application

Prepare Your Org for Approvals

Make sure that your users can submit their records for approval, and consider how you can make it easy for approvers to respond to approval requests.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Let Users Submit for Approval

After you activate an approval process for an object, customize the object's page layouts to support record submission.

Override the Sender for Email Approval Notifications

By default, the sender for email approval notifications is the user who submitted the record for approval. You can override the sender with an organization-wide address, like approval@acmewireless.com.

Let Users Respond to Approval Requests from Your Org

Give your users an instant view of their approval requests by customizing the Home page or navigation bar.

Let Users Respond to Approval Requests by Email

If the email notification includes all the information that an approver must decide, enable email approval response. That way, a user can simply reply to the email notification.

Let Users Respond to Approval Requests from Chatter

If your users don't need in-depth information to decide how to respond to an approval request, enable Approvals in Chatter. That way, they don't have to leave their feed to continue with their day-to-day tasks.

Let Users Respond to Approvals Requests in Slack

If your users don't need in-depth information to decide how to respond to an approval request, and they have a connection to Slack, enable Approvals in Slack. That way, a user can simply respond to the Slack notification.

SEE ALSO:

Set Up an Approval Process

Limits and Considerations for Approvals

Let Users Submit for Approval

After you activate an approval process for an object, customize the object's page layouts to support record submission.

Add the following components to your page layouts.

- Submit for Approval button
- Approval History related list

  The Approval History related list lets users submit approval requests and track a record's progress through an approval process from the record detail page.

SEE ALSO:

Page Layouts

Prepare Your Org for Approvals

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To modify page layouts:

- Customize Application

Override the Sender for Email Approval Notifications

By default, the sender for email approval notifications is the user who submitted the record for approval. You can override the sender with an organization-wide address, like approval@acmewireless.com.

User Permissions Needed

To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

After you add an organization-wide address to your org:

1. From Setup, enter `Process Automation Settings` in the Quick Find box, then select **Process Automation Settings**.
2. For Email Approval Sender, select the organization-wide address.
3. Save your changes.

Let Users Respond to Approval Requests from Your Org

Give your users an instant view of their approval requests by customizing the Home page or navigation bar.

Lightning Experience:

- Add the Items to Approve component to the appropriate Lightning Home pages.

  This component is available only for Home pages. To add it to a Home page, use the Lightning App Builder in Setup.
- Add the Approval Requests navigation item to the appropriate Lightning apps.

  This item is available only for Lightning apps. To add it to a Lightning app, use the App Manager in Setup.

Salesforce mobile app:

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Add the Approvals item to the navigation items of any Lightning app.

Salesforce Classic:

Add the Items to Approve related list to the appropriate home page layouts.

SEE ALSO:

Create Lightning Apps

Set Up the Lightning Experience Home Page

Salesforce Classic Home Tab Page Layouts

Prepare Your Org for Approvals

Let Users Respond to Approval Requests by Email

If the email notification includes all the information that an approver must decide, enable email approval response. That way, a user can simply reply to the email notification.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Email Approval Response

Before you enable the ability to act on approvals via email, review how email works with your approval processes.

Default Template for Email Approval Response

When you enable email approval response, Salesforce uses a default email template for approval processes—unless you specify a custom email template.

Enable Email Approval Response

After you've reviewed the considerations and prepared the right template, flip the switch that lets users respond to approval requests directly from their email.

SEE ALSO:

Prepare Your Org for Approvals

Let Users Respond to Approval Requests from Chatter

Considerations for Email Approval Response

Before you enable the ability to act on approvals via email, review how email works with your approval processes.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Compatibility with Approval Processes

Email approval response isn't supported for approval processes that:

- Assign approval to a queue
- After the first step, let the approver manually select the next approver

Implicit Agreement with Salesforce

By enabling the email approval response feature, you agree to let Salesforce:

- Process email approval responses

- Update approval requests for all active users in your org
- Update the approval object on behalf of your org's users

SEE ALSO:

Limits and Considerations for Approvals

Let Users Respond to Approval Requests by Email

Default Template for Email Approval Response

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

When you enable email approval response, Salesforce uses a default email template for approval processes—unless you specify a custom email template.

Example: *Requesting User* has requested your approval for the following item.

To approve or reject this item, reply to this email with the word APPROVE, APPROVED, YES, REJECT, REJECTED, or NO in the first line of the email message, or click this link:

Link to approval request page

If replying via email you can also add comments on the second line. The comments are stored with the approval request in Salesforce CRM.

Note: For Salesforce to process your response the word APPROVE, APPROVED, YES, REJECT, REJECTED, or NO must be in the first line of the reply email. Also, any comment must be in the second line.

If your org has Approvals in Chatter enabled and the approver opted to receive notifications as Chatter posts, the default email template is appended with:

Example: You can also approve, reject, and comment on this request from your Chatter feed:

Link to approval post in Chatter

Note: If you use a custom email template for your approval process, make sure that it explains both response options: clicking the link and replying by email. If the user doesn't respond correctly (for example, if the user misspells approve or types it on the wrong line), Salesforce doesn't register the response.

SEE ALSO:

Email Templates in Salesforce Classic

Merge Fields for Approvals

Let Users Respond to Approval Requests by Email

Enable Email Approval Response

After you've reviewed the considerations and prepared the right template, flip the switch that lets users respond to approval requests directly from their email.

Before you begin, give the appropriate users the "API Enabled" user permission so that they can respond to approval requests by email.

1. From Setup, enter `Process Automation Settings` in the `Quick Find` box, then select **Process Automation Settings**.
2. Select **Enable email approval response**.
3. Save your changes.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To enable Email Approval Response:

- Customize Application

SEE ALSO:

Considerations for Email Approval Response

Let Users Respond to Approval Requests by Email

Let Users Respond to Approval Requests from Chatter

If your users don't need in-depth information to decide how to respond to an approval request, enable Approvals in Chatter. That way, they don't have to leave their feed to continue with their day-to-day tasks.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Prepare to Enable Approvals in Chatter

Because Approvals in Chatter relies on both Chatter and the Approvals feature, getting your org set up involves more than just turning on the feature. Before you enable Approvals in Chatter, understand the limitations and considerations for Approvals in Chatter and post templates.

Considerations for Approvals in Chatter

Find out more about Approvals in Chatter, before you enable it.

Enable Approvals in Chatter

If your organization has both Approvals and Chatter enabled, administrators can turn on Approvals in Chatter. Users then receive approval requests as posts in their Chatter feeds.

Where Do Approval Request Posts Appear?

When your org has Approvals in Chatter enabled, approval request posts appear in various Chatter feeds. To see the approval request post, you must have access to the approval record.

Chatter Post Templates for Approval Requests

Approval post templates for Chatter let you customize the information that is included in the approval request post when it displays in a Chatter feed.

Prepare to Enable Approvals in Chatter

Because Approvals in Chatter relies on both Chatter and the Approvals feature, getting your org set up involves more than just turning on the feature. Before you enable Approvals in Chatter, understand the limitations and considerations for Approvals in Chatter and post templates.

Do the following for each object for which you want approval requests to appear in Chatter.

1. Enable feed tracking.
2. Create an approval post template.

   Tip: For each object, create one post template that works for all approval processes. Mark that post template the default for the object.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

- Feed Tracking
- Chatter Post Templates for Approval Requests
- Where Do Approval Request Posts Appear?
- Considerations for Approvals in Chatter

Considerations for Approvals in Chatter

Find out more about Approvals in Chatter, before you enable it.

- When you enable Approvals in Chatter in your org, it's turned on for all users. Users can then update their own Chatter settings to opt out of receiving approval requests as posts in their Chatter feeds.
- Chatter post approval notifications are available only for approval processes associated with an object that has been enabled for feed tracking.
- If the approval object is a detail object in a master-detail relationship, `Owner` isn't available for approval page layouts or approval post templates.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Limitations

- Approvals in Chatter doesn't support delegated approvers or queues.
- You can't recall or reassign an approval request from a post. Instead, perform these actions from the approval record.
- Approval requests from Sites or portal users aren't supported.

Approval Posts

- Approval posts can't be deleted in the Salesforce user interface; you can only delete them through the API.
- If you don't select an approval post template, the approval post uses the system default template or the default template for the object, if available.
- Only users with access to the approval record can see the approval request post. Comments on approval posts aren't persisted to the approval record.
- Different users see different configurations of the approval request post.
  - Only approvers see approval action buttons on their posts, and then only in their profile feed or their news feed.
  - Only approvers see approver names in the header.

- If you change the approver, step name, or the routing type on an approval process while it's in progress, existing approval posts aren't updated.
- When an approval request is recalled, a new post is generated. It appears on the news feeds of the submitter, all approvers, and followers of the object. It also appears on the record feed.
- If a step requires unanimous approval from multiple approvers, the approval request post for that step doesn't list all selected approvers in its header. Approvers see only their own name in the post header.

SEE ALSO:

Let Users Respond to Approval Requests by Email

Prepare to Enable Approvals in Chatter

Where Do Approval Request Posts Appear?

Limits and Considerations for Approvals

Enable Approvals in Chatter

If your organization has both Approvals and Chatter enabled, administrators can turn on Approvals in Chatter. Users then receive approval requests as posts in their Chatter feeds.

Before you begin, make sure that all approval processes in your org are properly configured to take advantage of Approvals in Chatter. After turning on this feature, all existing active approval processes start generating Chatter posts.

1. From Setup, enter `Chatter Settings` in the `Quick Find` box, then select **Chatter Settings**.
2. Click **Edit**.
3. Select **Allow Approvals**.
4. Save your changes.

SEE ALSO:

Prepare to Enable Approvals in Chatter

Considerations for Approvals in Chatter

Where Do Approval Request Posts Appear?

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To enable Approvals in Chatter:

- Customize Application

Where Do Approval Request Posts Appear?

When your org has Approvals in Chatter enabled, approval request posts appear in various Chatter feeds. To see the approval request post, you must have access to the approval record.

Approval request posts show up in these feeds.

- Chatter feed of the assigned approver
- Submitter's profile
- Chatter feed of the submitter if the submitter is following the approval request record
- Chatter feed of the approval request record
- Chatter feed of anyone following the approval request record
- Object-specific filter on the Chatter feed of anyone following the approval record

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Company filter of every user with access to the approval record

SEE ALSO:

What Happens When You Opt Out of Chatter Approval Requests?

Considerations for Approvals in Chatter

Let Users Respond to Approval Requests from Chatter

Chatter Post Templates for Approval Requests

Approval post templates for Chatter let you customize the information that is included in the approval request post when it displays in a Chatter feed.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Chatter Post Templates for Approval Requests

Keep these limitations and dependencies in mind when working with post templates.

Create a Chatter Post Template

Identify which fields to display in an approval request post.

SEE ALSO:

Manage Deleted Custom Fields

Considerations for Chatter Post Templates for Approval Requests

Keep these limitations and dependencies in mind when working with post templates.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Limitations

- The associated object must be enabled for approvals and feed tracking.
- If an approval post template is in use by an approval process, you can't delete it.
- Chatter posts for approval requests only appear in Salesforce Classic. To respond to approval requests in Lightning Experience, users go to the Approval Requests tab.

Dependencies

- Deleting a custom field removes it from any approval post template that references it. Existing posts aren't affected. Undeleting the custom field restores it to the available fields list, but doesn't restore it to any approval post templates that previously contained it.
- Deleting (or undeleting) a custom object also deletes (or undeletes) its associated approval post templates and any of its approval request posts that are already in Chatter feeds.
- If you rename a custom object, approval post templates associated with it update accordingly.

SEE ALSO:

Create a Chatter Post Template

Limits and Considerations for Approvals

Create a Chatter Post Template

Identify which fields to display in an approval request post.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create approval request post templates:

- Customize Application

1. From Setup, enter `Post Templates` in the `Quick Find` box, then select **Post Templates**.
2. Click **New Template**.
3. Select the object for your template.
4. Click **Next**.
5. Give the template a name and description.
6. If you want this template to be the default for the associated object, select **Default**.
7. Add up to four fields to display on the approval request post.

   We recommend putting any text-heavy fields—such as Comments or Description—at the bottom.
8. Save your changes.

SEE ALSO:

Choose Approval Request Notification Templates

Considerations for Chatter Post Templates for Approval Requests

Let Users Respond to Approvals Requests in Slack

If your users don't need in-depth information to decide how to respond to an approval request, and they have a connection to Slack, enable Approvals in Slack. That way, a user can simply respond to the Slack notification.

Considerations for Approvals in Slack

Find out more about Approvals in Slack, before you enable it.

Enable Approval Notifications in Slack

If your org uses both Approvals and Salesforce Digital HQ app, approval notifications are automatically enabled in Slack. Users receive approval requests as messages on the Salesforce Digital HQ's Messages tab.

Where Do Slack Approval Notifications Appear?

When you have Approvals in Slack enabled, approval notifications are sent to the approver via the Salesforce Digital HQ app as a direct message in Slack. To see the approval request post, you must have access to Slack.

Considerations for Approvals in Slack

Find out more about Approvals in Slack, before you enable it.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Users must have the Salesforce Digital HQ app in Slack. When you enable Approvals in Slack in your org, it's turned on for all users. Before you use Approvals in Slack, make sure you understand the limitations.

- You can connect the Salesforce Digital HQ app to only one Salesforce org.
- The only available actions are Approve and Reject.
- The Show More link doesn't work for Salesforce Classic users.

- If the approver has to manually select the next approver, they must log in to the full Salesforce site to complete the approval request.
- Users can respond only to approval requests without comments.
- Up to four fields only of an approval request can appear in a Slack notification.

Enable Approval Notifications in Slack

If your org uses both Approvals and Salesforce Digital HQ app, approval notifications are automatically enabled in Slack. Users receive approval requests as messages on the Salesforce Digital HQ's Messages tab.

Note: Slack notifications are turned on automatically. Admins can turn off Slack notifications from Setup on the Notification Delivery Settings page.

1. From Setup, in the Quick Find box, enter `Notification Delivery Settings`, and select **Notification Delivery Settings**.
2. From the Approval requests dropdown menu, select **Edit**.
3. Select **Slack**, and enable **Salesforce Digital HQ**.

SEE ALSO:

Salesforce for Slack

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

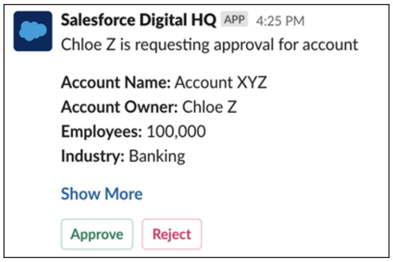
To enable approvals in Slack:

- Customize Application

Where Do Slack Approval Notifications Appear?

When you have Approvals in Slack enabled, approval notifications are sent to the approver via the Salesforce Digital HQ app as a direct message in Slack. To see the approval request post, you must have access to Slack.

- Users review the request, and select **Approve** or **Reject**, or select **Show More** to be directed to the Salesforce app to view details.



Salesforce Digital HQ APP 4:25 PM
Chloe Z is requesting approval for account

Account Name: Account XYZ
Account Owner: Chloe Z
Employees: 100,000
Industry: Banking

- Users can continue to receive email, Lightning Experience, and mobile notifications about approval requests.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Limits and Considerations for Approvals

Before you automate something with an approval process, be aware of the limits and considerations.

Users can't see which approval process is triggered when they click **Submit for Approval**. Familiarize users on the criteria for each approval process and what each approval process does. If the record doesn't meet the entry criteria or if they're not an allowed submitter for any approval processes, Salesforce displays an error.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approval Limits

Salesforce limits the number of approval processes in your org, as well as the number of steps and actions in each approval process.

Considerations for Configuring Approvals

When creating or editing an approval process, keep in mind how approvals are compatible with other features. Before you start, draw out the steps of your approval process.

Merge Fields for Approvals

Approval merge fields include *{!ApprovalRequest.fieldName}* and *{!Approval_Requesting_User.fieldName}*. They're supported in certain email templates and return different values based on the status of the approval process instance.

Considerations for Setting Approvers

When you specify approvers for a given approval step—or for the only step if you're using the jump start wizard—keep these considerations in mind.

Considerations for Managing Approvals

Keep these things in mind when maintaining existing approval processes—including activating and deleting them.

Considerations for the Salesforce Mobile App

Learn about the approvals functionality in Lightning Experience on desktop that isn't available or that works differently in the Salesforce mobile app.

SEE ALSO:

Considerations for Email Approval Response

Considerations for Approvals in Chatter

Approvals: What's Different or Not Available in the Salesforce Mobile App

Considerations for Approval History Reports

Restrictions for Approval Processes in Change Sets

Approval Limits

Salesforce limits the number of approval processes in your org, as well as the number of steps and actions in each approval process.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Per-Org Limit	Value
Active approval processes	1,000
Total approval processes	2,000
Active approval processes per object	300

Per-Org Limit	Value
Total approval processes per object	500
Steps per approval process	30
Approvers per step	25
Initial submission actions per approval process[2]	40
Final approval actions per approval process[2]	40
Final rejection actions per approval process[2]	40
Recall actions per approval process[2]	40
Maximum characters in approval request comments	4,000 In Chinese, Japanese, or Korean, the limit is 1,333 characters.

Considerations for Configuring Approvals

When creating or editing an approval process, keep in mind how approvals are compatible with other features. Before you start, draw out the steps of your approval process.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Associated Object

If the approval object is a detail object in a master-detail relationship, `Owner` isn't available for approval page layouts or approval post templates.

Approval Criteria

In approval criteria—either the entry criteria or step criteria—don't reference expressions that resolve to random values. That way, if the criteria must be evaluated again, the record is evaluated the same every time.

Compatibility with Other Features

- Flows can delete records that are pending approval.
- Design automated actions so that you can use them for both workflow rules and approval processes.

Field Update Actions in Approvals

- An approval process can specify a field update action that reevaluates workflow rules for the updated object. If, however, the reevaluated workflow rules include a cross-object field update, those cross-object field updates are ignored.
- Field updates that are executed as approval actions don't trigger workflow rules or entitlement processes.

Anticipate Errors

Consider reviewing the content on approvals errors. That way, you can anticipate common issues and configure your approval process so that the error is less likely.

Approvals in Unlocked Packages

- Unlocked packages can include Approvals with steps that reference related users or queues as approvers; users aren't supported.
- Queues and related user fields (lookup fields) referenced by the approval steps must be included in the unlocked package.
- An Approval Process can only be included in unlocked packages that don't have a specified namespace.

SEE ALSO:

What Does This Approvals Error Mean?

Set Up an Approval Process

Considerations for Automated Actions

Considerations for Chatter Post Templates for Approval Requests

Merge Fields for Approvals

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approval merge fields include *`{!ApprovalRequest.fieldName}`* and *`{!Approval_Requesting_User.fieldName}`*. They're supported in certain email templates and return different values based on the status of the approval process instance.

Tip: The submitter isn't always the current user. For custom email templates, use *`{!Approval_Requesting_User.fieldName}`* instead of *`{!User.fieldName}`*.

Where Are Approval Merge Fields Supported?

You can use approval process merge fields in email templates, but not mail merge templates. Except for `{!ApprovalRequest.Comments}`, approval merge fields named `{!ApprovalRequest.field_name}` in email templates return values only in approval assignment emails and email alerts for approval processes. When used in other emails—including email alerts for workflow rules—the approval merge field returns `null`.

What Values Does a Merge Field Provide?

The generated value of an ApprovalRequest merge field depends on which step the approval process is in.

- In the approval request email, a merge field returns the submitter's name and the name of the first step.
- When the request is approved, the merge field returns the most recent approver's name and the name of the second step, if applicable.
- For subsequent actions, a merge field value returns the previous completed step.
- For an approval step that requires unanimous approval from multiple approvers, *`{!ApprovalRequest.Comments}`* returns only the most recently entered comment in emails.

SEE ALSO:

Default Template for Email Approval Response

Email Templates in Salesforce Classic

Considerations for Setting Approvers

When you specify approvers for a given approval step—or for the only step if you're using the jump start wizard—keep these considerations in mind.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Users with these permissions can respond to approval requests, even if they aren't designated approvers.
  - Modify All Data
  - Modify All for an object
- Make sure that the assigned approver has access to read the records for the approval requests. For example, a user who can't view expense records can't view expense approval requests.
- Approval processes that let users select an approver manually also let users select themselves as the approver.
- You can assign an approval request to the same user multiple times in a single step. However, Salesforce sends the user only one request.
- In Lightning Experience, when an approval request has more than one assigned approvers, a `ProcessInstanceStep` is created for each assigned approver. When the approval request has the Approval based on first response setting enabled, the values displayed in `Assigned To` and `Actual Approver` are affected.
  - Assigned to is set to an approver assigned to the record
  - Actual Approver is set to the approver who approved the request
- Here's what happens to the list of approvers after a record enters an approval step and the approval process later returns to that step.
  - If the user who responded isn't in the designated approvers list and has either Modify All Data or Modify All permissions for the object, that user replaces the original approver in the list of approvers.
  - If a user who responded is in the designated approvers list, the list of approvers for that step doesn't change. This behavior occurs even if the field values that designate the approvers have changed.

  For example, an approval process's first step requests approval from a user's manager. If the approval request is rejected in the second step, the approval request returns to the first step. This table explores what happens to the list of approvers.

If...	The Designated Approver Is...
The user's manager originally responded to the approval request.	The manager
The user's manager originally responded to the approval request. Since then, the user's manager has changed.	The original manager The new manager isn't a designated approver for this step.
A user with Modify All Data permissions originally responded to the approval request.	The user with Modify All Data permissions That user replaces the user's manager in the list of designated approvers for this step.

- A manager's manager is not an option for a designated approver.

Assigning Approval Steps to Queues

You can assign approval requests to a queue only if the associated object supports queues. Email approval response isn't supported for approval processes that assign approval to a queue.

When the assigned approver is a queue:

- Any queue member can approve or reject an approval request that is assigned to the queue.
- Approval request emails are sent to the queue email address. If the queue is set up to send email to members, approval request emails get sent to the queue members, unless their approval user preferences are set to never receive approval request emails.
- Because email notifications to a queue aren't intended for an external audience, `{!ApprovalRequest.External_URL}` returns the equivalent internal URL.
- Salesforce mobile app notifications for approval requests aren't sent to queues. For each approval step involving a queue, we recommend adding individual users as assigned approvers, so at least those individuals can receive the approval request notifications in the Salesforce mobile app. To have both queues and individual users as assigned approvers, select `Automatically assign to approver(s)` instead of `Automatically assign to queue` in the approval step.
- When an approval request is rejected and returned to the previous approver and the previous approver was a queue, the approval request is assigned to the user who approved it instead of the queue.
- The Approval History related list displays the queue name in the `Assigned To` column and the actual user who approved or rejected the approval request in the `Actual Approver` column.

SEE ALSO:

Identify Assigned Approvers for an Approval Step

Limits and Considerations for Approvals

Considerations for Managing Approvals

Keep these things in mind when maintaining existing approval processes—including activating and deleting them.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Admin Permissions

Users with one of these permissions are considered approval admins.

- Modify All object-level permission for the given object
- Modify All Data user permission

Approval admins can:

- Approve or reject pending approval requests without being part of the approval process
- Edit records that have been locked for approval

Activating Approval Processes

- An approval process must have at least one step before you can activate it.
- Before you activate your approval process, test it in your Salesforce sandbox.
- After an approval process is activated, you can't add, delete, or change the order of the steps or change its reject or skip behavior, even if the process is inactive.

Monitoring In-Flight Approval Processes

Standard reports for approval requests are included in both the Administrative Reports folder and the Activity Reports folder.

Deploying over Existing Approval Processes

When you deploy an approval process with no entry criteria to overwrite an existing approval process with entry criteria, then the entry criteria from the existing process are applied to the deployed process.

Deleting Approval Processes

Before you delete an approval process:

- Make sure it's inactive.
- Delete all approval requests that are associated with it, and remove them from the Recycle Bin.
- Delete all records, for example, accounts that were submitted through the approval process regardless of status. By deleting the records, the associated `ProcessInstanceWorkitem` and `ProcessInstance` records are also deleted automatically.
- If you can't delete the approval process, try again after 2 days. Salesforce can take up to 2 days to delete the files that you removed from the recycle bin.

SEE ALSO:

Activate an Approval Process

Manage Multiple Approval Requests

Limits and Considerations for Approvals

Considerations for the Salesforce Mobile App

Learn about the approvals functionality in Lightning Experience on desktop that isn't available or that works differently in the Salesforce mobile app.

EDITIONS

Available in: Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approval Responses

You can't unlock a record that's locked for approval.

Salesforce Mobile App Notifications for Approval Requests

- Notifications for approval requests aren't sent to queues or delegates. For each approval step involving a queue, add individual users as assigned approvers, so those individuals can receive the approval request notifications in the mobile app. To have both queues and individual users as assigned approvers, select **Automatically assign to approver(s)** instead of **Automatically assign to queue** in the approval step.
- Notifications for approval requests are sent only to users who have access to the record being approved. Assigned approvers who don't have record access can receive email approval notifications, but they can't complete the approval request until someone grants record access.

Approvals in Chatter

In the Salesforce mobile app, you can't respond to approval requests from Chatter. To respond to approval requests, go to the Approvals navigation item.

Approval Comments

- The Salesforce mobile app prompts you for comments after you tap Approve or Reject.
- The Approval History related list displays truncated comments. To see the full comment for a given approval instance, tap the instance, then tap **Comments**.

Approval History Related List

- The Approval History related list doesn't include the Submit for Approval button.
- When working with approvals in Experience Cloud sites, role-based external users can see and take action from the Approval History related list, but they can't submit requests for approval.

Sample Approval Processes

Review samples of common approval processes to help you get started creating your own.

Sample Approval Process: PTO Requests

Most companies require employees to file a PTO (Paid Time Off) request and have their manager approve it. In three phases, here's how to automate a simple one-step PTO request process using Salesforce.

Sample Approval Process: Expense Reports

If your company requires that employees file expense reports for managers to approve, you can automate this process in Salesforce.

Sample Approval Process: Discounting Opportunities

Opportunities that are discounted more than 40% require a CEO approval. Use this example to create a one-step approval process.

Sample Approval Process: Job Candidates

When your company interviews candidates for a position, you can have several levels of approval before you can send an offer letter. Use this example to create a three-step approval process that requires approval from multiple management levels.

Sample Approval Process: PTO Requests

Most companies require employees to file a PTO (Paid Time Off) request and have their manager approve it. In three phases, here's how to automate a simple one-step PTO request process using Salesforce.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Prep Your Organization

Before creating the approval process:

- If you don't yet have a custom object to track your PTO requests, create a custom object and tab called PTO Requests. Add the appropriate fields for your PTO Requests such as `Start Date`, `End Date`, and `Employee Name`.
- To notify approvers about a pending approval request, create an email template. To direct users to the approval page in Salesforce, include approval process merge fields.

Create the Approval Process

Use the jump start wizard to create an approval process for the PTO Request custom object and specify the following:

Tip: To let the submitter withdraw a submitted PTO request, click **Edit** and choose **Initial Submitters**. Then select `Allow submitters to recall approval requests`.

- Select the email template you created for this approval process.
- Don't specify filter criteria. That way, PTO requests are included in this approval process regardless of their attributes.
- Select the `Automatically assign an approver using a standard or custom hierarchy field` option, then choose `Manager`.
- The jump start wizard automatically chooses the record owner as the only person who can submit PTO requests.

Wrap Things Up

- After you created the approval process, add the Approval History related list to the PTO Request object page layout.
- Consider adding the Items To Approved related list to your custom home page layouts. The related list shows users all approval requests that are waiting for their response.
- If you have a sandbox, test the approval process, then activate it.

SEE ALSO:

Create a Custom Object

Email Templates in Salesforce Classic

Create an Approval Process with the Jump Start Wizard

Prepare Your Org for Approvals

Sample Approval Process: Expense Reports

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

If your company requires that employees file expense reports for managers to approve, you can automate this process in Salesforce.

Use this example to create a two-step expense report approval process for all employees in your headquarters office. It specifies that expenses less than $50 are automatically approved, expenses $50 and over require manager approval, and expenses over $5,000 require additional approval from two VPs. This example highlights a parallel approval process and the “else” option.

Prep Your Organization:

Before creating the approval process:

- If you don’t yet have a custom object to track your expenses, create a custom object and tab called Expense Reports. Add the appropriate fields such as `Amount`, `Description`, `Status`, `Start Date`, and `End Date`.
- Create a custom field on the user object `Office Location`. Assign the “HQ” value to users in the headquarters office location.

Create the Approval Process:

Create an approval process using the Expense Report custom object and specify the following:

- The filter criteria for this approval process is *`Current User: Office Location equals HQ`*. Records must meet this criteria before they can be submitted to this approval process.
- Choose the `Manager` field as the next automated approver.
- To notify approvers that their approval is requested, create an email template. To direct users to the approval page in Salesforce, include approval process merge fields.
- Choose the record owner or any other user who you want to be able to submit expense reports.

- Create these approval steps.
  1. Create a step named *`Step 1: Manager Approval`* with these specifications:
     - Name this step *`Step 1: Manager Approval`*.
     - Select `Enter this step if the following` and choose **criteria are met**. Also, choose **approve record** for the `else` option.
     - Set the filter criteria to: *`Expense: Amount greater or equal 50`*.
     - In the `Automatically assign to approver(s)` option, select the manager of the user submitting the request.
     - If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
  2. Create an approval step named *`Step 2: Multiple VP Approval`* and specify these attributes.
     - Use the filter criteria *`Expense Amount greater or equal 5000`*.
     - Choose `Automatically assign to approver(s)` and select two users with a VP role.
     - Select the `Require UNANIMOUS approval from all selected approvers` option. The request isn't approved unless both designated users approve.
     - If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
     - Choose `Perform ONLY the rejection actions for this step...` so that the request returns to the manager for changes if one of the VPs rejects the request.

Tip: Consider creating these final approval actions:

- Define a field update to automatically change the `Status` field to "Approved."
- Send an approval notification to the user who submitted the expense report.
- To print a reimbursement check, send an outbound message to your back-office financial system.

Wrap Things Up:

- After you created the approval process, add the Approval History related list to the Expense Report object page layout.
- Consider adding the Items To Approved related list to your custom home page layouts. The related list shows users all approval requests that are waiting for their response.
- If you have a sandbox, test the approval process, then activate it.

SEE ALSO:

Create a Custom Object

Email Templates in Salesforce Classic

Create Custom Fields

Set Up an Approval Process

Prepare Your Org for Approvals

Sample Approval Process: Discounting Opportunities

Opportunities that are discounted more than 40% require a CEO approval. Use this example to create a one-step approval process.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Prep Your Organization

Before creating the approval process:

- To notify approvers about a pending approval request, create an email template. To direct users to the approval page in Salesforce, include approval process merge fields.
- Create the following custom fields for opportunities:
  - A percent field called `Discount Percent` so that users can enter a percentage discount.
  - A checkbox field called `Discount Approved` to indicate whether the CEO approved the discount.

Create the Approval Process

Create an approval process on the Opportunity object and specify the following:

- The filter criteria for this approval process is *`Discount Percent greater or equal 0.4`*. Records must meet this criteria before they can be submitted to this approval process.
- You don't need to choose a custom field as the next automated approver because you specify later that the CEO must approve all requests.
- Select the email template you created for this approval process.
- Choose the record owner as the only user who can submit a discount request for approval.
- Create one approval step with no filter criteria since all records submitted must be approved or rejected.
- Choose `Automatically assign to approver(s)` and select the name of your CEO.
- If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
- Consider creating the following final approval actions:
  - Email alert to notify the user who submitted the discount request.
  - Field update to automatically select the opportunity `Discount Approved` checkbox.

Wrap Things Up

- After you created the approval process, add the Approval History related list to the appropriate opportunity page layouts.
- Consider adding the Items To Approved related list to your custom home page layouts. The related list shows users all approval requests that are waiting for their response.

- If you have a sandbox, test the approval process, then activate it.

SEE ALSO:

Email Templates in Salesforce Classic

Create Custom Fields

Set Up an Approval Process

Prepare Your Org for Approvals

Sample Approval Process: Job Candidates

When your company interviews candidates for a position, you can have several levels of approval before you can send an offer letter. Use this example to create a three-step approval process that requires approval from multiple management levels.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Prep Your Organization:

Before creating the approval process:

- If you don't yet have a custom object to track candidates, create a custom object and tab called Candidates. Add the appropriate fields such as `Salary`, `Offer Extended` (checkbox), and `Date of Hire`.
- To notify approvers about a pending approval request, create an email template. To direct users to the approval page in Salesforce, include approval process merge fields.

Create the Approval Process:

Create an approval process on the Candidate custom object using the following specifications:

- Don't enter filter criteria because you want all submitted offers to be approved.
- Choose the `Manager` field as the next automated approver.
- Select the email template you created for this approval process.
- Choose the record owner or any other user that you want to be able to submit offer letters.
- Create these approval steps.

  1. Create a step named *`Step 1: Manager Approval`*:
     - No filter is necessary as you want all records to advance to this step.
     - In the `Automatically assign to approver(s)` option, select the manager of the user submitting the request.
     - If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
  2. Create a step named *`Step 2: VP Approval`*:
     - No filter is necessary as you want all records to advance to this step.
     - To allow the manager to select the appropriate VP to approve the request, choose `Let the user choose the approver`.
     - If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
     - Choose `Perform ONLY the rejection actions for this step...` so that the request returns to the manager for changes if the VP rejects the request.
  3. Create a step named *`Step 3: CFO Approval`*:

  - No filter is necessary as you want all records to advance to this step.
  - Choose `Automatically assign to approver(s)` and select the name of your CFO.
  - If appropriate, choose `The approver's delegate may also approve this request` if you want to allow the user in the `Delegated Approver` field to approve requests.
  - Choose `Perform all rejection actions for this step AND all final rejection actions. (Final Rejection)` so that offer letters rejected by your CFO are completely rejected.

Tip:

- Consider creating the following final approval actions:
  - Email alert to notify the user who submitted the offer letter request.
  - Field update to select the `Offer Extended` checkbox.
- Consider creating this final rejection action:
  - Email alert to notify the manager that the offer can’t be extended.

Wrap Things Up:

- After you created the approval process, add the Approval History related list to the Candidates object page layout.
- Consider adding the Items To Approved related list to your custom home page layouts. The related list shows users all approval requests that are waiting for their response.
- If you have a sandbox, test the approval process, then activate it.

SEE ALSO:

Create a Custom Object

Email Templates in Salesforce Classic

Set Up an Approval Process

Prepare Your Org for Approvals

Approval History Reports

If you create a custom report type for approval process instances, users can view the historical details of completed and in-progress approval processes and their individual steps.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Fields Available for Approval History Reports

If you create a custom report type with Process Instance as the primary object and Process Instance Node as the related object, you can create approval history reports with various combinations of fields that enable you to view a detailed history of executed and in-progress approval processes and their individual steps.

Examples of Approval History Reports

See sample reports to learn how you can obtain approval history data.

Considerations for Approval History Reports

Understand the limitations and special behaviors when you create or view approval history reports, which provide a detailed history of approval processes and steps.

SEE ALSO:

Create a Custom Report Type for Approval History

Fields Available for Approval History Reports

If you create a custom report type with Process Instance as the primary object and Process Instance Node as the related object, you can create approval history reports with various combinations of fields that enable you to view a detailed history of executed and in-progress approval processes and their individual steps.

EDITIONS

Available in: Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Group** (View Only), **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Available in: Enhanced Folder Sharing and Legacy Folder Sharing

Process Instance

A process instance represents one instance of an approval process. A new process instance is created each time a record is submitted for approval.

Field	Description
`Approval Process: Name`	Name of the approval process.
`Approval Process Instance ID`	ID of the approval process instance.
`Completed Date`	Date and time when the approval process instance was completed or recalled. If no step criteria are met and the record is auto-approved or auto-rejected, `Completed Date` and `Submitted Date` have the same values.
`Elapsed Days`	Length of time between when the record was submitted for approval and when the approval process was completed or recalled.
`Elapsed Hours`	
`Elapsed Minutes`	
`Last Actor: Full Name`	Full name of the user who most recently participated in the approval process instance. If no step criteria are met and the record is auto-approved or auto-rejected, `Last Actor: Full Name` and `Submitter: Full Name` have the same values.
`Object Type`	Object type of the record that was submitted for approval.
`Pending Step Name`	Name of the approval step at which the record is awaiting approval or rejection.
`Record Name`	Name of the record that was submitted for approval.

Field	Description
`Status`	Status of the approval process instance.
`Submitted Date`	Date and time when the record was submitted for approval.
`Submitter: Full Name`	Full name of the user who submitted the record for approval.

Process Instance Node

A process instance node represents an instance of an approval step. A new process instance node is created each time a record enters a step in an approval process. No process instance node is created when the record doesn't meet the step criteria or if the approval process instance is otherwise completed without entering the step.

Field	Description
`Step: Name`	Name of the approval step.
`Step: Completed Date`	Date and time when the approval step instance was completed or recalled.
`Step Elapsed Days`	Length of time between when the record entered the approval step and when the approval step instance was completed or recalled.
`Step Elapsed Hours`	
`Step Elapsed Minutes`	
`Step Last Actor: Full Name`	Full name of the user who most recently participated in the approval step instance.
`Step Start Date`	Date and time when the record entered the approval step.
`Step Status`	Status of the approval step instance.

SEE ALSO:

- Approval History Reports
- Considerations for Approval History Reports
- Examples of Approval History Reports

Examples of Approval History Reports

See sample reports to learn how you can obtain approval history data.

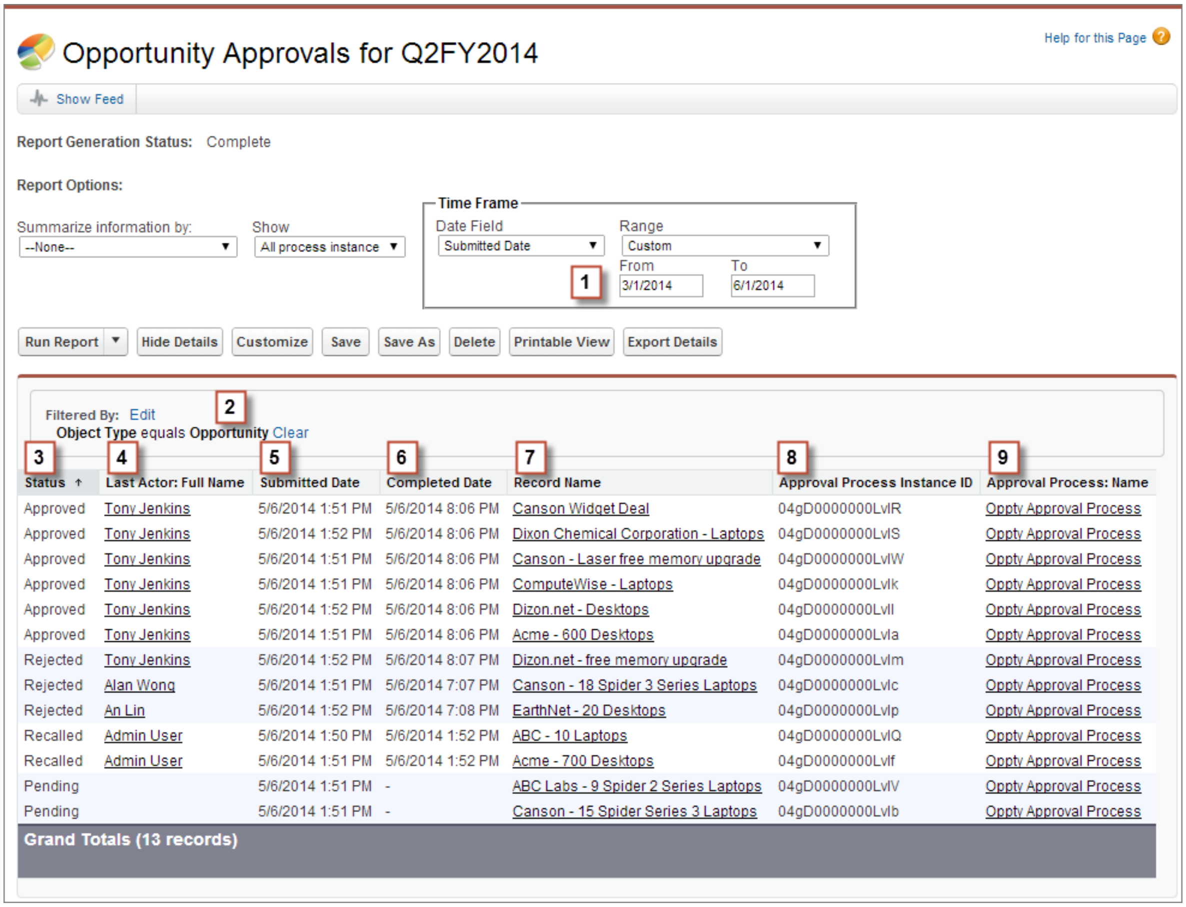
Report Example: Opportunity Approvals Submitted Within a Date Range

This sample report displays approval process instances that were submitted within a specified date range (1) for the Opportunity object (2). The results are sorted by status (3) and include the last actor (4), submitted date (5), completed date (6), record name (7), approval process instance ID (8), and approval process name (9).

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Opportunity Approvals for Q2FY2014

Report Generation Status: Complete

Report Options:

Summarize information by: --None--

Show: All process instance

Time Frame

Date Field: Submitted Date

Range: Custom

From: 3/1/2014 To: 6/1/2014

Filtered By: Edit

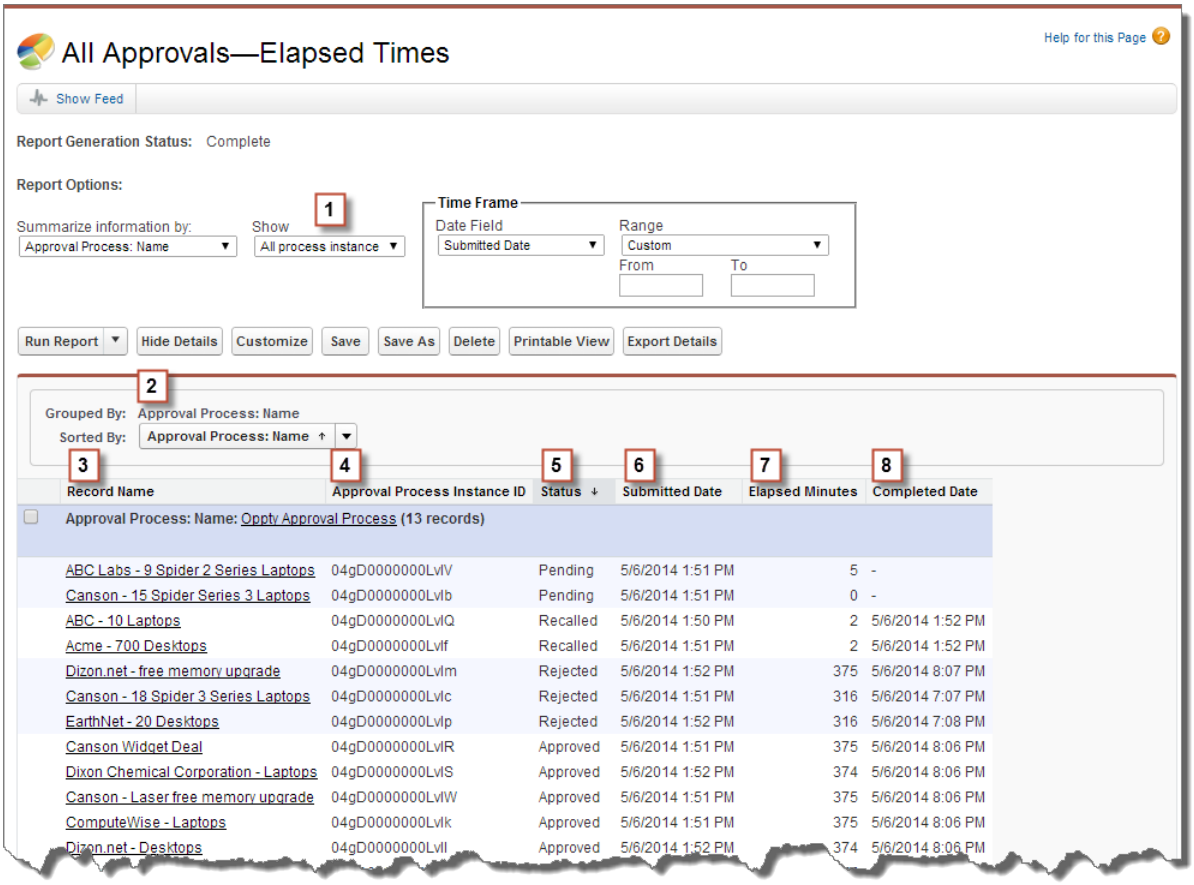
Object Type equals Opportunity Clear

Status ↑	Last Actor: Full Name	Submitted Date	Completed Date	Record Name	Approval Process Instance ID	Approval Process: Name
Approved	Tony Jenkins	5/6/2014 1:51 PM	5/6/2014 8:06 PM	Canson Widget Deal	04gD0000000LvlR	Oppty Approval Process
Approved	Tony Jenkins	5/6/2014 1:52 PM	5/6/2014 8:06 PM	Dixon Chemical Corporation - Laptops	04gD0000000LvlS	Oppty Approval Process
Approved	Tony Jenkins	5/6/2014 1:51 PM	5/6/2014 8:06 PM	Canson - Laser free memory upgrade	04gD0000000LvlW	Oppty Approval Process
Approved	Tony Jenkins	5/6/2014 1:51 PM	5/6/2014 8:06 PM	ComputeWise - Laptops	04gD0000000Lvlk	Oppty Approval Process
Approved	Tony Jenkins	5/6/2014 1:52 PM	5/6/2014 8:06 PM	Dizon.net - Desktops	04gD0000000Lvll	Oppty Approval Process
Approved	Tony Jenkins	5/6/2014 1:51 PM	5/6/2014 8:06 PM	Acme - 600 Desktops	04gD0000000Lvla	Oppty Approval Process
Rejected	Tony Jenkins	5/6/2014 1:52 PM	5/6/2014 8:07 PM	Dizon.net - free memory upgrade	04gD0000000Lvlm	Oppty Approval Process
Rejected	Alan Wong	5/6/2014 1:51 PM	5/6/2014 7:07 PM	Canson - 18 Spider 3 Series Laptops	04gD0000000Lvlc	Oppty Approval Process
Rejected	An Lin	5/6/2014 1:52 PM	5/6/2014 7:08 PM	EarthNet - 20 Desktops	04gD0000000Lvlp	Oppty Approval Process
Recalled	Admin User	5/6/2014 1:50 PM	5/6/2014 1:52 PM	ABC - 10 Laptops	04gD0000000LvlQ	Oppty Approval Process
Recalled	Admin User	5/6/2014 1:51 PM	5/6/2014 1:52 PM	Acme - 700 Desktops	04gD0000000Lvlf	Oppty Approval Process
Pending		5/6/2014 1:51 PM	-	ABC Labs - 9 Spider 2 Series Laptops	04gD0000000LvlV	Oppty Approval Process
Pending		5/6/2014 1:51 PM	-	Canson - 15 Spider Series 3 Laptops	04gD0000000Lvlb	Oppty Approval Process

Grand Totals (13 records)

Report Example: Approvals—Elapsed Times

This sample report displays all approval process instances (1) and groups results by the approval process name (2). The results include the record name (3), approval process instance ID (4), status (5), submitted date (6), elapsed minutes (7), and completed date (8).



All Approvals—Elapsed Times

Report Generation Status: Complete

Report Options:

Summarize information by: Approval Process: Name

Show: All process instance

Time Frame

Date Field: Submitted Date

Range: Custom

From

To

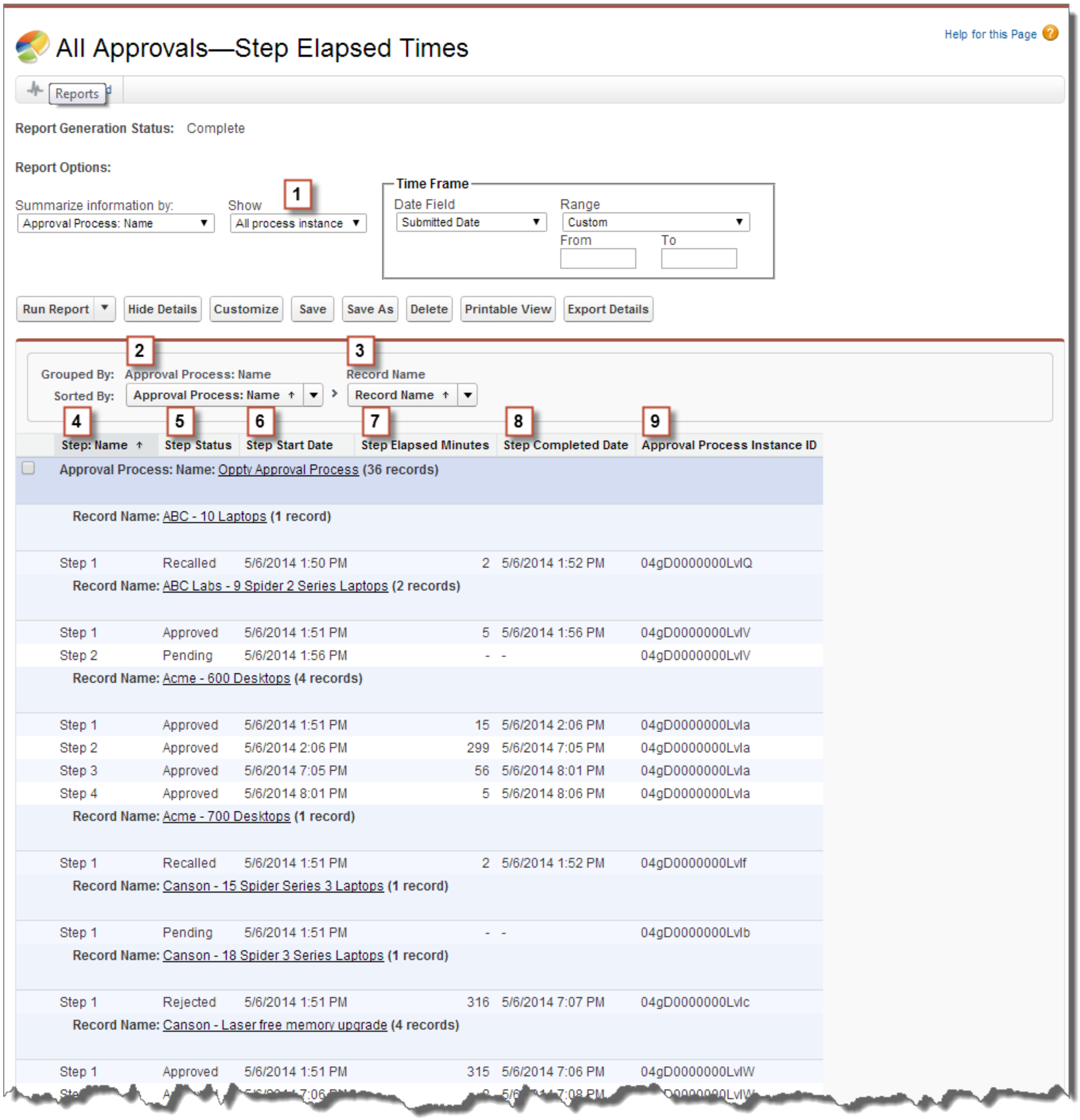
Grouped By: Approval Process: Name

Sorted By: Approval Process: Name ↑

Record Name	Approval Process Instance ID	Status ↓	Submitted Date	Elapsed Minutes	Completed Date
□ Approval Process: Name: Oppty Approval Process (13 records)					
ABC Labs - 9 Spider 2 Series Laptops	04gD0000000LvlV	Pending	5/6/2014 1:51 PM	5	-
Canson - 15 Spider Series 3 Laptops	04gD0000000Lvlb	Pending	5/6/2014 1:51 PM	0	-
ABC - 10 Laptops	04gD0000000LvlQ	Recalled	5/6/2014 1:50 PM	2	5/6/2014 1:52 PM
Acme - 700 Desktops	04gD0000000Lvlf	Recalled	5/6/2014 1:51 PM	2	5/6/2014 1:52 PM
Dizon.net - free memory upgrade	04gD0000000Lvlm	Rejected	5/6/2014 1:52 PM	375	5/6/2014 8:07 PM
Canson - 18 Spider 3 Series Laptops	04gD0000000Lvlc	Rejected	5/6/2014 1:51 PM	316	5/6/2014 7:07 PM
EarthNet - 20 Desktops	04gD0000000Lvlp	Rejected	5/6/2014 1:52 PM	316	5/6/2014 7:08 PM
Canson Widget Deal	04gD0000000LvlR	Approved	5/6/2014 1:51 PM	375	5/6/2014 8:06 PM
Dixon Chemical Corporation - Laptops	04gD0000000LvlS	Approved	5/6/2014 1:52 PM	374	5/6/2014 8:06 PM
Canson - Laser free memory upgrade	04gD0000000LvlW	Approved	5/6/2014 1:51 PM	375	5/6/2014 8:06 PM
ComputeWise - Laptops	04gD0000000Lvlk	Approved	5/6/2014 1:51 PM	375	5/6/2014 8:06 PM
Dizon.net - Desktops	04gD0000000Lvll	Approved	5/6/2014 1:52 PM	374	5/6/2014 8:06 PM

Report Example: Approval Steps—Elapsed Times

This sample report displays all approval process instances (1) and groups results by approval process name (2) and record name (3). The results are sorted by step name (4) and include step status (5), step start date (6), step elapsed minutes (7), step completed date (8), and approval process instance ID (9).



All Approvals—Step Elapsed Times

Report Generation Status: Complete

Report Options:

Summarize information by: Approval Process: Name

Show: All process instance

Time Frame
Date Field: Submitted Date
Range: Custom
From
To

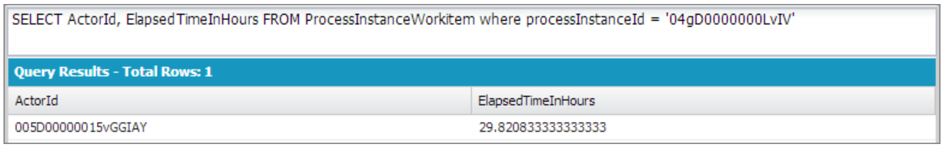
Grouped By: Approval Process: Name > Record Name

Sorted By: Approval Process: Name ↑ > Record Name ↑

Step: Name ↑	Step Status	Step Start Date	Step Elapsed Minutes	Step Completed Date	Approval Process Instance ID
□ Approval Process: Name: Oppty Approval Process (36 records)					
Record Name: ABC - 10 Laptops (1 record)					
Step 1	Recalled	5/6/2014 1:50 PM	2	5/6/2014 1:52 PM	04gD0000000LvIQ
Record Name: ABC Labs - 9 Spider 2 Series Laptops (2 records)					
Step 1	Approved	5/6/2014 1:51 PM	5	5/6/2014 1:56 PM	04gD0000000LvIV
Step 2	Pending	5/6/2014 1:56 PM	-	-	04gD0000000LvIV
Record Name: Acme - 600 Desktops (4 records)					
Step 1	Approved	5/6/2014 1:51 PM	15	5/6/2014 2:06 PM	04gD0000000LvIa
Step 2	Approved	5/6/2014 2:06 PM	299	5/6/2014 7:05 PM	04gD0000000LvIa
Step 3	Approved	5/6/2014 7:05 PM	56	5/6/2014 8:01 PM	04gD0000000LvIa
Step 4	Approved	5/6/2014 8:01 PM	5	5/6/2014 8:06 PM	04gD0000000LvIa
Record Name: Acme - 700 Desktops (1 record)					
Step 1	Recalled	5/6/2014 1:51 PM	2	5/6/2014 1:52 PM	04gD0000000LvIf
Record Name: Canson - 15 Spider Series 3 Laptops (1 record)					
Step 1	Pending	5/6/2014 1:51 PM	-	-	04gD0000000LvIb
Record Name: Canson - 18 Spider 3 Series Laptops (1 record)					
Step 1	Rejected	5/6/2014 1:51 PM	316	5/6/2014 7:07 PM	04gD0000000LvIc
Record Name: Canson - Laser free memory upgrade (4 records)					
Step 1	Approved	5/6/2014 1:51 PM	315	5/6/2014 7:06 PM	04gD0000000LvIW
Ste	A	5/6/2014 7:06 PM	2	5/6/2014 7:08 PM	D0000000LvIW

Notice that the previous sample report doesn't include the approvers for each step and the elapsed time for each approval request. To get this information, run a SOQL query by using the approval process instance ID from the report. The following sample SOQL query obtains the `ActorID` (user or queue that received the approval request) and the `ElapsedTimeInHours` (elapsed time since the approval request was sent) for the first pending step in the report.

```
SELECT ActorId,ElapsedTimeInHours FROM ProcessInstanceWorkitem where processInstanceId =
'04gD0000000LvIV'
```

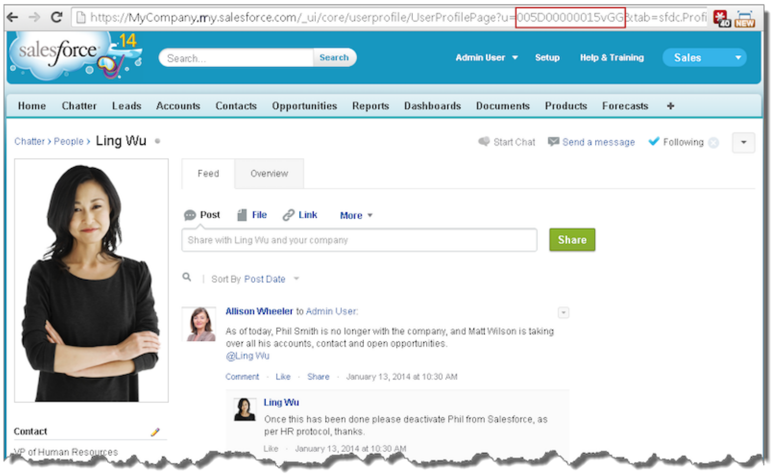


SELECT ActorId, ElapsedTimeInHours FROM ProcessInstanceWorkitem where processInstanceId = '04gD0000000LvIV'

Query Results - Total Rows: 1

ActorId	ElapsedTimeInHours
005D00000015vGGIAY	29.820833333333333

The sample query has only one result, and you can view that approver's user profile page by appending the resulting `ActorID` to the organization's base URL (`https://`*`MyDomainName`*`.my.salesforce.com/005D00000015vGGIAY`), which gets redirected to the user profile page.



Allison Wheeler to Admin User:

As of today, Phil Smith is no longer with the company, and Matt Wilson is taking over all his accounts, contact and open opportunities. @Ling Wu

Comment · Like · Share · January 13, 2014 at 10:30 AM

Ling Wu

Once this has been done please deactivate Phil from Salesforce, as per HR protocol, thanks.

Like · January 13, 2014 at 10:30 AM

SEE ALSO:

Approval History Reports

Considerations for Approval History Reports

Understand the limitations and special behaviors when you create or view approval history reports, which provide a detailed history of approval processes and steps.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Approval Processes That Were Completed Before or Pending during the Summer '14 Rollout

When Summer '14 became available for your organization, the approval history data was automatically populated for completed and pending approval processes. However, some approval history field values are never populated or are populated only after the approval process instance is next acted upon—such as when a user approves, rejects, or reassigns an approval request—after the Summer '14 rollout. Additional exceptions apply to approval history data that is available only via SOQL queries of certain objects. See ProcessInstance, ProcessInstanceNode, ProcessInstanceStep, and ProcessInstanceWorkitem in the *Object Reference for Salesforce*.

Object	When Fields are Populated
Process Instance	For approval process instances that were completed before the Summer '14 rollout, all Process Instance fields are automatically populated, with one exception: `Completed Date` is never populated for approval process instances that were completed before January 1, 2013. For approval process instances that were pending during the Summer '14 rollout, all Process Instance fields are automatically populated, with two exceptions: `Completed Date` and `Last Actor: Full Name` are populated only after the approval process instance is complete.

Object	When Fields are Populated
Process Instance Node	Never populated for approval process instances that were completed before the Summer '14 rollout. For approval process instances that were pending during the Summer '14 rollout, all Process Instance Node fields are populated only after the approval process instance is next acted upon after the Summer '14 rollout.

Considerations for the Sandbox Environment

If you copy approval history data to a sandbox, some field values are overwritten and don't reflect the actual approval history.

Object	Field	When an existing process instance or process instance node record is copied to a sandbox...
Process Instance	`Submitted Date`	This value is overwritten by the date and time when the process instance record is copied to the sandbox.
	`Submitter: Full Name`	This value is overwritten by the name of the user who copied the process instance record to the sandbox.
Process Instance Node	`Step Start Date`	This value is overwritten by the date and time when the process instance node record is copied to the sandbox.

SEE ALSO:

Fields Available for Approval History Reports

Approval History Reports

Manage Multiple Approval Requests

Transfer multiple approval requests from one user to another or remove multiple approval requests from the approval process.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Transfer Pending Approval Requests

If users move to a new role before they complete all their pending approval requests, transfer the remainder to another user.

Remove Pending Approval Requests

If you want to clean up old approval requests—such as to delete an approval process—remove them from your Salesforce org. After approval requests are removed, the associated records are unlocked and removed from all approval processes, so they no longer appear on the approver's list of pending approval requests.

SEE ALSO:

Considerations for Managing Approvals

Transfer Pending Approval Requests

If users move to a new role before they complete all their pending approval requests, transfer the remainder to another user.

1. From Setup, enter `Mass Transfer Approval Requests` in the `Quick Find` box, then select **Mass Transfer Approval Requests**.
2. Search for the approval requests to transfer.
3. Select **Mass transfer outstanding approval requests to a new user**.
4. Look up and select the user to transfer the requests to.

   Make sure that the user can view the records that are associated with the approval requests.
5. Add comments.

   The comments you enter display on the Approval History related list.
6. Select each approval request that you want to transfer.
7. Click **Transfer**.

SEE ALSO:

Remove Pending Approval Requests

Considerations for Managing Approvals

Manage Multiple Approval Requests

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To transfer multiple approval requests:

- Transfer Leads

  AND

  Transfer Record

Remove Pending Approval Requests

If you want to clean up old approval requests—such as to delete an approval process—remove them from your Salesforce org. After approval requests are removed, the associated records are unlocked and removed from all approval processes, so they no longer appear on the approver's list of pending approval requests.

1. From Setup, enter `Mass Transfer Approval Requests` in the `Quick Find` box, then select **Mass Transfer Approval Requests**.
2. Search for the approval requests that you want to remove.
3. Select **Mass remove records from an approval process**.
4. Add comments.

   The comments you enter display on the Approval History related list.
5. Select each approval request to remove from the approval process.
6. Click **Remove**.

SEE ALSO:

Transfer Pending Approval Requests

Considerations for Managing Approvals

Manage Multiple Approval Requests

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To remove multiple approval requests:

- Transfer Leads

  AND

  Transfer Record

Approval Requests for Users

Your admin can set up approval processes that let you and other users submit records for approval, which results in *approval requests*.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Submit a Record for Approval
Depending on your org's customizations, you can submit a record for approval directly from that record.

Withdraw an Approval Request
If you submitted a record for approval but suddenly must update information in the record, recall the approval request. However, whether you can recall an approval request depends on how your admin configured the approval process that the record was submitted to.

Respond to an Approval Request
When you receive an approval request, respond to it by approving, rejecting, or reassigning it. Depending on which Salesforce experience you're using, you have different options. Approval request comments are limited to 4,000 characters. In Chinese, Japanese, or Korean, the limit is 1,333 characters.

What Does This Approvals Error Mean?
Here are some errors that you can come across when you submit a record for approval or respond to an approval request.

Approval History Status
To track where a record is in an approval process, view its Approval History related list.

Approval User Preferences
Identify a delegated approver and control whether you receive approval request emails.

Submit a Record for Approval

Depending on your org's customizations, you can submit a record for approval directly from that record.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To submit a record for approval:
- Read on the record

1. Go to the record that you want to submit for approval.
2. Make sure it's ready to be submitted.

   Before you can submit a record for approval, it must meet the criteria for an active approval process. If you're not sure what the requirements are, ask your admin.
3. Click **Submit for Approval**.

   If an approval process applies to the record, Salesforce begins the approval process. This button isn't available after the record has been submitted.

To keep tabs on the progress of your submitted approval, we recommend following the approval record in Chatter.

SEE ALSO:

Withdraw an Approval Request

Approval User Preferences

Approval Requests for Users

Withdraw an Approval Request

If you submitted a record for approval but suddenly must update information in the record, recall the approval request. However, whether you can recall an approval request depends on how your admin configured the approval process that the record was submitted to.

1. Go to the detail page for the record associated with the approval request.
2. In the Approval History related list, recall the approval request.

SEE ALSO:

Submit a Record for Approval

Approval User Preferences

Approval Requests for Users

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To withdraw an approval request:

- Read on the Record

Respond to an Approval Request

When you receive an approval request, respond to it by approving, rejecting, or reassigning it. Depending on which Salesforce experience you're using, you have different options. Approval request comments are limited to 4,000 characters. In Chinese, Japanese, or Korean, the limit is 1,333 characters.

Respond from...	**Lightning Experience**	**Salesforce Classic**	**The Salesforce Mobile App**
An in-app notification	✓		✓
An email notification	✓	✓	✓
The record	✓	✓	✓
Chatter	✓	✓	✓
Home	✓	✓	
Slack	✓		✓

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To respond to an approval request from within Salesforce:

- Read on the associated record

To respond to an approval request from an email:

- API Enabled

In-App Notification

Depends on the `Receive Approval Request Emails` field in your approver preferences. If notifications are enabled for your org, you receive a notification whenever you receive an approval request email.

- Respond from the notification if your admin enabled actionable notifications.
- To open the approval request, click the notification.

Email Notification

Depends on the `Receive Approval Request Emails` field in your approver preferences.

- To open the approval request, click the link in the email.
- Reply to the email if your admin enabled email approval response.

Record

Respond from the Approval History related list.

Chatter

Depends on if your admin has enabled Approvals in Chatter and you haven't opted out of receiving approval requests through Chatter posts.

- Respond from the post if your admin enabled actionable notifications.
- Click the name of the record, then respond from the Approval History related list.

Home

Depends on if your admin added the Items to Approve component to your home page. From the Home tab, respond from the Items to Approve component.

Tip: From this component in Salesforce Classic, you can respond to multiple requests at once.

Slack

Slack notifications are enabled by default. If your admin hasn't disabled Slack notifications, an approver can respond to a request from the **Messages** tab on the Salesforce Digital HQ app in Slack. A **Show More** link opens the details of the approval request in Salesforce.

Respond to an Approval Request via Email

If your admin enabled email approval response, you can approve or reject requests by responding to the email notification. It doesn't matter which Salesforce experience or mobile email client you're using. Delegated approvers can also respond to approval requests by email.

Troubleshoot Email Responses to Approval Requests

When email responses aren't working correctly, review these common issues.

SEE ALSO:

Approval User Preferences

Approval Requests for Users

Respond to an Approval Request via Email

If your admin enabled email approval response, you can approve or reject requests by responding to the email notification. It doesn't matter which Salesforce experience or mobile email client you're using. Delegated approvers can also respond to approval requests by email.

Email approval response works in all languages that Salesforce supports. The response word or phrase is checked using the current user language dictionary. If no matches are found, the response word or phrase is checked in all other language dictionaries.

1. In the first line of your reply to the email notification, enter one of the supported response words.

   Periods and exclamation marks are allowed at the end of the word.

Approval Words	Rejection Words
approve	reject
approved	rejected
yes	no

2. Optionally, in the second line of your reply, add comments.
3. Send the email.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To respond to an approval request via email:
- API Enabled

SEE ALSO:

Approval User Preferences

Approval Requests for Users

Troubleshoot Email Responses to Approval Requests

When email responses aren't working correctly, review these common issues.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

I'm not receiving approval requests by email.

Here are a few possible reasons why.

- Your approval preferences opt you out of approval request emails.
- Your mail server thinks the approval request email is spam. Contact your email admin, who can check the logs of all inbound email to see if it's being delivered, rejected, or marked as spam.
- Your email admin has to add the Salesforce email addresses that the approval requests come from to the allowed email addresses for your mail server.
- Email delivery time can vary based on your ISP or connection.

My response wasn't delivered.

- An email approval request can only be processed one time. If another user has responded to the approval request before you do, you get an error.
- You must have the "API Enabled" user permission to respond to approval requests by email.

I received an email that said, "`The word used to approve or reject the item was not understood.`"

Salesforce doesn't process replies to error emails. Reply again to the original email notification, but this time use one of the supported response words on page 876.

I received an email that said, "`You are not authorized to update the referenced object.`"

The approval request email is tied to your email address. If you forward the request to another email address, or if your email client lets you respond from multiple email addresses, you receive this error. Reply again to the original email notification, but this time reply from the same email address that received the email approval request.

SEE ALSO:

- What Does This Approvals Error Mean?
- Respond to an Approval Request via Email
- Approval Requests for Users

What Does This Approvals Error Mean?

Here are some errors that you can come across when you submit a record for approval or respond to an approval request.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Manager undefined

This approval request requires the next approver to be determined by the *`Field Name`* field.

This value is empty.

Salesforce tried to route the approval request based on a hierarchical field, such as `Manager`. However, the field has no value or specifies an inactive user. This error can occur when someone submits a record for approval or when an approver responds to an approval request.

Required fields are missing: [*`FieldName`*].

The approval process includes a field update that fails standard validation rules for the identified field. This error can occur even if the field isn't visible on your page layout.

Note: Salesforce doesn't check whether field updates pass *custom* validation rules on fields.

SEE ALSO:

- Troubleshoot Email Responses to Approval Requests
- Approval Requests for Users

Approval History Status

To track where a record is in an approval process, view its Approval History related list.

Status	Definition
Submitted	The record has been submitted for approval.
Pending	The record has been submitted for approval and is awaiting approval or rejection.
Approved	The record has been approved.
Rejected	The record has been rejected.
Reassigned	The record has been submitted for approval but assigned to a different approver.
Recalled	The record was submitted for approval but recalled from the approval process.

EDITIONS

Available in: Salesforce Classic (not available in all orgs)

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Approval Processes

Submit a Record for Approval

Respond to an Approval Request

Approval Requests for Users

Approval User Preferences

Identify a delegated approver and control whether you receive approval request emails.

From your personal settings, enter *Approver Settings* in the Quick Find box, then select **Approver Settings**. No results? Enter *Personal Information* in the Quick Find box, then select **Personal Information**.

Field	Description
Delegated Approver	Your alternate approver. If populated, this user receives the same approval requests as you do. Delegated approvers can't reassign approval requests; they can only approve or reject approval requests. Internal Salesforce users are listed by and can be added using the Delegated Approver lookup field. Use Data Loader and a comma-delineated (CSV) file to add users with communities licenses as Delegated Approvers. The CSV uses the CommunityUserId rather than the UserId for DelegatedApproverId. Communities licenses are used with Experience Cloud sites and legacy portals.
Manager	Depending on how your admin sets up approval processes, requests for approval can automatically be routed to your manager.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Field	Description
`Receive Approval Request Emails`	Controls whether you receive approval request notifications by email, in the Salesforce mobile app, or in Lightning Experience. If you select **Never**, you don't receive approval request notifications. However, you still receive approval request emails from a queue, depending on how your admin configured the queue email.

Opt Out of Approval Request Posts in Chatter

By default, after your org enables Approvals in Chatter, you're notified about approval requests by email and a Chatter post. To stop seeing the Chatter posts, opt out. If you do opt out, the posts don't appear in your feed but they do appear in the associated record's feed.

What Happens When You Opt Out of Chatter Approval Requests?

By default, when your org has Approvals in Chatter enabled, you receive approval request notifications by email and Chatter. Here's what happens when you opt out of the Chatter posts.

SEE ALSO:

Approval Requests for Users

Personalize Your Salesforce Experience

Opt Out of Approval Request Posts in Chatter

By default, after your org enables Approvals in Chatter, you're notified about approval requests by email and a Chatter post. To stop seeing the Chatter posts, opt out. If you do opt out, the posts don't appear in your feed but they do appear in the associated record's feed.

Available in: both Salesforce Classic and Lightning Experience
Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, **Developer**, and **Contact Manager** Editions

USER PERMISSIONS

To view an approval request post for a record:

- Read on the record

1. In the page banner, click your profile avatar, and select **My Settings** (Salesforce Classic) or **Settings** (Lightning Experience).
2. Enter *`My Feeds`* in the `Quick Find` box, then select **My Feeds**.
3. Deselect **Receive approval requests as posts.**

   You see this setting only when approvals are enabled in your org.
4. Save your changes.

SEE ALSO:

Approval User Preferences

Personalize Your Salesforce Experience

What Happens When You Opt Out of Chatter Approval Requests?

By default, when your org has Approvals in Chatter enabled, you receive approval request notifications by email and Chatter. Here's what happens when you opt out of the Chatter posts.

Available in: both Salesforce Classic and Lightning Experience
Available in: **Group**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, **Developer**, and **Contact Manager** Editions

- If you opt out while an approval that you're assigned to is in progress, you see notification posts if you're following the approval record.
- If you're following the approval record, you see approval posts from the record with non-approver content.
- For approval notification posts that you've already received, you see non-approver content.
- The Approve and Reject buttons are removed from existing approval posts in your feed.

SEE ALSO:

Approval User Preferences

Opt Out of Approval Request Posts in Chatter

Approval Process Terminology

Salesforce uses this terminology for approval processes.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approval Actions

An approval action occurs when all required approvers approved a step.

Approval Process

An approval process automates how records are approved in Salesforce. An approval process specifies each step of approval, including from whom to request approval and what to do at each point of the process.

Approval Request

An approval request is an email, Salesforce app notification, Lightning Experience notification, or Chatter post. The approval request notifies the recipients that a record was submitted for them to approve.

Approval Steps

Approval steps define the chain of approval for a particular approval process. Each step determines:

- Which records can advance to that step
- To whom to assign approval requests
- Whether to let each approver's delegate respond to the requests

The first step specifies what to do if a record doesn't advance to that step. Later steps specify what happens if an approver rejects the request.

Assigned Approver

The assigned approver is the user responsible for responding to an approval request.

Delegated Approver

A delegated approver is someone appointed by an assigned approver as an alternate for approval requests.

Note: Internal Salesforce users are listed by and can be added using the Delegated Approver lookup field. Use Data Loader and a comma-delineated (CSV) file to add users with communities licenses as Delegated Approvers. The CSV uses the `CommunityUserId` rather than the `UserId` for `DelegatedApproverId`. Communities licenses are used with Experience Cloud sites and legacy portals.

Email Approval Response

Email approval response lets users respond to approval requests by replying to an email notification.

Initial Submission Actions

An initial submission action occurs when a user first submits a record for approval. By default, the record is locked.

Final Approval Actions

Final approval actions occur when all required approvals were obtained.

Final Rejection Actions

A final rejection action occurs when an approver rejects the request and it moves to the final rejection state.

Outbound Message

An outbound message sends information to a designated endpoint, like an external service. You can configure outbound messages from Setup. Configure the external endpoint and use SOAP API to create a listener for the messages.

Process Instance

A process instance represents one instance of an approval process. A new process instance is created each time a record is submitted for approval.

Process Instance Node

A process instance node represents an instance of an approval step. The system creates a process instance node each time a record enters a step in an approval process. The system doesn’t create a process instance node when the record doesn’t meet the step criteria, or the approval process instance is completed without entering the step.

Recall Actions

A recall action occurs when a submitted approval request is recalled. By default, the record is unlocked.

Record Locking

Record locking prevents users from editing a record, regardless of field-level security or sharing settings. By default, Salesforce locks records that are pending approval. Only admins can edit locked records.

SEE ALSO:

Approval Processes

Automated Actions

Set Up an Approval Process

Modify Process Automation Settings

Enable or disable features related to flows, processes, workflow rules, and approval processes.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Processes and flows are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Approvals and Workflow are available in **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

User Permissions Needed	
To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

- Identify Your Salesforce Org's Default Workflow User
- Override the Sender for Email Approval Notifications
- Let Users Respond to Approval Requests via Email
- Let Users Pause Flow Interviews
- Restrict Who Can Resume Shared Flow Interviews
- Enable Lightning Runtime for Custom Buttons and Links
- Require Access to Automation Home Charts (Beta)
- Control What Happens When a Flow Tries to Set Values for Read-Only Fields
- Select Flow and Process Error Email Recipients
- Deploy Processes and Flows as Active

Legacy Salesforce Flow Features

Legacy Salesforce Flow features include Process Builder and Workflow Rules.

Important: Starting in Winter '23, you can't create new processes or workflow rules. You can still activate, deactivate, and edit any existing processes and workflow rules. To migrate existing processes or workflow rules, use the process in Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool on page 894. For new automations, create flows in Flow Builder on page 16.

Switch to Flow Builder Learning Map

Flow Builder combines the capabilities of workflow rules and Process Builder in a single point-and-click tool, making it easier to create a triggered process. If you created triggered processes with workflow rules or Process Builder, use the Migrate to Flow tool to move them to Flow Builder. Start by migrating and testing in a sandbox environment before moving your new flows to production.

Process Builder

Many of the tasks you assign, the emails you send, and other record updates are vital parts of your standard processes. Instead of doing this repetitive work manually, you can configure flows or processes to do it automatically. We strongly recommend using Flow Builder, but Process Builder can also help you automate your business processes and give you a graphical representation as you build it.

Workflow Rules

Workflow rules let you automate standard internal procedures and processes to save time across your org. A workflow rule is the main container for a set of workflow instructions. These instructions can always be summed up in an if/then statement.

Switch to Flow Builder Learning Map

Flow Builder combines the capabilities of workflow rules and Process Builder in a single point-and-click tool, making it easier to create a triggered process. If you created triggered processes with workflow rules or Process Builder, use the Migrate to Flow tool to move them to Flow Builder. Start by migrating and testing in a sandbox environment before moving your new flows to production.

Get Started: Time to Plan Salesforce Admins Blog: Go with the Flow: What's Happening with Workflow Rules and Process Builder? Salesforce Help:Equivalent Features in Flows and Workflow Rules Salesforce Help:Migrate to Flow Tool Considerations	**Get Ready: Learn Flow** Trailhead: Build Flows with Flow Builder Trailhead: Automate Your Business Processes with Salesforce Flow Salesforce Admins Blog: 5 New Videos to Help You Understand Tricky Flow Concepts Salesforce Admins Blog: The Ultimate Guide to Flow Best Practices and Standards Flow Playbook: 3 Ways to Innovate Fast with Enterprise-Scale Automation Trailblazer Community: Salesforce Automation How To: Transition to Salesforce Flow Salesforce Admins YouTube Channel: Automate This!	**Get Moving: Start Your Migration** Salesforce Help: Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool Salesforce Architect Guide: Designing Your Record-Triggered Automation Salesforce Help:Scheduled Paths Salesforce Help:Debug a Flow in Flow Builder Video: Troubleshooting Flow Errors Using the Debug Button
Get Building: Dive into Triggers	**Get Visual: Manage Your Triggers**	

Salesforce Architects: Record-Triggered Automation Guide Salesforce Help:Record Triggers for Flows That Make Before-Save Updates on page 32 Define the Run Order of Record-Triggered Flows for an Object on page 33 Salesforce Dev Guide: Triggers and Order of Execution Trailhead: Record-Triggered Flows	Salesforce Admins Blog: Use Flow Trigger Explorer to Easily View All Your Triggered Automation in One Place Salesforce Help: Manage Record-Triggered Flows on page 31 Apex Blog: Flow Trigger Explorer	

Equivalent Features in Flows and Workflow Rules
Features and fields in workflow rules correspond to certain things in flows. Use the equivalent features and fields to build flows that can replace your workflow rules.

Planning Your Switch to Flow Builder
Workflow Rules and Process Builder are no longer the preferred tools for automating your business processes. With their pending retirement, now is the time to go with Flow Builder as the future of automated processes. Flow Builder is a foundation for the future and offers built-in extensibility, application lifecycle management, and faster performance.

Migrate to Flow Tool Considerations
Review considerations and supported workflow rules and processes for the Migrate to Flow tool.

Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool
Use the Migrate to Flow tool to convert your Process Builder processes and workflow rules into Flow Builder, including scheduled actions. The tool also supports partial migration of processes for most actions.

Sample Migration to a Flow: Workflow Rule with an Email Alert
The majority of workflow rules are used to send email alerts or perform same-record field updates. While these types of workflow rules have a reputation for being fast, triggered flows are even faster. It's time to migrate your workflow rules to Flow Builder.

Equivalent Features in Flows and Workflow Rules

Features and fields in workflow rules correspond to certain things in flows. Use the equivalent features and fields to build flows that can replace your workflow rules.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

General

Workflow Rules	Flows
Create New Rule	When creating a flow, select **Record-Triggered Flow**.
Description	Use the Description field. The field is available when saving the flow.
Email Alerts	Use the Action element.
Outbound Messages	Use the Action element.

Workflow Rules	Flows
Rule Name	Use the Flow Label field. The field is available when saving the flow.
Select Object	Use the Object field. The field is available in the Start element, in the Select Object section.
Tasks	Use the Create Records element. For How to Set the Record Fields, select **Use separate resources, and literal values** and then select the **Task** object.
Time Triggers	Use a scheduled path. To create a scheduled path, in the Start element, optimize the flow for **Actions and Related Records**. Then, from the flow canvas, select **Add Scheduled Path** on the Start node.

Evaluation Criteria

In flows, evaluation criteria are defined in the Start element, in the Set Entry Conditions section.

Workflow Rules	Flows
created	Trigger the Flow When: • A record is created
created, and any time it's edited to subsequently meet criteria	Trigger the Flow When: When to Run the Flow for Updated Records: • A record is created or updated • Only when a record is updated to meet the condition requirements
created, and every time it's edited	Trigger the Flow When: When to Run the Flow for Updated Records: • A record is created or updated • Every time a record is updated and meets the condition requirements

To manually convert workflow rules with unsupported use cases in entry criteria, create a Decision element inside the Flow. Then recreate the workflow rule steps with the condition builder.

Rule Criteria

In flows, rule criteria are defined in the Start element, in the Set Entry Conditions section.

Workflow Rules	Flows
Run this rule if the criteria are met	For Condition Requirements, select **All Conditions Are Met (AND)**
Run this rule if the formula evaluates to true	For Condition Requirements, select **Formula Evaluates to True**

Field Updates

In flows, field updates are done using the Update Records element. In the Update Records element, for How to Find Records to Update and Set Their Values, select **Use the {object name} record that triggered the flow**. In flows, you can enter values directly, or use formulas or references for field values. You can also update multiple fields in a single flow. To improve performance, place field updates in a flow optimized for fast field updates (before-save).

Flows don't support field update notifications for the Owner field.

A Get Records element can be required to reference certain elements, like users or groups.

Task Fields

To create a task in a flow, use the Create Records element.

Workflow Rules	Flows
Assigned To	**API Name** OwnerId **Field Name** Assigned To
Comments	**API Name** Description **Field Name** Comments
Due Date	**API Name** ActivityDate **Field Name** Due Date
Object	**API Name** WhatId **Field Name** Related To
Priority	**API Name** Priority **Field Name** Priority
Protected Component	N/A
Status	**API Name** Status **Field Name** Status

Workflow Rules	Flows
Subject	**API Name** Subject **Field Name** Subject
Unique Name	N/A

Operators

Flow Builder shows only the operators that are relevant to the field. `Less Than`, `Greater Than`, `Less Than or Equal`, and `Greater Than or Equal`, appear only for number fields.

Workflow Rules	Flows
`does not contain`	N/A
`equals`	`Equals`
`excludes`	N/A
`greater or equal`	`Greater Than or Equal` (only number fields)
`greater than`	`Greater Than` (only number fields)
`includes`	N/A
`less or equal`	`Less Than or Equal` (only number fields)
`less than`	`Less Than` (only number fields)
N/A	`Ends With`
N/A	`Is Null`
N/A	`Is Changed`
`not equal to`	`Does Not Equal`
`starts with`	`Starts With`
`within`	N/A

To manually convert workflow rules that use the `does not contain` operator, use custom condition logic. For example, create a condition that uses the `Contains` operator in the first condition, and in the condition logic, enter **NOT 1**.

SEE ALSO:

Migrate to Flow Tool Considerations

Planning Your Switch to Flow Builder

Workflow Rules and Process Builder are no longer the preferred tools for automating your business processes. With their pending retirement, now is the time to go with Flow Builder as the future of automated processes. Flow Builder is a foundation for the future and offers built-in extensibility, application lifecycle management, and faster performance.

There are several areas to focus your efforts as plan your switch to Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Analyze Your Automation

To start your migration journey, analyze your existing automation.

- Categorize your most commonly used automation types.
- Observe your org's flow activity in reports and dashboards, such as total errors and total started automations.
- View your Flow Interview Logs and Flow Interview Log Entries.
- Run the Sample Flow Report: Screen Flows report. Use the reports to examine run-time details about your screen flows.
- Use the metrics to discover usage patterns and in turn to optimize your screen flows for users.

Migrate in a Sandbox First

It's critical to keep your existing data safe before you make any changes. Working in a sandbox ensures that no data is harmed as you make you move to Flow Builder.

Catalog Your Current Automation

There are many ways to create a catalog. You can create a spreadsheet. Organize your automations by Object. Include the Category, Entry Criteria, and Related Actions as you catalog. You can create a diagram to aid in your visualization.

Identify and Remove Redundant Processes

Evaluate whether processes are still needed or can be improved. Common culprits of redundancy can include:

Redundancy	Example
Multiple entry points per Object	There are multiple processes and Workflow Rules on an Account update performing overlapping business cases.
Recursive updates	• Process Builder 1 update triggers Process Builder 2. • Process Builder 2 triggers Process Builder 3. • Process Builder 3 re-triggers Process Builder 1.
Conflicting Actions	A Fast Field Update flow that updates a field while a Workflow Rule overrides the same field value.

Prioritize Which Processes to Migrate First

Migrate the processes that speed up the record updates and take less time and effort to migrate. Start with a Single Object. Pick the object that has the least number of Workflow Rules and processes associated with it. Processes that send email alerts or perform same-record field updates are good beginner options for migration.

Structure Your Automation

As you think about your business needs, here are some common considerations:

- Performance: Can you use Entry Conditions or other optimizations to reduce unnecessary operations?
- Maintenance and Change Management: Who is responsible for this business process? What is the likelihood of change or iteration?
- Migration from Workflow Rules and processes to flows isn't one to one. You don't always have to create a flow for each process you're migrating.
- Analyze your existing flows and see if there are corresponding elements that you can update, or incorporate new actions in an existing flow.
- Consider whether the more complex automation processes can be reused and implemented as subflows.
- Review whether there's a better solution that doesn't involve automation.

  Example: A Workflow Rule or process only updates a field on the Case object after it's created. Replace it with a formula field instead.

Think Beyond One Flow Per Object

You can design your automation to have multiple flows per object. For a more scalable future, order your flows with the Trigger Order option. Use Flow Trigger Explorer to assign priority values to your flows. With this tool, multiple flows per object are manageable.

Optimize Your Record-Triggered Automations

Building efficient record-triggered flows can help minimize some flow limits. Here are the options you can select to improve efficiency as you build your flows.

Option	When the Flow Runs	When to Use It	Benefit
Fast Field Update	During the record update that triggered the flow and before that update is saved.	To update the record that triggered the transaction.	Optimal performance because it's limited to updating the triggering record
Related Records and Actions	During the record update that triggered the flow and after that update is saved.	• When you create, update, or delete other records. • Calling subflows • Calling actions, such as send email alert or post to Chatter.	Automating common processes triggered by record changes,
Run Asynchronously	Immediately after the record update that triggered the flow is complete.	When you execute more advanced scenarios like sending requests to external systems or performing other long-running processes.	Avoids slowing down or blocking the record update that triggered the flow.

Optimize Your Entry Conditions

Set Entry Conditions to decrease the performance impact. Used effectively, Entry Conditions prevent automation from running unnecessarily and improve performance. Set entry conditions to run a flow when a record is created or edited and a field has a specific value. Or set entry conditions when a record is created or edited and a field IS CHANGED to a specified value. If you only check the field values when the record is created or edited, there are no additional steps beyond creating the entry conditions. To create entry conditions that check what the field values are changed to when the record is created or edited, enable the Run When Conditions Met setting. Enabling this setting prevents repeat operations and maintains consistency.

Replace Time-Dependent Workflow Rules with Scheduled Paths

Add a Scheduled Path to a record-triggered flow. Scheduled Paths occur in the future, after the trigger has fired, based on dates and times. You can schedule such actions as reminders or follow-ups based on dates in the record that triggered the automation, such as Close Date. This feature also rechecks the entry conditions.

Example: Set your entry condition to Status = Escalated and then have an automation that sends a reminder two days before close. The reminder only sends if the status remains escalated.

Order Your Automation

You can use Flow Trigger Explorer to view the order in which your automation runs or to reorder flows. The flow executes in the order described to minimize disruption from other automation, managed packages, or movement between orgs. With flow trigger ordering, you can assign a priority value to your flows. To see all of the associated flows that run when a record is created, updated, or deleted, select an object. This action allows for easy navigation between flows that run under the same circumstances.

Add Descriptions to Your Flows

It's important to remember that documentation is as important as automation. When building new flows, document your work. Enter clear, unique names for objects. To describe your intent, use the Description field on every element in all of your flows. This documentation helps to avoid any confusion as to the purpose of the automation.

Test in a Sandbox

To protect the data in your org, always test in a sandbox before moving any changes to production.

Deactivate Old Automations as You Rebuild

By default, active processes and flows are deployed as inactive. After deployment, manually reactivate the new versions and deactivate the old.

Resources

The Salesforce Admins: Automation page is a great resource to help you start automating business processes. You can explore flow templates on AppExchange, or navigate to an automation tool directly.

SEE ALSO:

Video: Automate This: Migrate Workflow Rules and Processes to Flow

Success Events: Implement: Platform: Transition to Salesforce Flow

Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool

Equivalent Features in Flows and Workflow Rules

Migrate to Flow Tool Considerations

Developer Guide: Triggers and Order of Execution

Migrate to Flow Tool Considerations

Review considerations and supported workflow rules and processes for the Migrate to Flow tool.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Considerations for Migrating a Process to a Flow

Review considerations and supported Process Builder processes for the Migrate to Flow tool.

Considerations

Processes with recursion aren't fully supported. When a process with recursion is migrated, the record is evaluated only one time. Test and make sure that any processes with recursion work as intended after migration.

Processes are migrated as Actions and Related Record-optimized (after-save) flows. If necessary, you can edit and optimize the flow for Fast Field Updates (before-save) after the flow is migrated.

The invoke flow action is migrated as a subflow element instead of an invocable flow action. Subflows run in the same transaction as the parent flow. Any processes with invoke flow actions involving external callouts, external actions, or pauses must be redesigned using an asynchronous path.

You can migrate Process Builder's scheduled actions only if you select the single criteria associated with the scheduled action. If multiple criteria are selected, no scheduled actions are migrated. After migration, scheduled actions become Flow Builder's scheduled paths. In the flow, migrated scheduled actions follow the naming convention `ScheduledPath__#`. At run time, the new flow checks for pending actions from the original migrated process and then deletes them. If a record is updated, any pending scheduled actions are moved to the proper scheduled path or canceled if the record no longer meets the criteria. If a record isn't updated when the scheduled action is executed, it executes the process' scheduled action. See Monitor Your Processes' Pending Scheduled Actions.

You can't migrate a cross-object reference in a formula.

You can migrate a process that uses a custom metadata reference in a formula. After the migration, the custom metadata reference is used in flow formulas, but you can't configure it by using the resource picker.

When migrating a time-based process, you must migrate each outcome to its own scheduled action flow. Then activate the new flows and deactivate the process.

Supported Processes

The Migrate to Flow tool supports only record-triggered processes. Custom event and custom invocable type processes aren't supported. The tool also doesn't support processes that contain custom metadata types or criteria that contain a field that's from a related object (field traversals). For supported processes, you can migrate these action types without additional configuration.

- Record update
- Record create
- Invoke flow
- Invoke Apex
- Email alert

After migration, these action types retain their original positions in the flow, but they require additional configuration to function as expected.

- Post to Chatter
- Quick Action
- Submit for Approval
- Send Custom Notification
- Live Message Notification
- Send Surveys
- Quip-related action types

Considerations for Migrating a Workflow to a Flow

Review considerations and supported workflow rules for the Migrate to Flow tool. Learn manual conversion methods for specific workflow rules.

Considerations

If a workflow rule contains only field updates, the tool converts it into a fast field update (before-save) flow.

Due to their position in the order of execution, record-triggered flows can behave differently from similar workflow rules.

An at-rest pending time-based action is migrated to a scheduled path when the associated record is changed.

Supported Workflow Rules

The Migrate to Flow tool supports workflow rules that contain these items.

- Field-based criteria
- Field updates
- Email alerts
- Outbound messages
- Time-dependent workflow actions
- Rule criteria formulas that are set to true (unless the evaluation criteria are also set to created, and anytime it's edited to subsequently meet the criteria)
- `Equal to` null
- `Not equal` to null
- Rule criteria formulas

Workflow rules that contain the following can't migrate with the Migrate to Flow tool.

- Criteria with no defined workflow actions
- Global variable fields
- Fields on related records
- Record types
- The `does not contain`, `includes`, `excludes`, or `within` operators
- The `greater than`, `greater or equal`, `less than`, `less or equal` operators on picklist fields
- Formulas that use `Hour`, `Minute`, `Second`, `TimeNow`, `TimeValue`, `IsClone`, or `$RecordType`
- Tasks
- Relative date values in date fields
- Multiple currencies

Manual Conversion Methods

Certain features are unsupported by the Migrate to Flow tool, but you can manually convert them.

To manually convert workflow rules with unsupported use cases in entry criteria, create a Decision element inside the Flow. Then recreate the workflow rule steps with the condition builder.

Note: With this method, the flow will always run and check on the decision after entering. This method can impact performance or prevent time-based workflow triggers from migrating.

To manually convert workflow rules that use the `does not contain` operator, use custom condition logic. For example, create a condition that uses the `Contains` operator in the first condition, and in the condition logic, enter **NOT 1**.

To manually convert workflow rules that use tasks, use the Create Records option and create a record of the Task object.

Flows support workflow actions for Email Alerts and Outbound Messages. To add these workflow actions to a flow, use the Action element.

To replicate relative date values, such as `TODAY` or `NEXT WEEK`, use the Decision element.

SEE ALSO:

Equivalent Features in Flows and Workflow Rules

Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool

Planning Your Switch to Flow Builder

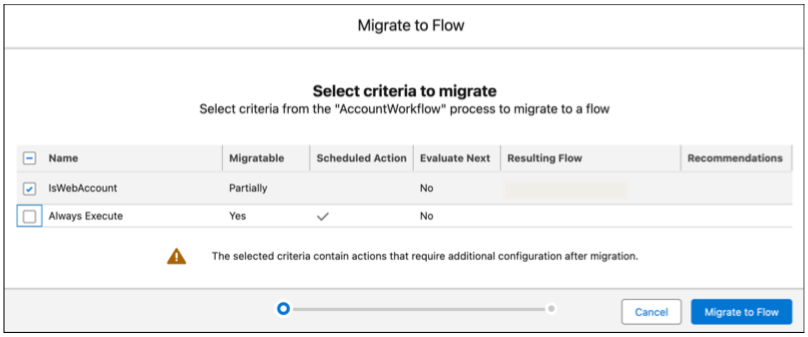
Move Processes and Workflows to Flow Builder with the Migrate to Flow Tool

Use the Migrate to Flow tool to convert your Process Builder processes and workflow rules into Flow Builder, including scheduled actions. The tool also supports partial migration of processes for most actions.

Before moving your new flows to production, start with migrating and testing in a sandbox environment.

1. From Setup, in the Quick Find box, enter `Migrate to Flow`, and then select **Migrate to Flow**.
2. Select the process that you want to convert to a flow.
3. Click **Migrate to Flow**.
4. Select the criteria that you want to migrate.

   If it's a process, the Migratable column indicates whether you can fully or partially migrate the process.



Migrate to Flow

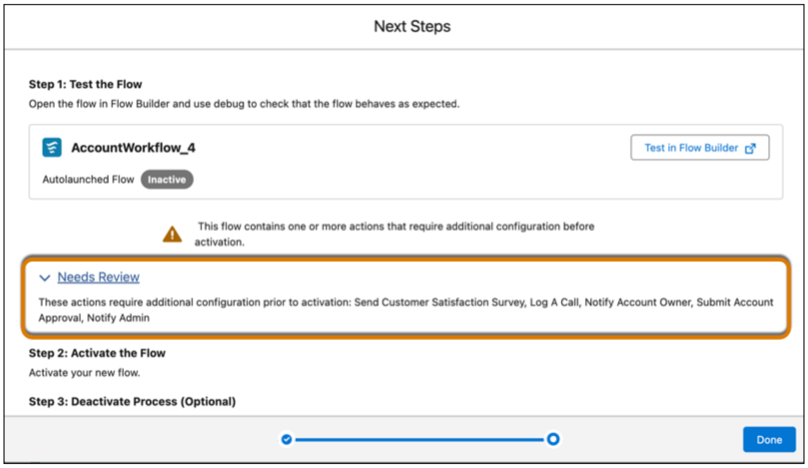
Select criteria to migrate

Select criteria from the "AccountWorkflow" process to migrate to a flow

⊟	Name	Migratable	Scheduled Action	Evaluate Next	Resulting Flow	Recommendations
☑	IsWebAccount	Partially		No		
□	Always Execute	Yes	✓	No		

⚠ The selected criteria contain actions that require additional configuration after migration.

5. Click **Migrate to Flow**.
6. If this is a partial migration of a process, click **Needs Review** when the migration is complete to see the list of actions that require additional configuration.



Next Steps

Step 1: Test the Flow

Open the flow in Flow Builder and use debug to check that the flow behaves as expected.

AccountWorkflow_4

Autolaunched Flow Inactive

⚠ This flow contains one or more actions that require additional configuration before activation.

Needs Review

These actions require additional configuration prior to activation: Send Customer Satisfaction Survey, Log A Call, Notify Account Owner, Submit Account Approval, Notify Admin

Step 2: Activate the Flow

Activate your new flow.

Step 3: Deactivate Process (Optional)

7. After you migrate a process or workflow rule, test the flow in Flow Builder.

EDITIONS

Available in: all editions except **Starter**

USER PERMISSIONS

To open, edit, create, activate, or deactivate a flow in Flow Builder:

- Manage Flow

To create, change, activate, or deactivate workflow rules and actions:

- Customize Application

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

To activate or deactivate processes:

- Manage Flow

  AND

  View All Data

  AND

  Customize Application

8. If everything works as expected, activate the flow.
9. Deactivate the process or workflow rule you migrated to Flow Builder.

Sample Migration to a Flow: Workflow Rule with an Email Alert

The majority of workflow rules are used to send email alerts or perform same-record field updates. While these types of workflow rules have a reputation for being fast, triggered flows are even faster. It's time to migrate your workflow rules to Flow Builder.

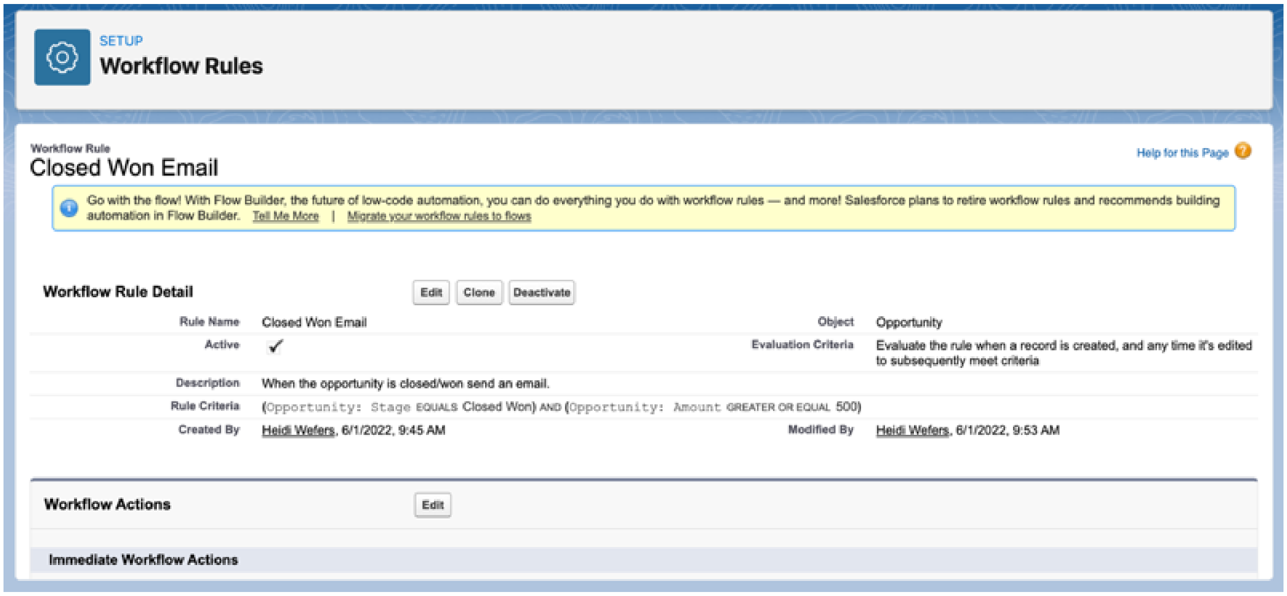
Let's look at a common workflow rule. This rule sends an email alert when an Opportunity is Closed-Won and the Amount is more than $500.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Processes and flows are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Workflow is available in **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions



Workflow Rule

Closed Won Email

Go with the flow! With Flow Builder, the future of low-code automation, you can do everything you do with workflow rules — and more! Salesforce plans to retire workflow rules and recommends building automation in Flow Builder. Tell Me More | Migrate your workflow rules to flows

Workflow Rule Detail

Rule Name	Closed Won Email	Object	Opportunity
Active	✓	Evaluation Criteria	Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria
Description	When the opportunity is closed/won send an email.		
Rule Criteria	(Opportunity: Stage EQUALS Closed Won) AND (Opportunity: Amount GREATER OR EQUAL 500)		
Created By	Heidi Wefers, 6/1/2022, 9:45 AM	Modified By	Heidi Wefers, 6/1/2022, 9:53 AM

Workflow Actions

Immediate Workflow Actions

This workflow rule can be built easily in Flow Builder. Here are the elements in a workflow rules and their flow equivalent.

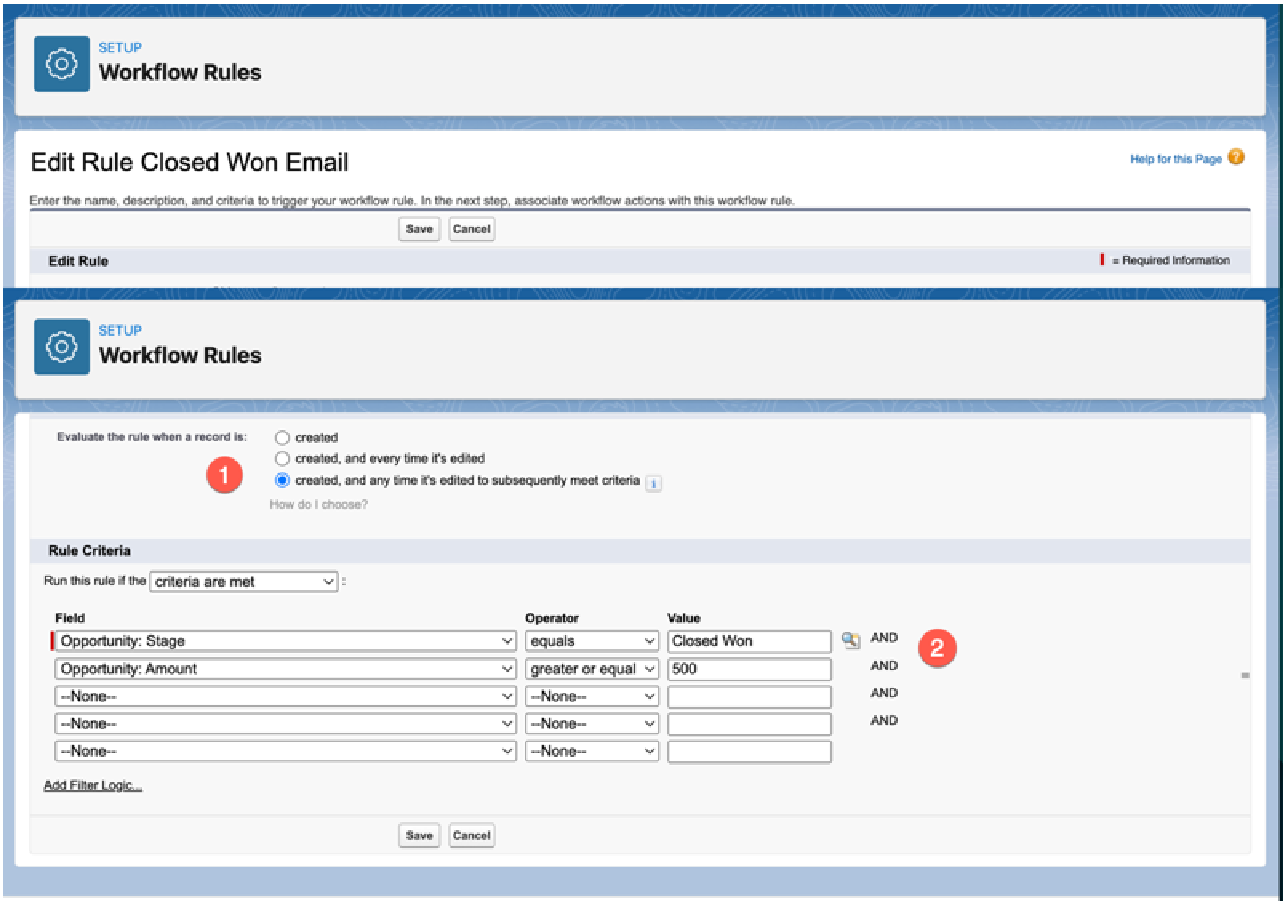
Workflow Rule Element	What to Use in Flow
Create New Rule	Create a Record-Triggered Flow.
Description	Use the Description field. This field is available when you save the flow.
Email Alerts	Use Send Email in the Action element. Existing Email Alerts can be selected.

Workflow Rule Element	What to Use in Flow
Rule Name	Use the Flow Label field. This field is available when you save the flow.
Select Object	Use the Object field. This field is available in the Select Object section of Configure Start.

Evaluation Criteria

When you build a flow, the evaluation criteria are defined in the Start element, in the Set Entry Conditions section. Use as specific as possible Entry Criteria. This way you don't run a flow when you don't need to.

Workflow Rule



Edit Rule Closed Won Email

Enter the name, description, and criteria to trigger your workflow rule. In the next step, associate workflow actions with this workflow rule.

Edit Rule | = Required Information

Evaluate the rule when a record is:

○ created
○ created, and every time it's edited
● created, and any time it's edited to subsequently meet criteria

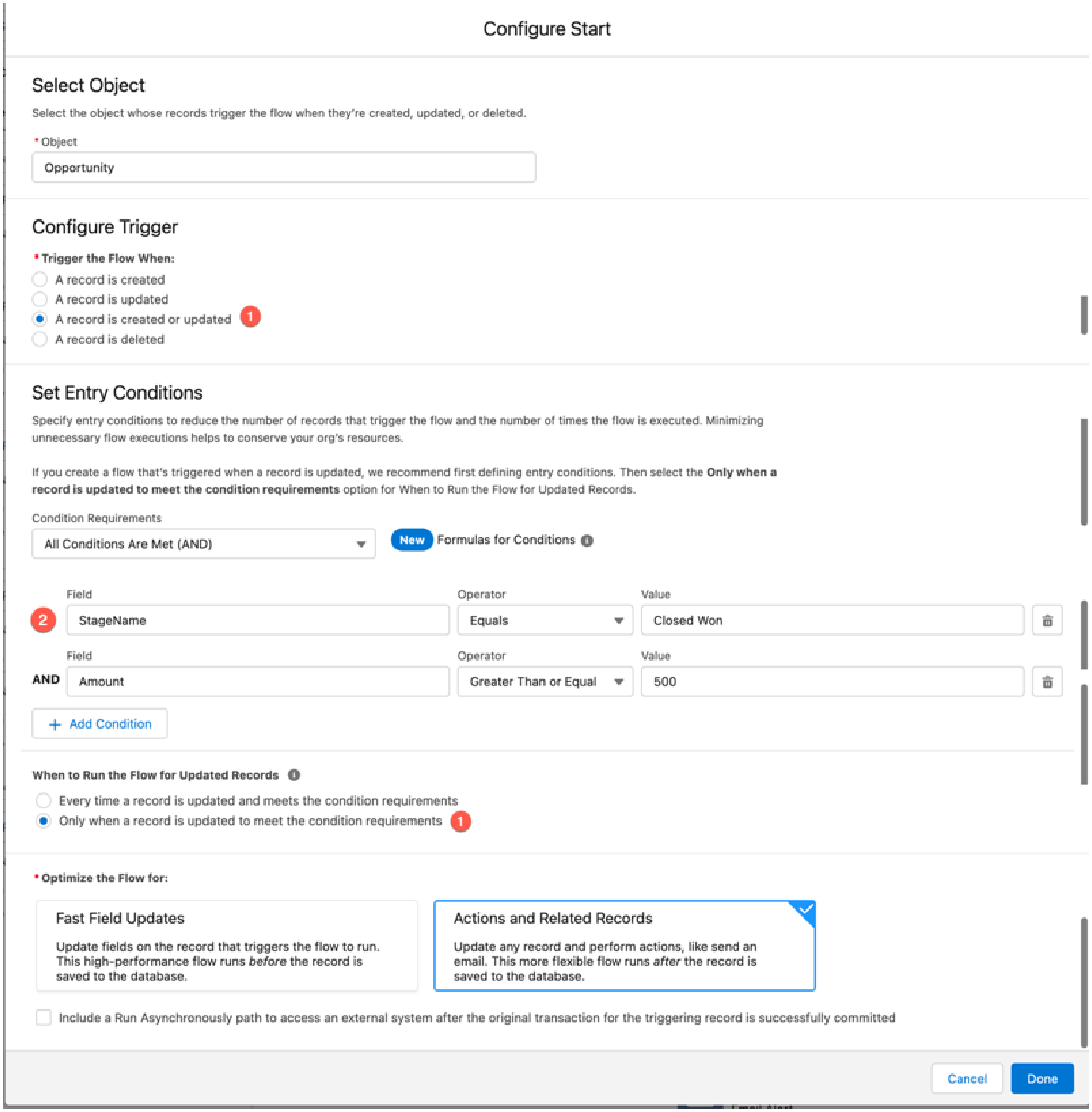
Rule Criteria

Run this rule if the criteria are met :

Field	Operator	Value	
Opportunity: Stage	equals	Closed Won	AND
Opportunity: Amount	greater or equal	500	AND
--None--	--None--		AND
--None--	--None--		AND
--None--	--None--		

Add Filter Logic...

Flow:



Configure Start

Select Object

Select the object whose records trigger the flow when they're created, updated, or deleted.

* Object: Opportunity

Configure Trigger

* Trigger the Flow When:

○ A record is created
○ A record is updated
● A record is created or updated
○ A record is deleted

Set Entry Conditions

Specify entry conditions to reduce the number of records that trigger the flow and the number of times the flow is executed. Minimizing unnecessary flow executions helps to conserve your org's resources.

If you create a flow that's triggered when a record is updated, we recommend first defining entry conditions. Then select the **Only when a record is updated to meet the condition requirements** option for When to Run the Flow for Updated Records.

Condition Requirements: All Conditions Are Met (AND)

New Formulas for Conditions

	Field	Operator	Value
	StageName	Equals	Closed Won
AND	Amount	Greater Than or Equal	500

+ Add Condition

When to Run the Flow for Updated Records

○ Every time a record is updated and meets the condition requirements
● Only when a record is updated to meet the condition requirements

* Optimize the Flow for:

Fast Field Updates
Update fields on the record that triggers the flow to run. This high-performance flow runs *before* the record is saved to the database.

Actions and Related Records ✓
Update any record and perform actions, like send an email. This more flexible flow runs *after* the record is saved to the database.

□ Include a Run Asynchronously path to access an external system after the original transaction for the triggering record is successfully committed

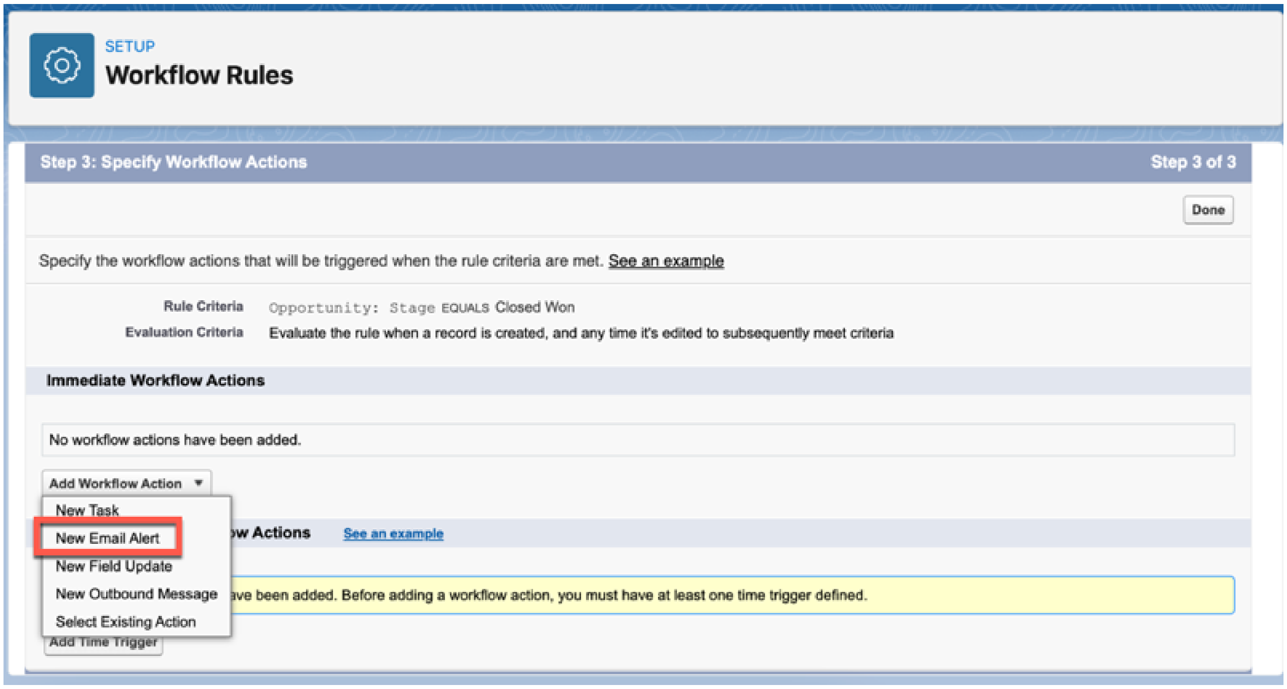
When to Run/Trigger (1)

Our example Workflow Rule uses created, and any time it's edited to subsequently meet criteria. For the flow, select "A record is created or updated under Configure Trigger". And select "Only when a record is updated and meets the condition requirements" for When to Run the Flow for Updated Records. You choose this option in the Set Entry Conditions.

Rule Criteria/Set Entry Conditions (2)

The criteria/conditions are similar in both the WFR and the flow. The Condition Requirements field is set to "conditions are met". The Field, Operator, and Value are almost identical. The field names are slightly different, as the Object isn't included in the Field Name in a flow.

Add an Action



SETUP

Workflow Rules

Step 3: Specify Workflow Actions

Step 3 of 3

Specify the workflow actions that will be triggered when the rule criteria are met. See an example

Rule Criteria: `Opportunity: Stage` EQUALS Closed Won

Evaluation Criteria: Evaluate the rule when a record is created, and any time it's edited to subsequently meet criteria

Immediate Workflow Actions

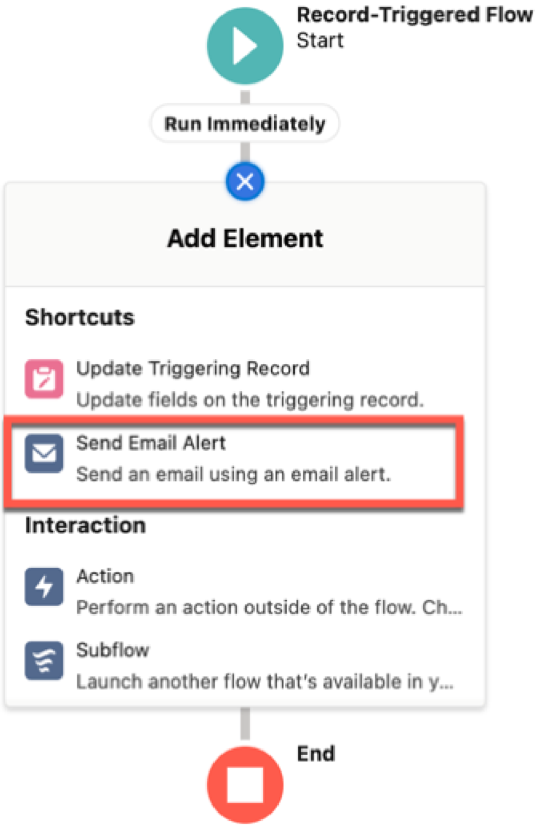
No workflow actions have been added.

ow Actions See an example

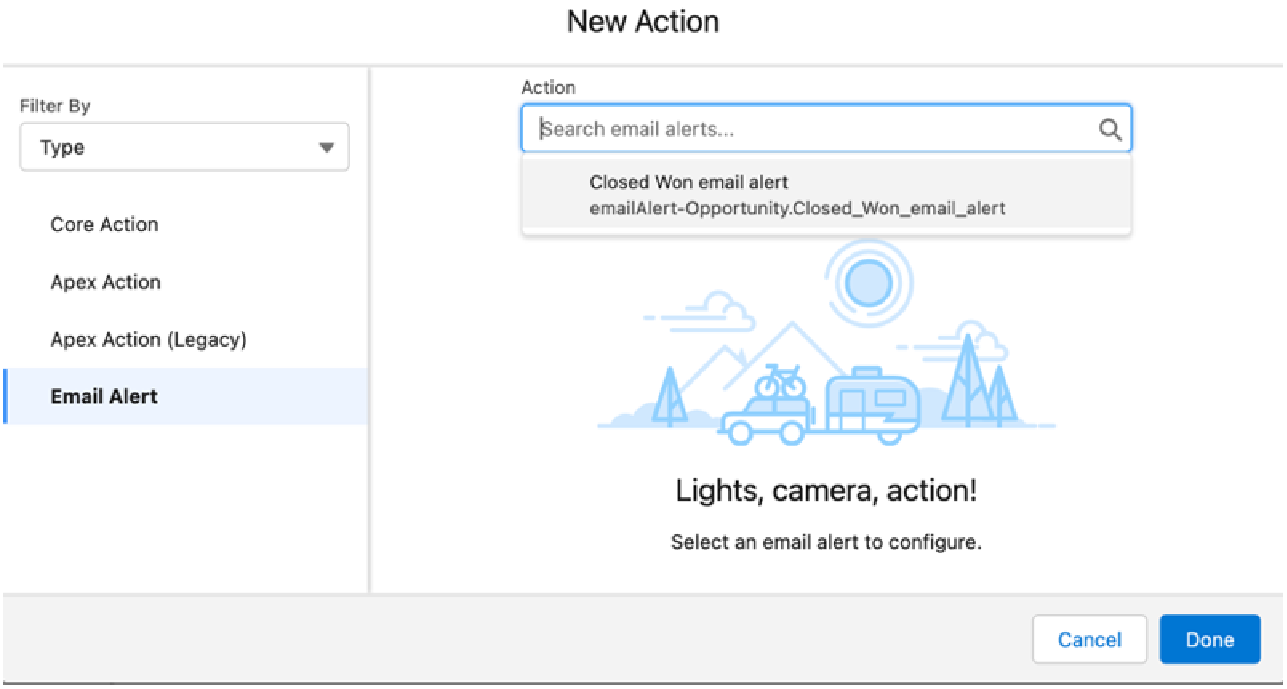
ave been added. Before adding a workflow action, you must have at least one time trigger defined.

When you build a workflow rule, the action is selected after the Rule Criteria is entered.

In Flow, there’s a Send Email Alert option from Add Element.


Record-Triggered Flow
Start
Run Immediately
Add Element
Shortcuts
Update Triggering Record
Update fields on the triggering record.
Send Email Alert
Send an email using an email alert.
Interaction
Action
Perform an action outside of the flow. Ch...
Subflow
Launch another flow that's available in y...
End

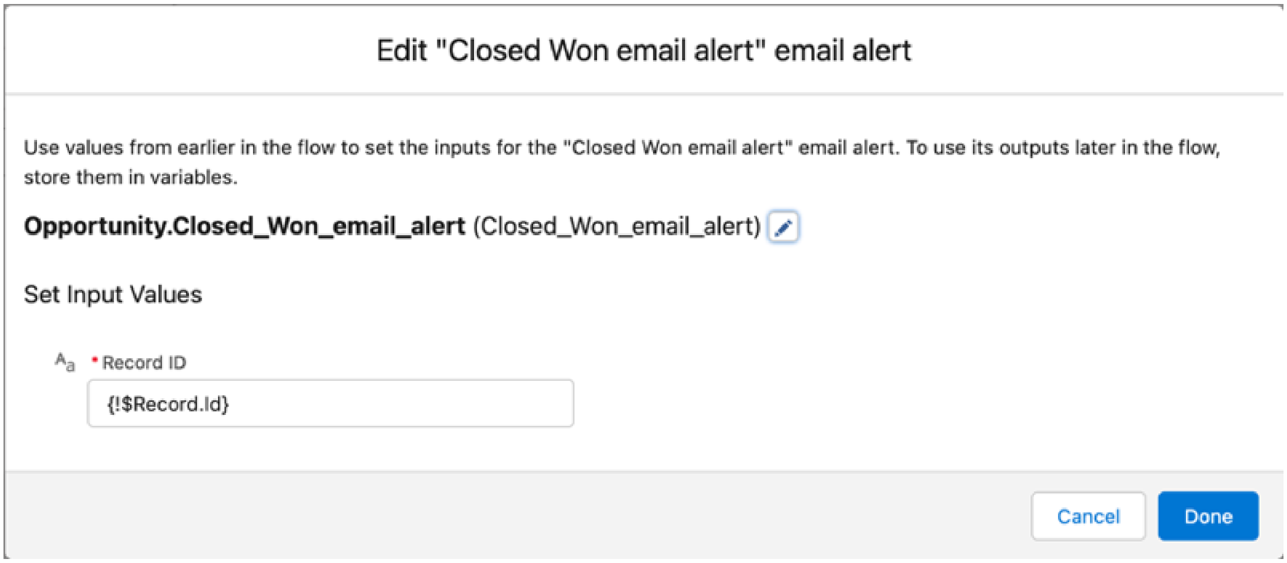
Select the email alert by clicking in the New Action window. Choose the alert to use from the list. You can use existing Email Alerts that you used previously in workflow rules. Configure Email Alerts to be used in flows just as you did for workflow rules. Email Alerts are configured under Workflow Actions in Process Automation.



Closed Won email alert
emailAlert-Opportunity.Closed_Won_email_alert

Select an email alert to configure.

In the flow, enter *$Record* into the Record ID field. This global variable contains the values from the record that triggers it to run. So, there's no need to add a Get Records element to obtain the record data. And, no flow variables have to be created to store the record data.

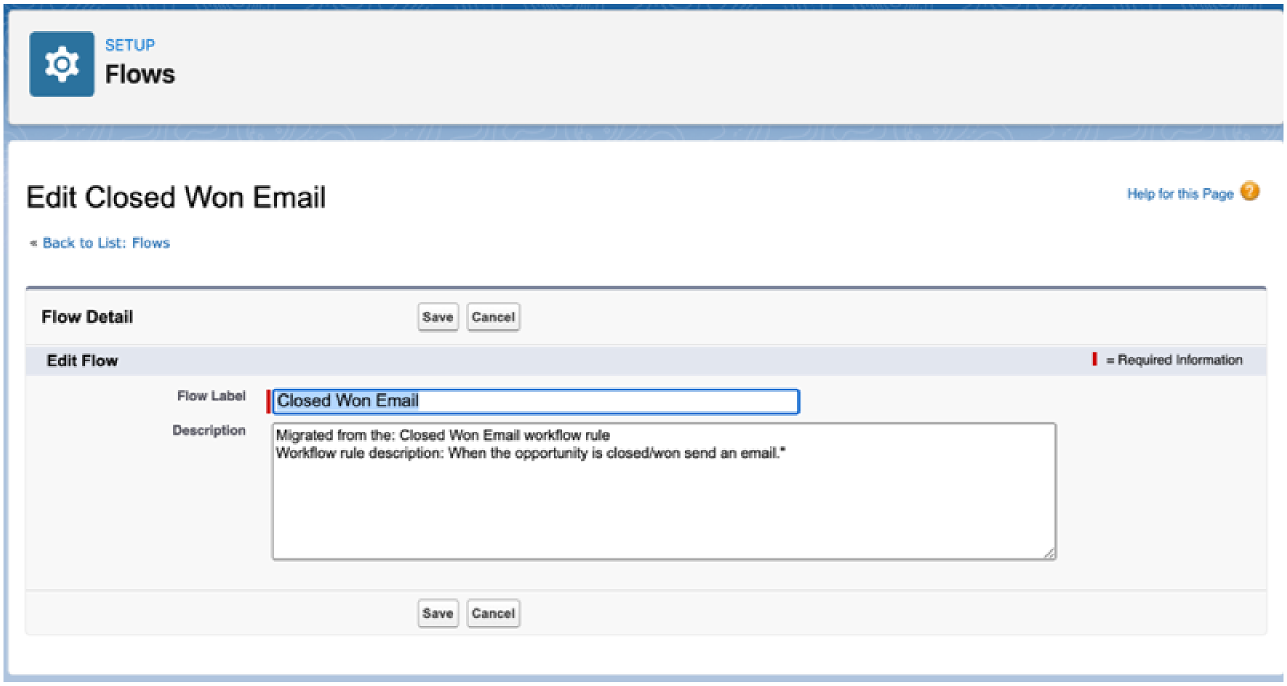


Use values from earlier in the flow to set the inputs for the "Closed Won email alert" email alert. To use its outputs later in the flow, store them in variables.

Opportunity.Closed_Won_email_alert (Closed_Won_email_alert)

* Record ID: {!$Record.Id}

If you have a workflow rule similar to this example, use the migration tool. The migration tool does a great job and even adds the workflow rule name and description in the details of the new flow.



Edit Closed Won Email

Edit Flow

Flow Label: Closed Won Email

Description: Migrated from the: Closed Won Email workflow rule
Workflow rule description: When the opportunity is closed/won send an email."

Process Builder

Many of the tasks you assign, the emails you send, and other record updates are vital parts of your standard processes. Instead of doing this repetitive work manually, you can configure flows or processes to do it automatically. We strongly recommend using Flow Builder, but Process Builder can also help you automate your business processes and give you a graphical representation as you build it.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Process Builder supports three types of processes for your automation needs. The type determines what triggers the process.

- A record change process starts when a record is created or updated.
- An event process starts when a platform event message is received.
- An invocable process starts when something else, like another process, invokes it.

Each process consists of:

- Criteria that determine when to execute an action group.
- Action groups, which consist of immediate or scheduled actions. Only record change processes support scheduled actions.

If you need an existing process to do more than what process actions allow, don't worry. You can also call a flow or Apex from a process.

Examples of Processes
See how Process Builder can make automating your business processes super easy.

Process Limits and Considerations
Before you start creating, managing, and activating processes, understand the limits and considerations.

Setting Values in the Process Builder

Throughout the Process Builder, you must set values, for example, to set conditions in a criteria node, to set the fields on a new case in a Create a Record action, or to specify an Apex method to reference.

Setting Advanced Options in the Process Builder

The Process Builder lets you choose some advanced options for executing actions in your processes.

Create a Process

To create a process, define its properties and which records it evaluates, and then add criteria nodes and actions.

Troubleshoot Processes

Use the error messages that appear in the Process Builder and the emails you receive when a process fails to help solve problems that arise when you're working with processes. When all else fails, look at the Apex debug logs for your processes.

SEE ALSO:

Choose Which Salesforce Flow Feature to Use

Examples of Processes

See how Process Builder can make automating your business processes super easy.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Sample Process: Opportunity Management

This example automates a single business process by using the Process Builder instead of workflow rules.

Sample Process: Printer Management

The example demonstrates how you can use Process Builder to subscribe to and evaluate a platform event.

Sample Process: Managing Documents

This example uses Process Builder to manage documents in Salesforce. The example moves a document to a shared folder in Quip whenever the record that the document is associated with is created or updated. This process ensures that the documents associated with a Salesforce record object are always available to users who have access to the shared folder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Sample Process: Opportunity Management

This example automates a single business process by using the Process Builder instead of workflow rules.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

The example demonstrates how you can use the Process Builder to automate a single process by adding multiple groups of criteria and then associating individual actions with those criteria. In addition, some actions are available with the Process Builder that you can't perform with workflow rules, such as creating records.

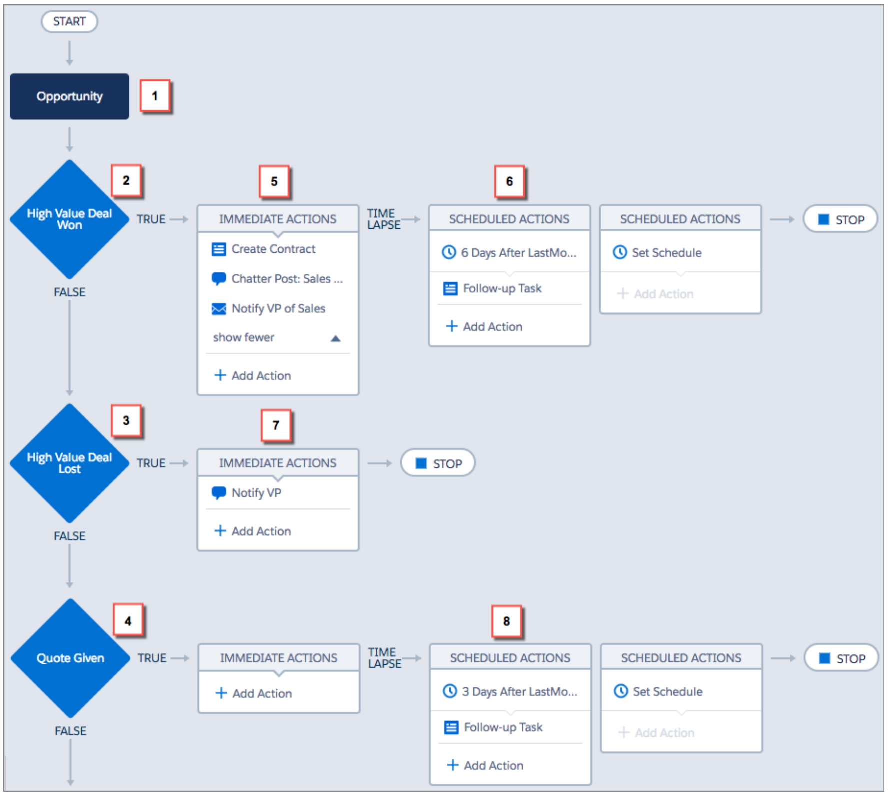
In this example, the process is defined to start when an opportunity record (1) is created or edited.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Three criteria nodes are then set up to check whether a high-value deal was won (2), a high-value deal was lost (3), or a quote was given (4). For the first criteria node that evaluates to true, the associated action group is executed.



The High Value Deal Won criteria checks whether the opportunity's stage is closed and won and also whether the opportunity's amount is greater than $1,000,000.00. If both of these conditions are met, the associated action group is executed. For this criteria node, three immediate actions (5) and one scheduled action (6) are defined.

These actions:

- Create a draft contract record that's associated with the opportunity's account.
- Congratulate the opportunity owner for closing and winning the opportunity by posting to the Sales Chatter group.
- Notify the VP of sales via email that the opportunity was closed and won.
- Create a high priority follow-up task for the associated account's owner, which is scheduled to execute six days after the opportunity's `Close Date`.

If the High Value Deal Won criteria conditions aren't met, the associated group of actions doesn't execute and the next criteria node (High Value Deal Lost) is evaluated.

The High Value Deal Lost criteria node checks whether the opportunity stage is closed and lost and whether the opportunity amount is greater than or equal to $1,000,000.00. If these conditions are true, we've set up an action (7) to notify the VP of sales by creating a chatter post on the opportunity record. The post identifies the opportunity and the opportunity amount that was lost.

If neither of the previous criteria conditions are met, the next criteria node defined in this process checks whether the opportunity stage is set to "Proposal/Quote Given." If this condition is true, a scheduled action (8) is executed three days after the record is updated. The scheduled action creates a follow-up task for the opportunity owner to call to inquire about the opportunity.

Using the Process Builder, we've combined three criteria nodes and associated actions into a single, automated process. To automate the same business process with workflow, you would have to create three different workflow rules and use Apex triggers to create the contract record and post to the Sales Chatter group.

Sample Process: Printer Management

The example demonstrates how you can use Process Builder to subscribe to and evaluate a platform event.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

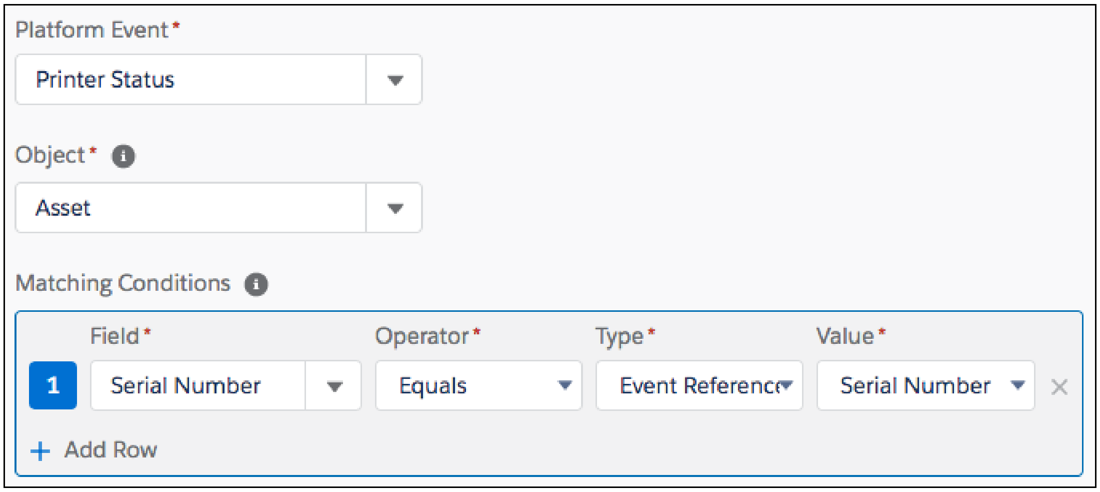
Using platform events and the Salesforce REST API, your printer publishes a Printer Status event at the end of each work day. This event includes the printer's serial number, how much ink and paper it has, and how many pages it has printed in total.

When Salesforce receives the Printer Status event, your Printer Management process uses the serial number to find a matching asset in your Salesforce org.

EDITIONS

Available in both Salesforce Classic and Lightning Experience

Available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions



Platform Event* Printer Status

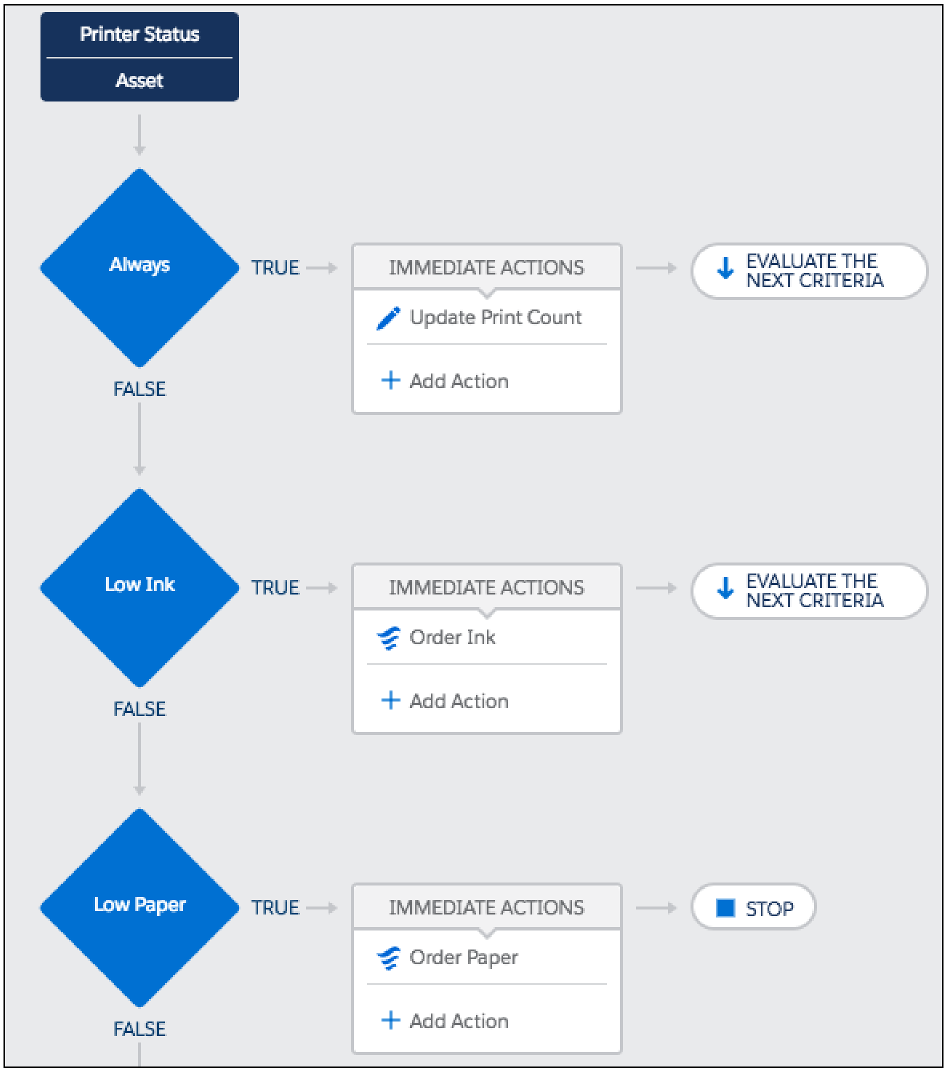
Object* Asset

Matching Conditions

	Field*	Operator*	Type*	Value*
1	Serial Number	Equals	Event Referenc	Serial Number

If the process finds a match, it evaluates the event's data.

- The first criteria always update the asset's print count.
- The second criteria checks if the ink is low. If so, it launches a flow that orders more ink and assigns a service technician to install the ink cartridge.
- The third criteria checks if the paper is low. If so, it launches a flow that orders more paper and assigns a service technician to add the paper.



SEE ALSO:

Platform Events Developer Guide: Considerations for Defining and Publishing Platform Events

Sample Process: Managing Documents

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

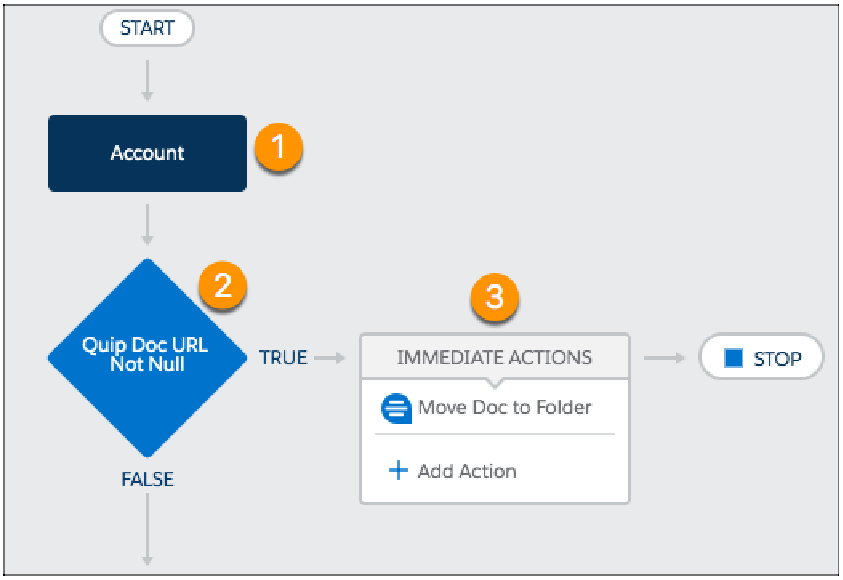
This example uses Process Builder to manage documents in Salesforce. The example moves a document to a shared folder in Quip whenever the record that the document is associated with is created or updated. This process ensures that the documents associated with a Salesforce record object are always available to users who have access to the shared folder.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

In the example, it's assumed that:

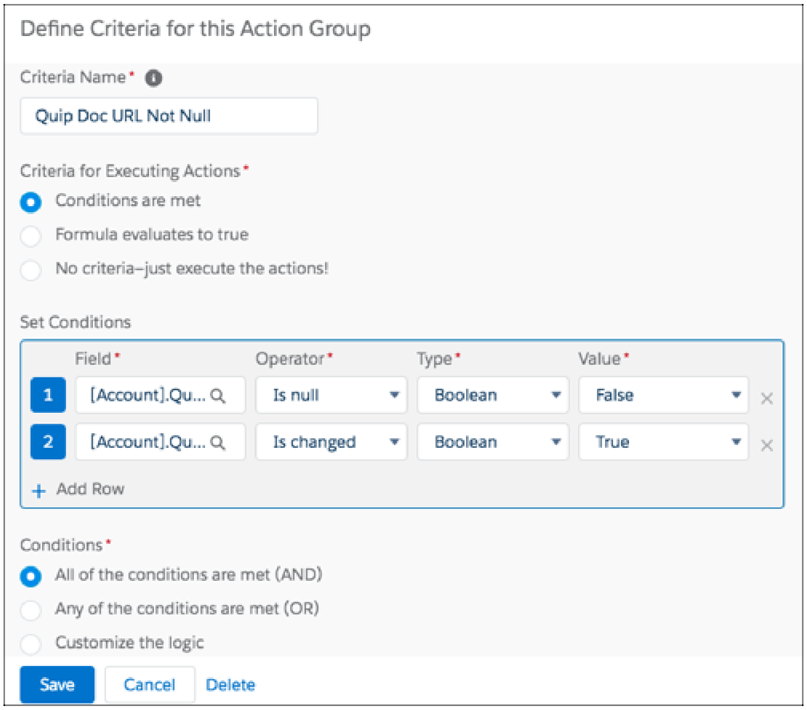
- The Account object has a custom field labeled Quip Account Plan Doc (API name Quip_Account_Plan_Doc__c). For each account, the custom field stores the URL of an associated document that contains a plan for the account.
- All the Account Plan documents are in a shared folder. The folder's URL is `https://acme.quip.com/123FakeURL456`.

The process starts when an Account record (1) is created or updated.



The criteria (2) checks whether the value of the Quip Account Plan Doc custom field has changed and whether the field isn't null. If both conditions are true, an immediate action (3) moves the Quip Account Plan doc to the specified shared folder. Let's dig a little deeper into the criteria and action in this sample process.

The criteria's Set Conditions section defines which conditions must be met in the Quip Account Plan Doc field to move a document. There are two conditions: the Quip Account Plan Doc field isn't null and that it's changed. The Field column requires the full API name of the field, in this case, `[Account].Quip_Account_Plan_Doc__c`. The Conditions section specifies that all the conditions must be met to execute the action.



Define Criteria for this Action Group

Criteria Name*: Quip Doc URL Not Null

Criteria for Executing Actions*

- ● Conditions are met
- ○ Formula evaluates to true
- ○ No criteria–just execute the actions!

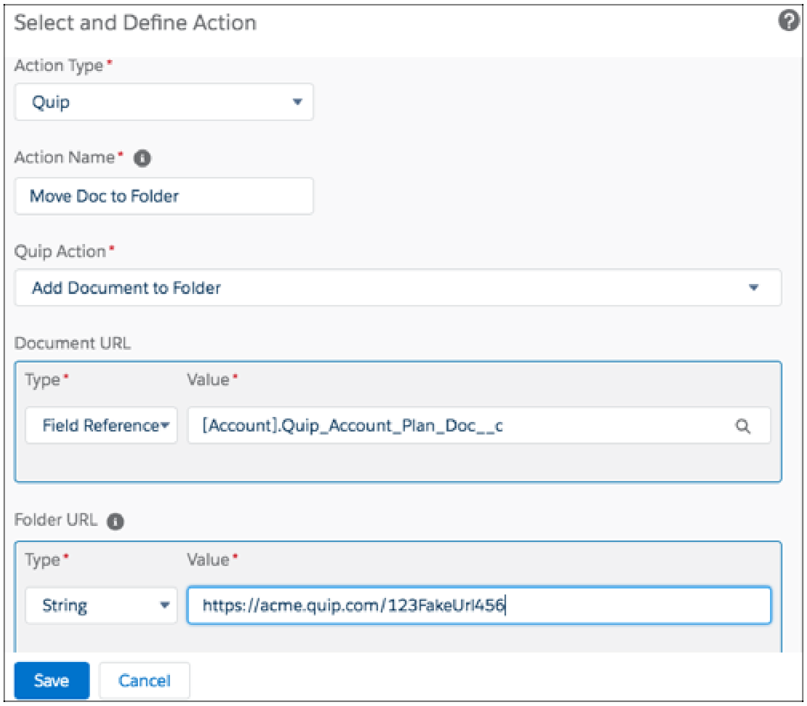
Set Conditions

	Field*	Operator*	Type*	Value*
1	[Account].Qu...	Is null	Boolean	False
2	[Account].Qu...	Is changed	Boolean	True

Conditions*

- ● All of the conditions are met (AND)
- ○ Any of the conditions are met (OR)
- ○ Customize the logic

For the action definition, you select **Quip** for Action Type to view the Quip-related actions. Enter an Action Name (`Move Doc to Folder` in our example) then select the action (**Add Document to Folder**). The Document URL is a field reference to the custom field ([Account].Quip_Account_Plan_Doc__c) that contains the URL of the document to move. The Folder URL is a String type that specifies the URL of the shared folder.



Select and Define Action

Action Type*: Quip

Action Name*: Move Doc to Folder

Quip Action*: Add Document to Folder

Document URL

Type*	Value*
Field Reference	[Account].Quip_Account_Plan_Doc__c

Folder URL

Type*	Value*
String	https://acme.quip.com/123FakeUrl456

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Process Limits and Considerations

Before you start creating, managing, and activating processes, understand the limits and considerations.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Process Limits

When building processes, keep shared limits and Apex governor limits in mind. In addition, a process's API name must be unique across all processes and flows in your Salesforce org.

Process Usage-Based Entitlements

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce contacts you to discuss additions to your contract. In the meantime, your processes continue to run as usual.

When Do Processes Evaluate Record Changes?

Processes start automatically and are invisible to the user. Before you design or activate a process, understand which changes trigger processes.

Considerations for Designing Processes

Before you design a process, understand the limitations and guidelines.

Considerations for Managing Processes

Understand what happens when you install, activate, or delete processes.

Considerations for Deploying Processes

Keep these considerations in mind when deploying processes, such as when using packages or change sets.

Considerations for Processes in Transactions

Each process runs in the context of a transaction. A transaction represents a set of operations that are executed as a single unit. When a process is triggered more than one time in a single transaction, Salesforce executes similar actions in one batch.

Process Limits

When building processes, keep shared limits and Apex governor limits in mind. In addition, a process's API name must be unique across all processes and flows in your Salesforce org.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Limits Shared with Other Features

Processes share some limits with rules and flows.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Per-Org Limit	Essentials or Professional Edition	Enterprise, Unlimited, Performance, or Developer Edition
Active record change processes and rules per object Rules include workflow rules, escalation rules, assignment rules, and auto-assignment rules.	50	50
Total processes	5 per process type	4,000 per process type
Active processes	5 per process type	2,000 per process type
Criteria nodes that are evaluated and actions that are executed at runtime per process	None[1]	None[1]
Groups of scheduled actions that are executed or flow interviews that are resumed per hour	1,000	1,000
Combined total of these automations that start or resume based on a record's field value. • Resume events that are defined in active flows • Groups of scheduled actions that are defined in active processes • Time triggers that are defined in active workflow rules • Inactive flow interviews that are resumed	20,000	20,000

The daily limit for emails sent from email alerts is 1,000 per standard Salesforce license per org, except for Developer Edition orgs. For a Developer Edition org, the daily workflow email limit is 15 per standard Salesforce license. The overall org limit is 2,000,000, which is shared across all features that use workflow email alerts: workflow rules, approval processes, flows, and processes.

[1] In API version 57.0, the limit of 2000 flow elements was removed. In API version 56.0 and earlier, flows could have a maximum of 2000 flow elements.

Apex Governors and Limits for Processes

Salesforce strictly enforces limits to ensure that runaway processes don't monopolize shared resources in the multitenant environment. Processes are governed by the per-transaction limits that are enforced by Apex. If a process launches other automation in the same transaction, that automation shares the process transaction's limits. If the process or its launched automation causes the transaction to exceed governor limits, the system rolls back the entire transaction. For details about the operations that are included in the transaction, see Triggers and Order of Execution in the *Apex Developer Guide*.

Description	Per-Transaction Limit
Total number of SOQL queries issued	100
Total number of records retrieved by SOQL queries	50,000
Total number of DML statements issued	150
Total number of records processed as a result of DML statements	10,000
Maximum CPU time on the Salesforce servers	10,000 milliseconds

Each Create a Record action uses one DML statement. Each Quick Action action uses one DML statement. Each Update Records action uses one SOQL query and one DML statement. Each Flows action can use multiple SOQL queries and DML statements, depending on the elements that the flow executes. For details, see Per-Transaction Flow Limits on page 246.

Limits for Creating and Managing Processes

Consider these limits when creating and managing processes.

Per-Process Limit	Value
Total characters in a process name	255
Total characters in a process's API name	79
Total versions of a process	50
Total criteria nodes in a process	200

Process Usage-Based Entitlements

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce contacts you to discuss additions to your contract. In the meantime, your processes continue to run as usual.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For per-month entitlements, your contract determines the start and end of the month. You can view the start and end dates for your org's usage-based entitlements on the Company Information page in Setup.

Note:

- When a process built in Process Builder launches a flow, both the process and the flow count toward your allocation of flow interviews.
- If you enable recursion for a process built in Process Builder, a separate flow interview starts each time the process evaluates a record. Each flow interview counts toward your allocation of flow interviews.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

This table describes the free allocations that are granted based on your org's edition.

Per-Org Usage-Based Entitlement	What's Counted	Essentials and Professional Editions	Performance and Developer Editions	Enterprise and Unlimited Editions
Maximum flow interviews without UI per month	Interviews of flow types that can't have screen elements, such as autolaunched flows, transaction security flows, and processes built in Process Builder	10,000,000	10,000,000,000	10,000,000,000

If you have questions about increasing your allocation, contact your Salesforce account executive.

SEE ALSO:

- Usage-Based Entitlements
- View Your Salesforce Org's Usage-Based Entitlements
- Reevaluate Records in the Process Builder
- Flow Types
- How Does Salesforce Process Scheduled Actions?

When Do Processes Evaluate Record Changes?

Processes start automatically and are invisible to the user. Before you design or activate a process, understand which changes trigger processes.

Important: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

A record change can apply to more than just processes. When a record is created or edited, Salesforce evaluates whether to run other setup items, such as validation rules on the record. Salesforce evaluates the setup items in this order. For more information, see Triggers and Order of Execution in the *Apex Developer Guide*.

- Validation rules
- Assignment rules
- Auto-response rules
- Workflow rules and processes (and their immediate actions)
- Escalation rules

When you create a process, you associate the process with exactly one object. You also specify whether to evaluate only created records or both created and edited records. When you activate a process, it starts when a record change meets those settings.

Processes evaluate record changes when:

- A record is saved or created. Processes that are created after records are saved don't evaluate those records retroactively.
- A standard object in a master-detail relationship is reparented.
- Users synchronize records that were changed while using Connect Offline.
- If the lead setting **Require Validation for Converted Leads** is enabled, leads are converted.
- Another process, workflow rule, or flow updates the record in the same save operation, if the process is configured to reevaluate records.

  Processes can reevaluate records up to five times in one save operation. In a batch update, processes reevaluate only changed records.

Note: A record change can trigger more than one process. You can't determine which process starts first.

Processes don't evaluate record changes when:

- Campaign statistic fields, such as individual campaign statistics or campaign hierarchy statistics, are updated.
- Picklist values are mass replaced.
- Address fields are mass updated.
- Divisions are mass updated.
- Territory assignments of accounts and opportunities are modified.
- Self-Service Portal, Customer Portal, or partner portal users are deactivated.
- State and country/territory data is converted with the Convert tool.
- Values for state and country/territory picklists are modified using `AddressSettings` in the Metadata API.

Considerations for Designing Processes

Before you design a process, understand the limitations and guidelines.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

We recommend that you use the most recent stable version of Google Chrome™.

Best Practices for Designing Processes

Before you design a process in Process Builder, understand the best practices.

Process Builder Accessibility Considerations

Process Builder is 508-compliant, with one exception. You can't close window dialogs with your keyboard.

Considerations for Event Processes

These considerations are specific to processes that start when a platform event message is received.

Compatibility Considerations for Processes

Before you design a process, understand how processes interact with other Salesforce features, like custom objects and fields.

Process Formula Limitations

Formulas that are used as conditions in a criteria node have some limitations. If a process contains an invalid formula, you can't save or activate the process.

Considerations for Scheduling Process Actions

Scheduled actions are supported only in record-change processes and event processes. The scheduled time depends on the type of schedule, whether the field changed, and whether the process was deactivated.

Considerations for Processes that Send Custom Notifications

Before you begin sending custom notifications, learn about important storage, recipient, and org limits.

Considerations for Processes That Post to Chatter

The Post to Chatter action doesn't support Experience Cloud sites, and there are some limitations around what you put in the message.

Considerations for Processes That Update Records

Understand what happens when you change a record owner, update the same field multiple times, or update currency fields in a multiple currency org.

SEE ALSO:

Best Practices for Designing Processes

Best Practices for Designing Processes

Before you design a process in Process Builder, understand the best practices.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Build in a test environment.

To test whether a process is working properly, you must activate it. Build and test your processes in a sandbox environment, so that you can identify any issues without affecting your production data.

For each object, use one automation tool.

If an object has one process, one Apex trigger, and three workflow rules, you can't reliably predict the results of a record change.

Tip: When you replace a workflow rule with a process, deactivate the workflow rule before you activate the process. Otherwise, you can get unexpected results, such as overwritten records or redundant email messages. This best practice also applies when you replace an Apex trigger with a process.

Have only one record-change process per object.

Each time a record is created or updated, all record-change processes for its object are evaluated. We recommend restricting your org to one record-change process per object. Here's why.

- Get a Consolidated View of Your Org's Automation for an Object

  With one consolidated record-change process for an object, you can see all the criteria that are evaluated each time that object's records are updated, as well as the actions that are performed when the criteria are met.

- Avoid Hitting Limits

  When you consolidate your processes for one object into one main process, you also consolidate the actions in those processes. With fewer actions, your org is less likely to hit limits, such as number of SOQL queries.

- Determine the Order of Operations

  If you create multiple record-change processes for an object, Salesforce can't guarantee the order in which those processes are evaluated. When you automate everything in a single process, you explicitly set the order. The first criteria node is evaluated first, the second criteria node is evaluated second, and so on.

Here are a couple features that can ease your path to one main process.

- ISNEW()—Some automation applies only to created records. The rest applies to created and edited records. How could you possibly combine all of them into one process? Meet the formula function that detects whether the record being evaluated was recently created: `ISNEW()`.

  To add a create-only automation to a process that starts when a record is created or edited, convert the associated criterion's conditions to a formula. Then add *`&&ISNEW()`* to your formula.

- Invocable processes—Just like a process can call flows, a process can call other processes. Invocable processes are modular processes that start only when another process tells them to.

  For example, several criteria nodes in your "Account" process each evaluate some conditions, including whether the account is high value. Move those criteria nodes, without the high-value conditions that they have in common, into a "Top Account" invocable process. Then configure your "Account" process to invoke the "Top Account" process if the account is high value.

Combine actions when possible.

The more actions that a process executes, the more likely your org is to reach limits, such as the number of DML statements or total CPU usage. Avoid creating multiple actions when a single action would do.

For example, a process updates an account's address. Instead of creating a different action to update each individual field, create one action to update all the address fields.

Build reusable actions.

Some process actions are always reusable: email alerts, quick actions, processes, flows, and Apex. But how do you reuse other types of actions in multiple criteria groups or multiple processes?

- To reuse a Create a Record action or an Update Records action, build a quick action. Quick actions can be used in processes, flows, and on record pages.
- To reuse other process actions, configure the actions in an invocable process. In the relevant criteria groups, add a Processes action to call the invocable process. Invocable processes can be used only in processes.

Watch out for actions that overwrite previous changes.

Avoid having or be careful when multiple action groups update the same field.

Avoid generating infinite loops.

For example, an Update Records action in Process1 triggers Process2, and a Create a Record action in Process2 triggers Process1. The looping causes your org to exceed its limits.

Make sure that immediate actions don't cancel scheduled actions.

Pending scheduled actions are canceled when the associated criteria are no longer true. Make sure that the later immediate actions in your process don't unintentionally cancel pending scheduled actions.

Test as many permutations of your process as you possibly can.

As with all customizations in Salesforce, it's important to test your work. Make sure that you test as many possibilities as you can think of before you deploy the process to your production org.

To access external data after changing Salesforce data, use scheduled actions.

If Salesforce creates, updates, or deletes data in your org and then accesses external data in the same transaction, an error occurs. In your processes, we recommend using a separate transaction to access data in an external system. To do so, end the prior transaction by adding a scheduled action. For a record-change process, don't use a field-based schedule.

For example, an event process starts when it receives a platform event message from the custom platform event, Order Status. If the order status is new, the process creates a contact and schedules an action to update the order status in the external system. The event process doesn't fail because the scheduled action creates a separate transaction to access the external system.

SEE ALSO:

- Considerations for Designing Processes
- Considerations for the ISNEW Function
- Transactions and Scheduled Actions

Process Builder Accessibility Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Process Builder is 508-compliant, with one exception. You can't close window dialogs with your keyboard.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Close UI Elements with the Esc key:

You can close window dialogs using the Esc key on your keyboard, but you can't close side panels.

Reorder Criteria Nodes:

Follow these steps to reorder criteria nodes with your keyboard.

1. Select a criteria node by pressing the Space key.
2. Change the order of a criteria node by using the Up and Down arrow keys.
3. Save your changes by pressing the Space key.
4. Cancel by pressing the Esc key.

Considerations for Event Processes

EDITIONS

Available in both Salesforce Classic and Lightning Experience

Available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions

These considerations are specific to processes that start when a platform event message is received.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Supported Platform Events

Processes can subscribe to custom platform events and these standard platform events.

- AIPredictionEvent
- BatchApexErrorEvent
- FlowExecutionErrorEvent
- FOStatusChangedEvent

- OrderSummaryCreatedEvent
- OrderSumStatusChangedEvent
- PlatformStatusAlertEvent

Apex Actions

You can't use an event reference to set an sObject variable in the Apex class.

Email Alerts Actions

Email alerts can't use values from platform event messages. For the process to send an email that contains values from the platform event message that starts the process, use this workaround.

Flows Actions

You can't use an event reference to set a record variable in the flow, even when the platform event is specified as the record variable's object. To pass values into the flow from the platform event message that starts the process, use this workaround.

- In the flow, create a variable for each field in the platform event. Be sure to use compatible data types and make the variables available for input.
- In the process, when you add the Flows action, use event references to assign each platform event field to its corresponding flow variable.

Associating with a Record

Use the process's matching conditions to find exactly one record. If the process can't find one record based on your matching conditions—because either it found multiple records or no records—the creator of the process receives an error email. If an error occurs, adjust the conditions in the process's trigger to be more specific.

Publishing Event Messages

With event processes, we don't block you from publishing the same event message that starts the process. To avoid creating an endless loop, make sure that the new event message's field values don't meet the filter criteria for the associated criteria node.

If a platform event is configured to publish immediately, the process publishes each event message outside of the database transaction. If the transaction fails and is rolled back, the event message is still published and can't be rolled back. So if you see an informational message under the selected platform event, consider whether you want the process to publish an event message only after the transaction commits successfully.

Subscriptions Related List

On the platform event's detail page, the Subscriptions related list shows which entities are waiting to receive that platform event's messages. The related list includes a link to each subscribed process. If flow interviews are waiting for that platform event's messages, one "Process" subscriber appears in the Subscriptions related list.

Packaging

When you package an event process, the associated object isn't automatically included. Advise your subscribers to create the object or manually add the object to your package. For example, when you package an event process that's associated with the Participants custom object, manually add the object to your package.

Uninstalling Events

Before you uninstall a package that includes a platform event, deactivate all processes that reference the platform event.

Einstein Predictions

A prediction event is sent for each Einstein prediction result, so use process matching conditions if you want your process to be triggered only by predictions on a specific object. For example, if your process acts only on predictions written to Lead records, add a matching condition to check that the Lead ID field equals the AI Predicted Object ID event reference.

If your process updates a field that is used by an Einstein prediction, Einstein runs the prediction again and writes back the new results. The new results generate a new prediction event that could trigger your process again, resulting in a loop. To avoid creating a process loop, only update fields that aren't used in Einstein predictions.

SEE ALSO:

Platform Events Developer Guide: Decoupled Publishing and Subscription

Platform Events Developer Guide: Platform Event Fields

Platform Events Developer Guide: Subscribe to Platform Event Messages with Processes

Compatibility Considerations for Processes

Before you design a process, understand how processes interact with other Salesforce features, like custom objects and fields.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Objects

Process Builder doesn't support:

- Deprecated custom objects
- Signup Request—unsupported in schedules only
- Social Post
- Social Persona

External Objects

- External objects aren't supported in record-change processes.
- When you create or update external object records, don't set values for indirect lookup relationships that map to a different data type on the external system. For example, don't set a value for a Text indirect lookup relationship that maps to a Date value on the external system.

Custom Fields

- Process Builder doesn't support custom fields of type File.
- If a process references a custom field:
  - You can't delete the field.
  - If you change the field type or name, the process breaks.

  - If you change the field label, the process doesn't break. But it still uses the original label.

Derived Fields

Process Builder doesn't support fields whose values are derived from other fields. Examples of derived fields include `Contact.Name`, `User.MediumPhotoUrl`, and `EmailMessage.Name`.

Polymorphic Fields

Queue labels aren't supported in process criteria. For example, you can't use `[Lead].Owner:Queue.Name` in process criteria. Instead, use `[Lead].Owner:Queue.DeveloperName` to reference the queue's API name.

Validation Rules

- Scheduled Update Records actions skip validation rules.
- Immediate Update Records actions obey validation rules.

Shield Platform Encryption

You can't use an encrypted field as a filter in an Update Records action.

Duplicate Rules

If a duplicate is found when a process tries to create or update a record, the process fails.

Converted Leads

To evaluate records that result from converted leads, enable the lead setting **Require Validation for Converted Leads**.

Formula Field Values

If a standard formula field references a field on a related object, that field's value is always null when a process starts. This limitation doesn't apply to custom formula fields that reference a field on a related object. For a custom formula field that uses the same formula, the field's value is derived when a process starts.

For example, the RevenueShare field on Campaign Influence calculates `CampaignInfluence.Opportunity.Amount * CampaignInfluence.Influence`. Because the formula references a field on Opportunity (a related object), the field's value is null.

Platform Cache

When a process contains a scheduled action, make sure that later actions in the process don't invoke Apex code that stores or retrieves values from the session cache. The session-cache restriction applies to Apex actions and to changes that the process makes to the database that cause Apex triggers to fire.

Process Formula Limitations

Formulas that are used as conditions in a criteria node have some limitations. If a process contains an invalid formula, you can't save or activate the process.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

All formulas that are used in a criteria node must:

- Return `true` or `false`. If the formula returns `true`, the associated actions are executed.
- Not contain more than 3,000 characters.
- Not contain an unsupported function.
- Reference the process trigger object for that process.
- Use the correct capitalization when referring to the process trigger object.

Tip: Parentheses aren't included automatically when you insert a function. Be sure to add parentheses, such as *`TODAY()`*, when building a formula.

Unsupported Functions

If a formula in a process uses any of the following functions, the formula returns `null`.

- GETRECORDIDS
- IMAGE
- INCLUDE
- PARENTGROUPVAL
- PREVGROUPVAL
- REQUIRE SCRIPT
- VLOOKUP

For a complete list of operators and functions for building formulas in Salesforce, see Formula Operators and Functions by Context.

Note:

- If your process criteria uses a formula, don't create a formula that always evaluates to true, such as *`2 < 5`*.
- ISCHANGED is available as both a formula function and as an operator. When it's used as a formula function in process criteria, you can't reference a child record's related fields. For example, ISCHANGED isn't supported when referencing a *`[Case].Contact.AccountId`* field, but it can be used when referencing *`[Case].ContactId`*.

SEE ALSO:

Tips for Working with Picklist and Multi-Select Picklist Formula Fields

Process Builder Advanced Option Considerations

Tips for Working with Picklist and Multi-Select Picklist Formula Fields

Custom Metadata Types and Process Builder

Considerations for Scheduling Process Actions

Scheduled actions are supported only in record-change processes and event processes. The scheduled time depends on the type of schedule, whether the field changed, and whether the process was deactivated.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Process Schedule Limitations
Before you add a schedule to a process, understand the limits and what isn't supported.

How Does Salesforce Process Scheduled Actions?
Understand the expected behavior for processing scheduled actions based on which type of schedule they're associated with, whether the field changed, and whether the process was deactivated.

Transactions and Scheduled Actions
Immediate actions in processes are executed in the same transaction as the operation that triggered the process, such as when a user creates or edits a record. Scheduled actions are included in a separate transaction.

Process Schedule Limitations

Before you add a schedule to a process, understand the limits and what isn't supported.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

- If an action group contains scheduled actions, you can't continue evaluating the next criteria in your process after executing those actions.
- SignupRequest processes don't support scheduled actions.
- Field-based schedules can't reference a Date or Date/Time field that contains automatically derived functions, such as TODAY or NOW.
- Field-based schedules can't reference a formula field that includes related-object merge fields.
- If you add a schedule for 0 Days Before a date, when you later reopen the process, the schedule changes to 0 Days After the date. The process still executes at the specified time.

How Does Salesforce Process Scheduled Actions?

Understand the expected behavior for processing scheduled actions based on which type of schedule they're associated with, whether the field changed, and whether the process was deactivated.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Limits for Processing Scheduled Actions

- An org can process up to 1,000 groups of scheduled actions per hour.

Each group of scheduled actions is associated with a schedule, such as "3 days from now." When a schedule is processed, the associated actions are executed. If an org exceeds this limit, Salesforce processes the remaining schedules in the next hour.

For example, an org has 1,200 groups of pending actions scheduled to be processed between 4:00 and 5:00 PM. Salesforce processes 1,000 groups between 4:00 and 5:00 PM, and it processes the remaining 200 groups between 5:00 and 6:00 PM.

Schedules Based on the Current Time

For example: 3 days from now.

The schedule is evaluated based on the time zone of the user who created the process.

Schedules Based on a Field Value

For example: 3 days after a case's `Created Date`.

Field-based schedules behave differently for record-change processes than they do for event processes.

All Processes:

These considerations apply to both record-change processes and event processes.

- If a schedule evaluates to a time in the past, Salesforce executes the associated actions as soon as possible. Depending on how many actions Salesforce is processing at the time, actions are executed within 1 hour.

  For example, a process emails an opportunity owner 7 days before the close date. The process starts for an opportunity with the close date set to today, so Salesforce executes the scheduled action as soon as possible.

- If you set a schedule to **0 Days After** a date, Salesforce executes the associated actions as soon as possible after the time represented by the date field. Depending on how many actions Salesforce is processing at the time, actions are executed within 1 hour.
- If the field referenced by a schedule has a null value, Salesforce ignores the schedule and the associated actions aren't executed.
- When a process schedules an action, Salesforce creates a flow interview record and pauses the interview until the scheduled time occurs. If the paused flow interview is deleted, Salesforce doesn't resume the paused flow interview, and the scheduled action isn't executed.

Record-Change Processes:

These considerations apply only to record-change processes.

When a record-change process executes a field-based schedule, Salesforce uses the field's current value. If the value is a date/time field, Salesforce uses the time zone of the user who created the process. If the value is a date field, Salesforce uses the org's time zone.

What Happens When the Field Value Changes?

- For processes that start when a record is created or edited:
  - Actions remain scheduled only as long as the criteria for the actions are still valid. If a record no longer matches the criteria, Salesforce cancels the scheduled actions for the record.
  - If the referenced field value changes, and the schedule hasn't been processed, Salesforce recalculates the scheduled time for the actions using the updated field value.

    For example, a process emails an opportunity owner 7 days before the opportunity close date. The close date is set to 2/20/20XX, and Salesforce schedules the email to be sent on 2/13/20XX. Before the email is sent, the close date is updated to 2/10/20XX. Salesforce recalculates the scheduled time and schedules the email to be sent on 2/3/20XX.

- For processes that start when a record is created, Salesforce never reevaluates the record associated with that process. The scheduled time for the actions stays the same, even if the record no longer meets the associated criteria when the scheduled actions are executed.

- If the record or object that the schedule is associated with is deleted, Salesforce cancels the scheduled actions for the record.

Limitations for Converted Leads:

- You can't convert a lead when an unexecuted schedule is based on one of the lead's fields.
- When **Validation and Triggers from Lead Convert** is enabled, scheduled actions on leads aren't executed during lead conversion.
- If a lead is converted into a campaign member before the associated scheduled actions finish, Salesforce still executes the scheduled actions.

Event Processes:

These considerations apply only to event processes.

- When an event process executes a field-based schedule, Salesforce uses the field's current value in the time zone of the user who created the process.
- The scheduled time for the actions stays the same, even if the field value changes, the associated record or object is deleted, or the record no longer meets the associated criteria.
- If the criteria are met when the process starts, Salesforce executes the scheduled actions.

What Happens When the Associated Process Is Deactivated?

After you deactivate a process, the scheduled time for pending scheduled actions stays the same. If a deactivated process has pending scheduled actions and the record whose field the schedule is based on is changed, Salesforce recalculates the schedule for those actions.

After a process is deactivated, Salesforce ignores all other changes to the associated records. Scheduled actions remain queued and continue to be processed on time unless the schedule is recalculated.

What Happens When Scheduled Actions Fail?

If a scheduled action fails—for example, because the user who caused the process to start is inactive—the admin who created the process receives an email with details about the failure. Salesforce makes additional attempts to execute a failed scheduled action before canceling it.

Transactions and Scheduled Actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Immediate actions in processes are executed in the same transaction as the operation that triggered the process, such as when a user creates or edits a record. Scheduled actions are included in a separate transaction.

> Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Scheduled actions aren't performed independently. They're batched in one transaction with other actions that are scheduled to execute at the same time, have the same process version ID, and are executed by the same user ID. This behavior can cause you to exceed your Apex governor limits if the batch's actions execute DML operations or SOQL queries.

A DML operation is used each time a Salesforce record is created, updated, or deleted, such as when a process executes a Create a Record action. A SOQL query is used each time Salesforce looks up information about an existing record, such as when a process executes an Update Records action. For details on Apex governor limits, see Process Limits on page 908.

To improve performance further and help avoid Apex governor limits, design scheduled actions to take advantage of bulkification.

Tip: Design a process with a scheduled action so that it doesn't execute mixed DML operations. A single transaction can't mix DML operations on data objects (such as Account), Setup objects (such as User Role), and external objects. For example, you can't update an account and a user role in a single transaction.

If a process has more than one group of scheduled actions and a group fails to execute in a batch:

- Prior groups of scheduled actions in that batch's transaction are successful.
- The immediate actions for that process are successful.
- All scheduled actions in that group aren't executed.
- Each remaining group of scheduled actions in that batch is executed in a separate transaction.

Example: Salesforce processes two groups of scheduled actions in the same batch. In the first group, one action fails, so the process fails. Subsequent actions within the first group aren't tried. Salesforce processes the second group in a separate transaction.

Note: The execution time is now in the past, so Salesforce executes the second group of scheduled actions within 1 hour.

Considerations for Processes that Send Custom Notifications

Before you begin sending custom notifications, learn about important storage, recipient, and org limits.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

- You can create up to 500 custom notification types.
- Each notification can have up to 10,000 users as recipients. However, you can add an action to the same process within Process Builder or to the same flow in Flow Builder to have more recipients.
- Your org saves your most recent 1 million custom notifications for view in notification trays. Your org can save up to 1.2 million custom notifications, but it trims the amount to the most recent 1 million notifications when you reach the 1.2 million limit.
- An org can execute up to 10,000 notification actions per hour. When you exceed this limit, no more notifications are sent in that hour, and all unsent notifications are lost. Notification actions resume in the next hour.

  For example, your notification action processes are triggered 10,250 times between 4:00 and 4:59. Salesforce executes the first 10,000 of those actions. The remaining 250 notifications aren't sent and are lost. Salesforce begins executing notification actions again at 5:00.
- When you send a custom notification from a process, the Target ID for the notification is the record that started the process. However, target records that don't have their own detail page (for example, a case comment, which appears only in a Case Comment related list) don't support direct navigation. Use Flow Builder to send the notification from a flow and specify either a different Target ID or Target Page Reference.

  Tip: To see how to specify the target using JSON, see pageReference.
- Custom notification title and body fields support plain text only.
- The content of custom push notifications depends on the Display full content push notifications setting. If full content push notifications aren't enabled, only the notification title is sent.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Considerations for Processes That Post to Chatter

The Post to Chatter action doesn't support Experience Cloud sites, and there are some limitations around what you put in the message.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Unsupported Feeds

Processes can't post to an Experience Cloud site user or group.

Chatter Message

- You can add up to 25 @mentions to a Chatter message.
- If you use Microsoft® Internet Explorer® version 11, you can't paste text into a message. Copy and paste actions are allowed in all other supported browsers.
- Don't start the message with a field reference, such as `{![Account].Name}`. Otherwise, the action fails to save. To work around this issue, add a space at the beginning of the message.

Deploying Processes That Post to Chatter

If your process posts to the Chatter feed of a specific user or group, the process runs only in the source org. The IDs referenced by the Post to Chatter action don't exist in the target org.

Considerations for Processes That Update Records

Understand what happens when you change a record owner, update the same field multiple times, or update currency fields in a multiple currency org.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Updating a Record's Owner

Processes that update owners don't automatically transfer the associated items. Use an Update Records action for each type of child record that you want to transfer.

For example, you want to transfer an account to a new owner. Add four Update Records actions to your process. The first updates the account. The second updates the child contacts. The third updates the child opportunities. And the fourth updates the child contracts.

Multiple Updates to the Same Field

If multiple Update Records actions apply different values to the same field, the last action's value is used.

Multiple Currencies

If your org uses multiple currencies, the currency fields are updated using the record's currency code. If you use a formula to update a field, the formula values use the record's currency code.

Inactive Users

Processes can't update records that inactive users own. When you deactivate a user, also transfer that user's records to an active user to avoid failed processes.

SEE ALSO:

Transferring Records

Considerations for Managing Processes

Understand what happens when you install, activate, or delete processes.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Installed Processes

If you install a non-template process from a managed package, you can only activate or deactivate it. If the process is a template, you can view and clone it, and you can edit the clone.

Active Processes

After you activate a process, you can no longer edit it.

Deleting Processes

You can't delete a process when it has unexecuted groups of scheduled actions. The workaround is to delete the unexecuted groups of unscheduled actions on page 980.

Considerations for Deploying Processes

Keep these considerations in mind when deploying processes, such as when using packages or change sets.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Change sets are available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Packages are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Processes That Reference Other Components

If you deploy a process that contains any of the following actions, the corresponding components aren't included in the package or change set automatically. To deploy successfully, manually add the referenced components to the package or change set.

- Apex
- Email Alerts
- Launch a Flow
- Post to Chatter
- Quick Actions

- Submit for Approval

Templates

If you install a process template from a package, make sure the process is inactive unless you want it to actively run in your org.

If you add a process template to a package, first deactivate the process unless you're sure that installers want that process to be active in their orgs.

Deploying Processes That Post to Chatter

If your process posts to the Chatter feed of a specific user or group, the process runs only in the source org. The IDs referenced by the Post to Chatter action don't exist in the target org.

SEE ALSO:

Deploy Processes and Flows as Active

Considerations for Processes in Transactions

Each process runs in the context of a transaction. A transaction represents a set of operations that are executed as a single unit. When a process is triggered more than one time in a single transaction, Salesforce executes similar actions in one batch.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

By default, if one process in a batch fails, it causes all the processes in the batch to fail, and the transaction rolls back. If one process in a batch fails while executing one of these actions, Salesforce attempts to save all successful record changes in the batch up to three times.

- Create a Record
- Flows (Create Records and Update Records elements only)
- Processes (Create a Record and Update Records actions only)
- Update Records

Example: When you upload 100 cases, the flow MyProcess_2 triggers one process for each case.

- 50 processes stop at Create a Record action Create_Task_1.
- The other 50 processes stop at Create a Record action Create_Task_2.

The result? At least two groups of bulk operations to execute.

- One for the 50 processes that execute Create_Task_1
- One for the 50 processes that execute Create_Task_2

Setting Values in the Process Builder

Throughout the Process Builder, you must set values, for example, to set conditions in a criteria node, to set the fields on a new case in a Create a Record action, or to specify an Apex method to reference.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Field Picker
Use the field picker to reference fields on the record that started the process or fields on related records.

Process Builder Value Types
When setting a value for a given field—whether on the record that started the process or a related record— the available value types are filtered based on the field that you've selected.

Multi-Select Picklists in the Process Builder
The Process Builder lets you select multiple values for a multi-select picklist field.

Field Picker

Use the field picker to reference fields on the record that started the process or fields on related records.

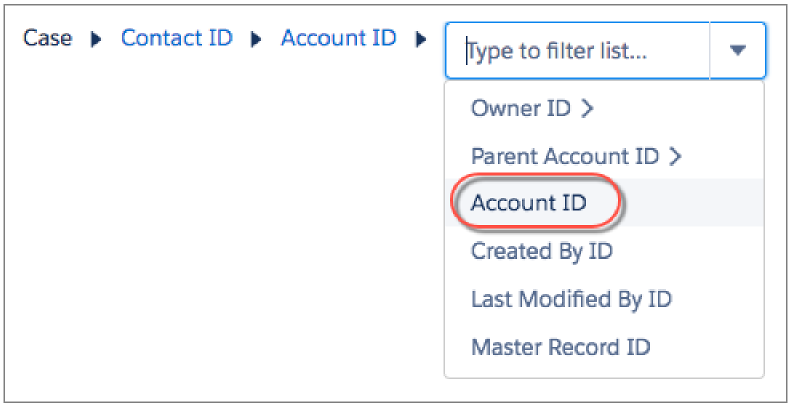
EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

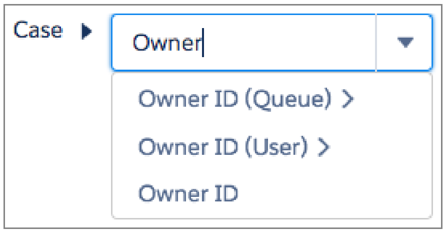
Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To use fields on a related record, click a field with > next to the value. For example, use the Account ID field value on the case's contact account.



The field picker displays only the fields that are compatible with the selected parameter.

If you see a field multiple times, it means that the field can relate to multiple objects. For example, if you created a queue for cases, a case's owner can be either a user or a queue. `Owner ID` is a *polymorphic field*—a field that relates to more than one object.



To access a field on the case's owner, choose the appropriate relationship. If you select **Owner ID (User)** and the owner of the record is a queue, the process fails.

Note: Queue labels aren't supported in process criteria. For example, you can't use `[Lead].Owner:Queue.Name` in process criteria. Instead, use `[Lead].Owner:Queue.DeveloperName` to reference the queue's API name.

Process Builder Value Types

When setting a value for a given field—whether on the record that started the process or a related record— the available value types are filtered based on the field that you've selected.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

The available value types are:

- Currency—Manually enter a currency value.
- Boolean—Choose a true/false boolean value.
- Date/Time or Date—Manually enter a date/time or date value.
- Formula—Create a formula expression.
- Global Constant—Choose a global constant to set a value to null or an empty string—for example, choose $GlobalConstant.Null.

  Note: These global constant values aren't supported with the `is null` operator.

  - `$GlobalConstant.Null`
  - `$GlobalConstant.EmptyString`

- ID—Manually enter a Salesforce ID value, for example, *`00300000003T2PGAA0`*.

  Note: If your process is based on a user ID (for example, when an [Event].OwnerId equals a specific ID value) make sure that the ID value is an 18-character ID and not a 15-character ID. You can convert a 15-character ID to 18 characters at www.adminbooster.com/tool/15to18.

- Multi-Picklist—Choose one or more multi-select picklist values.
- Number—Manually enter a number value.
- Picklist—Choose a picklist value.
- Queue—Search for a specific queue in your org.
- Reference—Choose a field on the record or on a related record.
- String—Manually enter a string value.

- User—Search for a specific user in your org.

Multi-Select Picklists in the Process Builder

The Process Builder lets you select multiple values for a multi-select picklist field.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For example, set multiple values for the `Country` field for a company record that operates in Ireland, the UK, and France.

You can use multi-select picklists in:

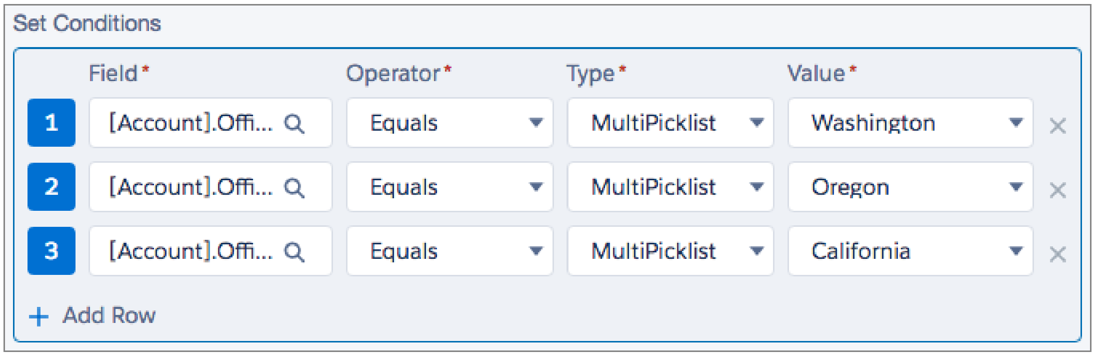
- Formulas
- Process criteria
- Create a Record actions
- Quick Actions
- Update Records actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

In process criteria, set multiple values by creating one condition for each individual multi-select picklist value. For example, if your process checks whether changes were made to an account's office locations, reference multiple values by choosing the same field for each multi-select picklist value.



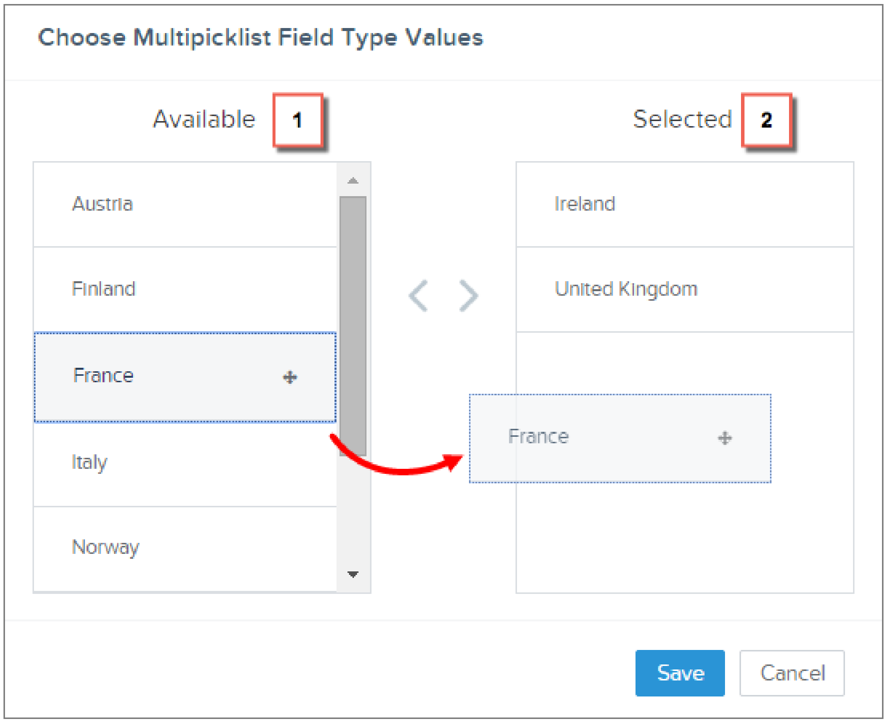
Set Conditions

	Field*	Operator*	Type*	Value*
1	[Account].Offi...	Equals	MultiPicklist	Washington
2	[Account].Offi...	Equals	MultiPicklist	Oregon
3	[Account].Offi...	Equals	MultiPicklist	California

Refer to Tips for Working with Picklist and Multi-Select Picklist Formula Fields for more information.

When you reference a multi-select picklist field in an action, enter values by clicking **Choose values…**

Add or remove values by dragging them between the Available (1) and Selected (2) columns.



Choose Multipicklist Field Type Values

Keep these considerations in mind when using operators with multi-select picklists.

- If you select only one value from a multi-select picklist field, you can use the Equals operator.
- If you use the Equals operator with multiple multi-select picklist values and choose the **Any of the conditions are met (OR)** option, the condition matches on one value only. For example, if your process checks whether a Region field equals West or East, the condition evaluates to true when the value is West or when the value is East, but doesn't evaluate to true when both West and East are selected values.
- If you use **Contains** and **OR** to evaluate multiple multi-select picklist values, the condition evaluates to true on multiple values. For example, if your process checks whether a `Region` field contains West or East, the condition evaluates to true when a `Region` field contains West and East or when a `Region` field contains West or East values.

Setting Advanced Options in the Process Builder

The Process Builder lets you choose some advanced options for executing actions in your processes.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Process Builder Advanced Option Considerations

Keep these considerations in mind when choosing advanced options.

Reevaluate Records in the Process Builder

When you add objects to your process, you can choose to evaluate a record multiple times in a single save operation.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Nest Processes in the Process Builder

Invocable processes let you modularize sections of your processes and add more logic to them. An *invocable process* is a process that starts when another process invokes it. `The process starts when` in the process's properties controls whether a process is invocable.

Avoid Unwanted Actions in Processes

When you add criteria to your process, you can choose to execute actions when specified criteria change.

Process Builder Advanced Option Considerations

Keep these considerations in mind when choosing advanced options.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

- Avoid creating an infinite loop when allowing your process to reevaluate records. For example, if your process checks whether an account description changes and then updates an account description and creates a Chatter post every time an account record is created or edited, the process evaluates and triggers actions resulting in six Chatter posts.
- If you choose to evaluate a record multiple times in a single save operation when you specify an object for your process, we recommend not setting any of your criteria to No criteria—simply execute the actions!
- If your process uses ISCHANGED, ISNEW, or PRIORVALUE formula functions, we recommend that you don't use the advanced options. If you do use advanced options, keep these considerations in mind.
  - When a record is first created, ISNEW evaluates to true. If your process uses the ISNEW formula function and reevaluates a record multiple times in a single save operation, the process executes actions multiple times.

    For example, your process checks whether an account is created or updated. Each time the criteria is evaluated, ISNEW evaluates to true. The result is six Chatter posts. This example is true only if the process is triggered because an account record is created.

    - When ISNEW evaluates to true, the process updates the account's annual revenue and posts to Chatter.
    - When the process updates the account's annual revenue, the process then reevaluates the record (up to five additional times) because the record was changed.
  - When a record is created, PRIORVALUE returns the current value as the prior value. When a record is updated, PRIORVALUE returns the field value that was set immediately before the save operation started. If your process uses the PRIORVALUE formula function and reevaluates a record multiple times in a single operation, the process executes actions multiple times. If your process reevaluates a record multiple times in a single save operation and executes actions only when specified criteria changes, the prior value returns the values that existed before the record was saved.

    For example, your process checks whether an account is created or updated. Each time the record is reevaluated, the prior value of the account's type is Prospect. The result is six Chatter posts.

    - Wen *`PRIORVALUE([Account].Type) = 'Prospect'`* evaluates to true, the process updates the account's annual revenue and posts to Chatter.
    - When an account is created with *`Prospect`* as the account type, the criteria is always true until the end of the process transaction.
    - If the process is changed to update the account type to *`Other`* when the criteria is true, then for an account created with *`Prospect`* as the account type, the formula *`PRIORVALUE([Account].Type) = 'Prospect'`* is true until the end of the process transaction.
  - ISCHANGED always evaluates to false when a record is first created.

For example, your process checks whether an account description changes—`ISCHANGED([Account].Description)`—and the process also reevaluates records multiple times in a single save operation. If an account is first created with a blank description value and another process updates the account description in the same save operation, ISCHANGED evaluates to true every time the record is reevaluated because it compares the account description value when the record was first created (a blank value) with whatever is set for the current value.

Let's say this same process creates a Chatter post every time ISCHANGED([Account].Description) evaluates to true. This process would create a recursive loop resulting in six Chatter posts because ISCHANGED evaluates to true throughout the save operation.

Reevaluate Records in the Process Builder

When you add objects to your process, you can choose to evaluate a record multiple times in a single save operation.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

It's kind of like using a roundabout instead of a four-way stop to control process traffic. Instead of stopping and waiting for separate save operations, reevaluating records helps your business traffic flow a little more freely.

If you choose this option, the process can evaluate the same record up to five additional times in a single save operation. It might reevaluate the record because a process, workflow rule, or flow updated the record in the same save operation. When a record is reevaluated, the process uses the most recent values for that record.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

For example, your sales review process includes multiple steps, approvals, notifications, and fields that need to be updated. Some of these changes may be part of your process, or they may be managed by other workflow rules or flows in your org. If you let the process reevaluate a record multiple times in a single save operation, you can manage and evaluate all of these changes—even changes from other processes—in a single save operation in your process.

SEE ALSO:

Process Builder Advanced Option Considerations

Nest Processes in the Process Builder

Invocable processes let you modularize sections of your processes and add more logic to them. An *invocable process* is a process that starts when another process invokes it. `The process starts when` in the process's properties controls whether a process is invocable.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To invoke a process from another process, you configure a Processes action. That configuration includes passing a record to the invocable process, which is how the process knows which record to start with. Because the record is passed from one process to another, the invocable process receives a certain version of that record. That version differs depending on when the Processes action is executed.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Immediate Action

When you invoke a process through an immediate action, the process receives the values that the record contained when the top-level process starts.

Example: Process 1 updates an account and then invokes Process 2 based on that account. Process 2 receives the version of the account when Process 1 started.

Scheduled Action

When you invoke a process through a scheduled action, the process receives the latest values for the record.

Example: Process 1 updates an account and, 15 minutes later, invokes Process 2 based on that account. Process 2 receives the latest version of the account from the database.

When Should I Build an Invocable Process?

Do you find yourself building the exact same actions for multiple action groups? Configure those actions one time in an invocable process, and then invoke that process from all the relevant action groups. Later, to update those actions, update the one invocable process. Then all the other processes use the updated actions.

Another cool scenario for invocable processes: nesting simple logic. Processes handle simple “if/then” statements. But what if you must nest some of those statements? Rather than having to build a flow or write code, build the second level of logic into another process. Invoke the second process from the first, and voila!

Example: Let’s say you handle all of your case management in a single process. But you must treat escalated cases for high-revenue accounts differently from escalated cases for regular accounts. If an account whose renewal date is less than a month away escalates the case, notify the account owner, the regional manager, and the VP of that region. If an account whose renewal date is more than a month away escalates the case, notify only the account owner and the regional manager.

To do so, you build an invocable process. Let’s call it “Escalated Cases.” The process operates on the Case object and has two criteria nodes.

- The first criteria node evaluates whether the associated account’s renewal date is less than a month away. When a case meets that criteria, the process posts to the account’s feed with a link to the case and mentions the account owner, regional manager, and regional VP.
- The second criteria node has no criteria. If a case doesn’t meet the first node’s criteria, the process performs the same action, except that it doesn’t mention the regional VP.

Now back to the process that automates your case management. You already have a criteria node that checks whether the case is escalated. Add a Processes action to that criteria’s action group, and configure the action to invoke the “Escalated Cases” process.

Avoid Unwanted Actions in Processes

When you add criteria to your process, you can choose to execute actions when specified criteria change.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For example, your process sends an email alert whenever a case has an Escalated status. Let's say your support team repeatedly updates the case description with new information. Whenever the case is saved with a new description, your process can check specifically whether the Escalated status changed, rather than repeatedly sending email alerts. This way, the process executes actions only if the status was changed to Escalated during the latest update.

Tip: Check out this short video Avoid Unwanted Actions in Your Process to learn more about this option.

This setting isn't supported if:

If Yes is...	Actions are executed if...	Actions are not executed if...
Selected	• The record was created. • The record was updated. Its current values meet the conditions, and its most recent previous values did not meet the conditions.	• The record's current values meet the conditions, and the record's most recent previous values met the criteria. • The record's current values don't meet the conditions.
Deselected	• The record was created. • The record was updated, and its current values meet the conditions.	The record's current values don't meet the conditions.

- Your process starts only when a record is created.
- Your process starts when a record is created or edited and the criteria node doesn't evaluate any criteria.
- The criteria node evaluates a formula, but the formula doesn't include a reference to the record that started the process.
- Your process uses the Is changed operator in a filter condition.

SEE ALSO:

Process Builder Advanced Option Considerations

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Create a Process

To create a process, define its properties and which records it evaluates, and then add criteria nodes and actions.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. Define the Process Properties

   The process properties uniquely identify your process.

2. Configure the Process Trigger

   Every process includes a trigger, which tells the process when to start. How you configure that trigger depends on what type of process you're creating.

3. Add Process Criteria

   Define the criteria that must be true before the process can execute the associated actions.

4. Add Actions to Your Process

   After you define a criteria node, define the actions that are executed when the criteria are met. Actions are executed in the order in which they appear in the Process Builder.

5. Execute Actions for Multiple Criteria

   Choose whether to stop or continue your process after specific criteria are met and associated actions execute.

Define the Process Properties

The process properties uniquely identify your process.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. From Setup, in the Quick Find box, enter `Builder`, and select **Process Builder**.
2. Click **New**, or click the process name and then click Edit Properties.
3. Define the process properties by completing the fields.

Field	Description
`Process Name`	The name for your process, up to 255 characters. This name appears in the process management page, so name your process to differentiate it from other processes. To see the page in Setup, enter `Builder` in the Quick Find box, then select **Process Builder**.
`API Name`	The name that the API and managed packages use, up to 79 characters. This name must be unique across all processes and flows. The name must begin with a letter and use only alphanumeric characters and underscores. It can't include spaces, end with an underscore, or have two consecutive underscores.

Field	Description
	After it's saved, you can't change the process's API name.
`Description`	Optional. A description for your process. The description also appears in the process management page. It's intended to help you differentiate between processes, such as to explain what a process does.
`The process starts when`	Identifies when the process begins. You can set your process to start when: • A record changes • A platform event message is received • It's invoked by another process This field is available only when creating a process.
`Template`	Specifies whether the process is a template. When a template is installed from a managed package, the subscriber can view and clone the process and customize the clones. Non-template processes that are installed from managed packages can only be activated and deactivated. Suppose that your company needs a process that differs slightly for each country where you do business. You can create or install a template for the base process and then clone it to create each country-specific process. Even if you don't use managed packages, you can use this field to clearly identify the base process. This field is available only when editing a process.
`API Version for Running the Process`	Determines which versioned run-time behavior improvements the process adopts. Changing this field requires the Manage Flows permission. Before you select a new API version, review all run-time improvements that were delivered between the currently selected API version and the new API version. You can find all flow and process run-time improvements for an API version in the Salesforce Release Notes. By default, when you create a process, it runs in the latest API version. If you clone an existing process as a new process or process version, the existing process's run-time API version is used in the new process or process version. The run-time API version doesn't change as future Salesforce releases roll out. You decide when, if ever, to change the API version for running each process. This field lets you test and upgrade your processes one by one, and at your own pace. You can even opt to never adopt versioned updates for one or all your processes.

4. Click **Save**.

Configure the Process Trigger

Every process includes a trigger, which tells the process when to start. How you configure that trigger depends on what type of process you're creating.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Record Change
If the process starts when a record changes, associate the process with an object, and specify when to start the process.

Event
If the process starts when a platform event message is received, associate the process with a platform event and an object, and specify matching conditions. Because every process acts on a Salesforce record, it requires a single record as a starting point. That way, the criteria and actions know where to start evaluating and executing.

Invocable
If the process starts when another process invokes it, associate the process with an object.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials, Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Event processes are available in: **Performance**, **Unlimited**, **Enterprise**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow
  AND
  View All Data

Record Change

If the process starts when a record changes, associate the process with an object, and specify when to start the process.

1. Click **Add Object**.
2. Configure the trigger.

For this field ...	Select ...
Object	The object that you want to base this process on. Click **Find an object**. Type to filter the dropdown list.
Start the process	Which type of record change triggers the process. • only when a record is created • when a record is created or edited If you're familiar with workflow rules and you're wondering what happened to the third option (created, and anytime it's edited to subsequently meet criteria), don't worry! You see that setting when you add criteria nodes if you selected **when a record is created or edited**.
Recursion - Allow process to evaluate a record multiple times in a single save operation?	Yes, if you want the process to evaluate a record multiple times in a single save operation. To see this field, expand the **Advanced** area. When enabled, the process can evaluate the same record up to five more times in a single save operation. It reevaluates the record because a process, workflow rule, or flow updated the record in the same save operation. For more information, see Reevaluate Records in the Process Builder on page 932.

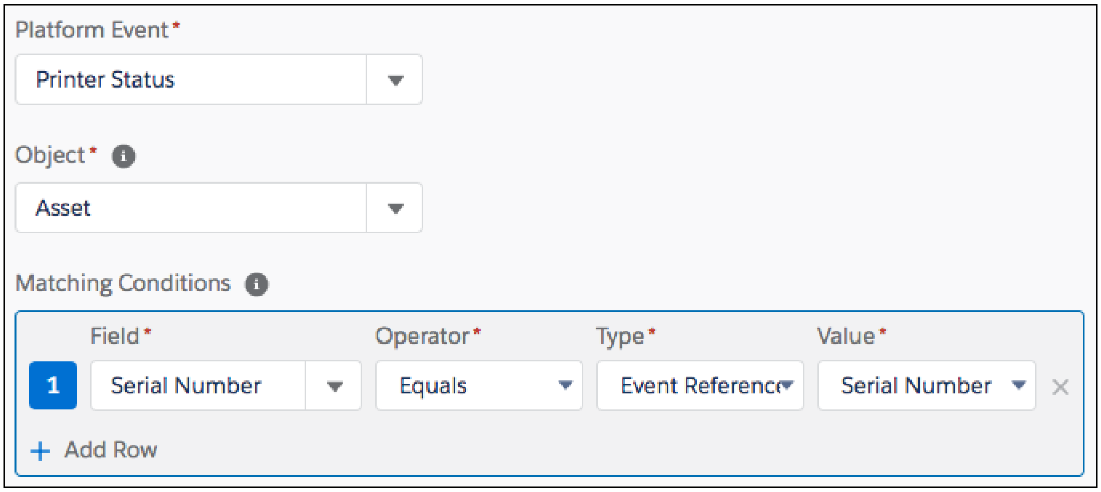
3. Before saving your changes, confirm the selection because you can't change the object after you save it.

Event

If the process starts when a platform event message is received, associate the process with a platform event and an object, and specify matching conditions. Because every process acts on a Salesforce record, it requires a single record as a starting point. That way, the criteria and actions know where to start evaluating and executing.

1. Click **Add Trigger**.
2. Configure the trigger.

For this field ...	Select...
Platform Event	The platform event whose event messages can start the process.
Object	The object whose records you want to associate with the event.
Matching Conditions	Criteria to identify one record to associate the event with. We recommend using an ID or other field that uniquely identifies records so the process can pick exactly one record. The process fails if it finds: • Multiple records that match the criteria. • No records that match the criteria.



Platform Event* Printer Status

Object* Asset

Matching Conditions

	Field*	Operator*	Type*	Value*
1	Serial Number	Equals	Event Reference	Serial Number

3. Before saving your changes, confirm the selection because you can't change the platform event or object after you save it.

Invocable

If the process starts when another process invokes it, associate the process with an object.

1. Click **Add Object**.
2. Select an object to associate with the process. Type to filter the dropdown list.

   This process can be invoked from any other process as long as the main process passes a record of this object type. For example, an Account-based invocable process can be called from a Contact-based record change process, because you can pass the contact's account to the invocable process.

3. Before saving your changes, confirm the selection because you can't change the object after you save it.

Add Process Criteria

Define the criteria that must be true before the process can execute the associated actions.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

When the criteria are met, the process executes the associated action group. When criteria aren't met, the process skips the action group and evaluates the criteria for the next action group.

1. Click **Add Criteria**.
2. Enter a name for this criteria node.

   Note: The name appears on the canvas, so use a name that helps you differentiate it from other criteria nodes. Actions have their own names, so we recommend summarizing only the criteria with this name. For example, if the criteria checks whether a case is escalated, use *`Is Escalated?`*.

3. Select the type of criteria that you must define. This selection determines which fields appear later in the dialog box.

If you need...	Select
The record to have certain field values. For example, to execute the associated actions on opportunity records with an amount greater than $5,000, set the filter to: `[Opportunity].Amount greater than $5000.00`	Conditions are met
To evaluate the record by using a formula. For example, to execute the associated actions on accounts whose annual revenue is over $1,000,000 when the account is changed by someone other than the owner, use this formula. `AND (([Account].LastModifiedBy.Id <> [Account].Owner.Id) , ([Account].AnnualRevenue > 1000000) )`	Formula evaluates to true
To simply execute the associated actions without evaluating the record. The process executes all the actions that are associated with this criteria node and, unless you specify otherwise, doesn't evaluate any remaining criteria nodes in the process. If you choose to stop your process after executing these actions, we recommend choosing this option for only the last criteria node in your process.	No criteria—execute the actions!

4. If you selected "Conditions are met":

   a. Define the filter conditions by identifying the field values that the process must evaluate.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

`Field`	Select the field whose value you want to evaluate. You can also evaluate values for records that are related to the one that started the process. To do so, click a related record with > next to the ID field. For example, if a contact record started the process, you can evaluate the value for the contact's account's `Annual Revenue` field. To access that field, click `Account Id` >, select **Annual Revenue**, and then click **Choose**.
`Operator`	The available operators depend on the field's data type.
`Type`	The available value types depend on the field' data type.
`Value`	Identify the value that you want to evaluate the field for. See Setting Values in the Process Builder on page 927 for details.

b. For `Conditions`, identify which conditions must be true for the process to execute the associated actions.

If you choose to use custom logic, enter up to 1000 characters by using:

- Numbers to refer to each condition
- *AND*, *OR*, or *NOT* to identify which combination of conditions must be true
- Parentheses to group parts of the string together

For example, if you enter *1 AND NOT (2 OR 3)*, the outcome evaluates to true if the first condition is true and the second or third outcome is false.

Tip: Ambiguous logic can cause validation errors. To avoid ambiguity, use parentheses in your custom logic. For example:

- *1 AND 2 OR 3* results in an error
- *1 AND (2 AND 3) OR 4* doesn't result in an error

5. If you selected "Formula evaluates to true," define the formula.

6. Optionally, to specify whether you want to execute the actions only if the record was created or edited to meet criteria, click **Advanced** at the bottom of the panel.

For details, see Avoid Unwanted Actions in Processes on page 934.

Note: This setting is available only if the process starts when a record is created or edited and you selected "Filter conditions are met" or "Formula evaluates to true."

7. Click **Save**.

SEE ALSO:

Execute Actions for Multiple Criteria

Add Actions to Your Process

After you define a criteria node, define the actions that are executed when the criteria are met. Actions are executed in the order in which they appear in the Process Builder.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

An action group can consist of a combination of immediate and scheduled actions. Immediate actions are executed when evaluation criteria are met. Scheduled actions are executed at a specified time. For example, Salesforce can automatically send an email reminder to the account team if a high-value opportunity is still open 10 days before the specified close date.

Before you begin, consider whether you want this action to be executed immediately or at a specific time. If you want to execute the action at a specific time, identify when those actions should be executed.

1. Click **Add Action**.
2. Select the type of action to create, and then fill out the fields to define the action.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Create a Record from a Process

Create a record by manually entering values or by using the values of related records.

Invoke a Process from Another Process

Invoke a process from another process. With invocable processes, you have the option of reuse so that you don't spend your time on repetitive work.

Create a Chatter Post from a Process

Post to the feed of a user, a Chatter group, or the record that started the process.

Use a Quick Action from a Process

Create a record, update a record, or log a call by using a quick action that you or another admin created for your organization.

Work with Quip from a Process

Create documents, chat rooms, and folders when important events occur. Attach a document to a record so your users have information in context. Update your spreadsheets with the latest Salesforce data. Send a message to a chat room or document. Add new slides to a deck, copy documents, add members to a document or chat, and more.

Launch a Flow from a Process

Start an autolaunched flow from your process to automate complex business processes. Create flows to perform logic and have events trigger the flows via processes without writing code.

Send an Email from a Process

Easily send an email from a process by using an email alert. Email alerts are configured outside of the Process Builder and contain the standard text, list of recipients, and template for an email.

Send a Custom Notification from a Process

Send customized notifications when important events occur. Alert an account owner if a new support case is logged while trying to close a deal, or send a notification for a workflow built entirely with custom objects. Add recipients and content to your custom notification, then add it to your process.

Send a Survey Invitation from a Process

Send an email invitation containing the link to a particular survey question or to launch a survey.

Submit a Record for Approval from a Process

Submit the record that started the process for approval.

Update Records from a Process

Update one or more records that are related to the record that started the process by manually entering values or by using the values from related records.

Call Apex Code from a Process

Add customized functionality to your process by calling Apex from the process.

Specify When Your Actions Execute with a Schedule

In record-change processes and event processes, you can schedule actions to execute at a specific time. An action group that supports scheduled actions can have multiple schedules. For example, you can schedule some actions to execute one day from now and others to execute three days from now.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Note: Before adding scheduled actions to your process, understand how they work. Review Scheduled Actions Considerations.

To schedule actions in a record-change process, make sure that one of the following options is true for your action group.

- The process starts only when a record is created (1).
- The process starts when a record is created or edited (2), and the associated criteria node executes actions only when specified changes are made (3).

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

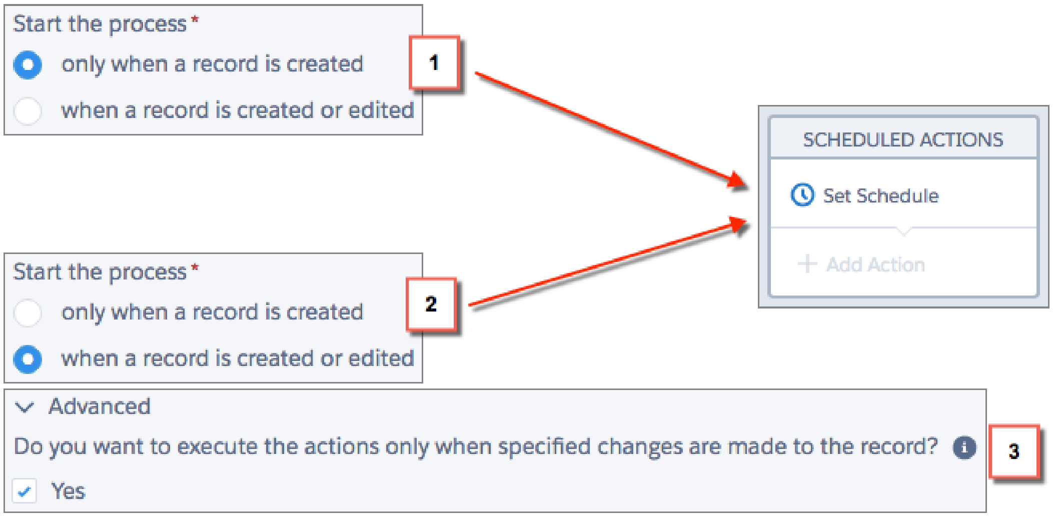
USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data



Start the process *

● only when a record is created

○ when a record is created or edited

Start the process *

○ only when a record is created

● when a record is created or edited

Advanced

Do you want to execute the actions only when specified changes are made to the record?

☑ Yes

1. In an action group that supports scheduled actions, click **Set Schedule**.
2. If you must schedule actions based on a date/time field on the record that started the process:
   a. Leave the first radio button selected.
   b. From the dropdown list on the right side of the panel, select the date to schedule the action.
      For example, if your process is based on an account record, choose the account's **Created Date**.
   c. Specify the number of days or hours before or after the field.

      For a record-change process, if the criteria for this action group are still met when this time occurs, Salesforce executes the scheduled actions. For an event process, the criteria aren't checked when this time occurs. If the criteria was met when the process started, Salesforce executes the scheduled actions.

3. If you must schedule actions after a certain number of days or hours from when the process is executed:
   a. Select the second radio button.
   b. Specify the number of days or hours from when the process is executed.

      If the criteria for this action group are still met when this time occurs, Salesforce executes the scheduled actions.

4. Save the schedule.

Create a Record from a Process

Create a record by manually entering values or by using the values of related records.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create an action and select "Create a Record" for the type, fill in the relevant fields to add the action to your process. The new record's `Created By` field is then set to the user who started the process by creating or editing a record.

Warning: If you create processes to replace any workflow rules, ensure that you delete those workflow rules when you activate the equivalent processes. Otherwise, both workflow rules and processes fire and cause unexpected results, such as overwritten records or redundant email messages. Do the same if you create processes to replace any Apex triggers.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For `Record Type`, select the object that you want to create a record for. To filter the dropdown list, type the name of the object to filter the dropdown list.

   When you select an object, at least one row appears to allow you to set field values for the new record.

   Warning: Rows appear automatically for fields required by the API. If you must provide values for other fields, we recommend that you refer to your organization's page layouts to determine which fields are required.

3. Set the record's field values.

`Field`	Select the field whose value you want to set. To filter the dropdown list, type the name of the field.

`Type`	Select the type of value that you want to use. The available types depend on the field that you've selected.
`Value`	Set a value for the field. by using the text entry field to manually enter a value or the field picker to use a field value from a related record. See Setting Values in the Process Builder on page 927 for details.

4. Click **Save**.

Tip:

- If you set up your process to create an account record, *`Name`* appears as a required field. If you want to create a person account, you can add *`LastName`* as a field but it doesn't appear as required by default. You can enter a dummy value for the *`Name`* field.
- When you create a record, required fields normally appear at the top of the list. However, if you save a Create a Record action, close the process, and then reopen the action, required fields don't always appear in the normal order.
- If a platform event is configured to publish immediately, the process publishes each event message outside of the database transaction. If the transaction fails and is rolled back, the event message is still published and can't be rolled back. So if you see an informational message under the selected platform event, consider whether you want the process to publish an event message only after the transaction commits successfully.

SEE ALSO:

Platform Events Developer Guide: Decoupled Publishing and Subscription

Platform Events Developer Guide: Platform Event Fields

Invoke a Process from Another Process

Invoke a process from another process. With invocable processes, you have the option of reuse so that you don't spend your time on repetitive work.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create an action and select "Processes" for the type, fill in the relevant fields to add the action to your process.

You can invoke processes with objects that share at least one unique ID. For example, in the Account and Case objects, the `AccountId` field is unique to Account and also used by Case. You can create an invocable process that updates a Case record. Then you can invoke it from:

- A process that updates an Account record's owner
- A process that adds an Account shipping address or updates it

When you create a process that invokes another process, each one counts toward your process and other applicable limits. DML limits in processes that invoke processes count as one transaction.

Warning: If you create processes to replace any workflow rules, delete those workflow rules when you activate the equivalent processes. Otherwise, both workflow rules and processes fire and cause unexpected results, such as overwritten records or redundant email messages. Do the same if you create processes to replace any Apex triggers.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

1. Enter a name for this action.

This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select an invocable process. You can only select active invocable processes.
3. Select your process variable. Remember that you can only select fields related to the object associated with the process you invoke.

Create a Chatter Post from a Process

Post to the feed of a user, a Chatter group, or the record that started the process.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

This action is available only if your organization has Chatter enabled. The feed item appears as if the user who started the process—by creating or editing a record—created the post.

Post to a User's Chatter Feed from a Process
Post to the feed of a user by identifying a specific user in your organization or a User lookup field on a record.

Post to a Chatter Group from a Process
Post to the feed of a specific Chatter group.

Post to a Record's Chatter Feed from a Process
Post to the feed of the record that started the process.

Mention a User or Group in a "Post to Chatter" Process Action
When you post to a Chatter feed from a process, you can mention users if you can reference the corresponding User ID field from the field picker.

SEE ALSO:

Chatter Settings

Considerations for Processes That Post to Chatter

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Post to a User's Chatter Feed from a Process

Post to the feed of a user by identifying a specific user in your organization or a User lookup field on a record.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created an action and selected "Post to Chatter" for the action type, fill in the relevant fields to add the action to your process.

Warning: If the feed that the process tries to post to isn't available when the process is triggered (for example, because the user is now inactive), the user sees an error and the process fails.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. In the `Post to` field, select User.
3. For `User`, select where you want to find the user.
4. Based on your selection for `User`, search for or browse for the user whose feed you want to post to.

   When you select a user from a record, you must ultimately select a field that contains a user's ID—for example, `Owner ID` or `User ID`.

5. Fill out the message that you want to post. You can insert merge fields, add a topic, and mention users or groups.

   The message can contain up to 10,000 characters.

   You can only reference topics that exist. If you reference a merge field and that field doesn't have a value, it appears as a blank value.

6. Save the action.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

SEE ALSO:

Chatter Settings

Post to a Chatter Group from a Process

Post to the feed of a specific Chatter group.

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created an action and selected "Post to Chatter" for the action type, fill in the relevant fields to add the action to your process.

> **Warning:** If the feed that the process tries to post to isn't available when the process is triggered, the user sees an error and the process fails.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. In the `Post to` field, select Chatter Group.
3. For `Group`, search for the Chatter group whose feed you want to post to.
4. Fill out the message that you want to post. You can insert merge fields, add a topic, and mention users or groups.

   The message can contain up to 10,000 characters.

   You can only reference topics that exist. If you reference a merge field and that field doesn't have a value, it appears as a blank value.

5. Save the action.

SEE ALSO:

Chatter Settings

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Post to a Record's Chatter Feed from a Process

Post to the feed of the record that started the process.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

You can post to the record's Chatter feed only if feed tracking is enabled for the object that the process is associated with.

After you've created an action and selected "Post to Chatter" for the action type, fill in the relevant fields to add the action to your process.

Warning: If the feed that the process tries to post to isn't available when the process is triggered (for example, because the user is now inactive), the user sees an error and the process fails.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. In the `Post to` field, select This Record.
3. Fill out the message that you want to post. You can insert merge fields, add a topic, and mention users or groups.

   The message can contain up to 10,000 characters.

   You can only reference topics that already exist. If you reference a merge field and that field doesn't have a value, it appears as a blank value.

4. Save the action.

SEE ALSO:

Chatter Settings

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Mention a User or Group in a "Post to Chatter" Process Action

When you post to a Chatter feed from a process, you can mention users if you can reference the corresponding User ID field from the field picker.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

When you're configuring the Post to Chatter action:

1. In the Message field, enter `@[]`.
2. Place your cursor between the square brackets.
3. Click **Merge Field**, navigate to the user who you want to mention, select the corresponding ID field, and click **Choose**.
   The field reference appears between the square brackets.

```
@[{!fieldReference}]
```

4. Save the action.

Example: To @mention a case's account owner, navigate to the account's fields and select **Owner ID**. Insert that field reference between the square brackets in `@[]`, so that the result is:

```
@[{![Case].Account.OwnerId}]
```

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Use a Quick Action from a Process

Create a record, update a record, or log a call by using a quick action that you or another admin created for your organization.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Quick actions can be object-specific or global actions. Only Create, Update, and Log a Call actions are supported. To use a quick action from a process, the action must exist in your organization.

If your organization is using quick actions to help your users more easily create and update records, you can also use those actions in your process. When you use these quick actions in a process, you can only set values for fields that are part of the action's layout. If you don't already have one of these actions created, see Create Global Quick Actions or Create Object-Specific Quick Actions for details.

After you've created an action and selected "Quick Actions" for the type, fill in the relevant fields to add the action to your process. The new or updated record appears as if the user who started the process—by creating or editing a record—created or updated it.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

1. Enter a name for this action. This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.
2. Filter to specify the kind of action you want to use.

Filter Search By	Lets You Search Through
Global actions	All global actions in your organization. You then filter even further by selecting the Type of actions that you must search through.
Object	All object-specific actions in your organization that are associated with a certain `Object Name`. Objects can't filter global actions.
Type	All object-specific and global actions in your organization based on the type, such as Create a Record or Log a Call.

   - If you selected Global actions or Type, for `Type` select the specific type of quick action that you want to use.
   - If you selected Object, for `Object` search for and select the object that you want to filter by.

3. For `Action`, search for and select the action that you want to use.
4. Set field values for the action.

   Rows that appear automatically represent the action's required fields. To set values for the action's optional fields, add rows.

`Field`	Select the field whose value you want to set. To filter the dropdown list, type the name of the field. You can set values for fields that are included in the action's layout only.
`Type`	Select the type of value that you want to use. The available types depend on the field that you've selected.
`Value`	Set a value for the field. by using the text entry field to manually enter a value or the field picker to use a field value from a related record. See Setting Values in the Process Builder on page 927 for details.

5. Save the action.

Work with Quip from a Process

Create documents, chat rooms, and folders when important events occur. Attach a document to a record so your users have information in context. Update your spreadsheets with the latest Salesforce data. Send a message to a chat room or document. Add new slides to a deck, copy documents, add members to a document or chat, and more.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Create a Document, Folder, or Chat Room from a Process

Use Process Builder to create documents, folders, and chat rooms.

Add a Document to a Folder from a Process

Add a document to one or more folders.

Remove a Document from a Process

Remove a document from a folder. Make a shared document private again.

Add a Live App to a Template from a Process

Keep your templates up to date with the latest Salesforce data. Add live Salesforce records and list views to your templates using Process Builder.

Attach a Document to a Record from a Process

Keep information in context by attaching a document to a Salesforce record.

Copy a Document from a Process

To use a document as a template, create a copy. By default, copied documents are saved to the running user's Private folder in Quip.

Add Members to a Document or Chat from a Process

Add up to 50 members to a document or chat.

Add Members with Different Access Levels to a Document from a Process

Automatically share a document with members with different permissions using Process Builder.

Remove Document Members from a Process

Trigger a process to auto-remove users from a document when the collaboration is over.

Copy Content from a Process (Retired)

Copy content from one slide deck to another.

Edit a Document from a Process

Add content to an existing document.

Update a Template Section from a Process

Edit a section of a template using text detection. Update sections of cloned documents at scale.

Copy Content with Live Paste from a Process

Copy content from a source document and paste it with Live Paste in a new document. After you update the source content, set the content to automatically update in all documents that reference it.

Update Content Based on a Document Range from a Process

Edit or add content to a defined template section, called a document range, when something changes in Salesforce.

Edit a Spreadsheet from a Process

Add data to an existing spreadsheet.

Edit a Slide from a Process (Retired)

Insert a new slide or change an image in an existing slide deck.

Lock Document Edits from a Process

Lock edits to mark a document as complete.

Lock Section Edits from a Process

To keep a document or template section safe from edits, lock it.

Export a Document to a PDF from a Process

To mark a document as complete or to keep a document view-only for record keeping, export it to a PDF. You can choose to attach the PDF to a document or to a Salesforce record.

Send a Message from a Process

Send a message in a chat room or in a document. Messages sent in a document appear as inline comments or in the document body.

Copy Comments from a Process

Copy comments from a template's source document to the newly-created target document.

Create a Document, Folder, or Chat Room from a Process

Use Process Builder to create documents, folders, and chat rooms.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Create New Document**, **Create New Folder**, or **Create Chat**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Choose a document, folder, or chat name.

   Names can be formatted as a string, field reference, global constant, or formula.

3. Enter the content that you want to add to your chat or document.

   This step is optional for new documents.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Add a Document to a Folder from a Process

Add a document to one or more folders.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Add Document to Folder**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document you want to add.
3. Enter the URL of the folder where you want to add your document.

   Add your document to multiple folders by adding commas between each folder URL.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

Remove a Document from a Process

Remove a document from a folder. Make a shared document private again.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Remove Document from Folder**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document you want to move.
3. Enter the URL of the folder your document is in.

   To remove a document from multiple folders, separate folder URLs with commas.

   Note: Removing a document from your Private folder removes your access to it.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

Add a Live App to a Template from a Process

Keep your templates up to date with the latest Salesforce data. Add live Salesforce records and list views to your templates using Process Builder.

> Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

After you create a Quip action and select **Create New Document** or **Edit Document**, fill in the relevant fields to add the action to your process.

1. From the Content Type dropdown, select **Quip Live App**.
2. To add a live Salesforce record, select **Salesforce Record**.
   a. Enter the Salesforce Record ID.

      > Note: To add a dynamic Salesforce record that updates based on the record that the document is embedded in, enter the value as a Reference. To add a specific record, enter the numbers that appear in the record URL as a String.

   b. These steps are optional and used as placeholders if the record can't be found.
   c. Optional: Enter the Salesforce record name.
   d. Optional: Enter the record type.
   e. Optional: Enter the name of the Salesforce org.
3. To add a live Salesforce list view, select **Salesforce List**.
   a. Enter the Salesforce List View ID.

      > Note: To add a dynamic Salesforce list view that updates based on the record that the document is embedded in, enter the value as a Reference. To add a specific list view, enter the numbers that appear in the record URL as a String.

   b. Enter the list view object type.
   c. Optional: Enter the name of the Salesforce org.
4. Save the action.

Take note of these considerations to using Process Builder to add a Salesforce live app to your templates.

- You can't select which record fields to display from Process Builder.
- The *owner* of a live app added by Process Builder to a template is the first user to open the copied document. Only the live app *owner* can save changes to Salesforce. Other users can edit and comment on the live app, but these changes don't sync to Salesforce.

> Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Attach a Document to a Record from a Process

Keep information in context by attaching a document to a Salesforce record.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Attach Document to Record**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For **Document URL**, enter the URL of the document you want to attach to a record.
3. Select the record type that you want to attach a document to, and then click Choose.
4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Copy a Document from a Process

To use a document as a template, create a copy. By default, copied documents are saved to the running user's Private folder in Quip.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Copy Document**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For Document URL, enter the URL of the document you want to copy.

   By default, newly copied documents appear in the Private folder in Quip.

   Note: Newly copied documents aren't automatically attached to the record. See Step 5 for more info.

3. Use the Advanced section to enter a document title, add members by email address, or add the document to a specific parent folder.
4. Save the action.
5. Optional: To attach the newly created document to the record and use Synced Sharing, use the Attach Document to Record action after the Copy Document action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

SEE ALSO:

Automate Pricing Proposals with Flow Builder

Automate Close Plans with Flow Builder

Add Opportunity Team Members to a Close Plan

Add Members to a Document or Chat from a Process

Add up to 50 members to a document or chat.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Add Members to Document** or **Add Members to Chat**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document or chat where you want to add members.
3. Enter up to 50 email addresses.

   Emails must belong to Quip users in the same Quip site as the acting user.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Add Members with Different Access Levels to a Document from a Process

Automatically share a document with members with different permissions using Process Builder.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Add Members to Document** or **Add Members to Chat**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

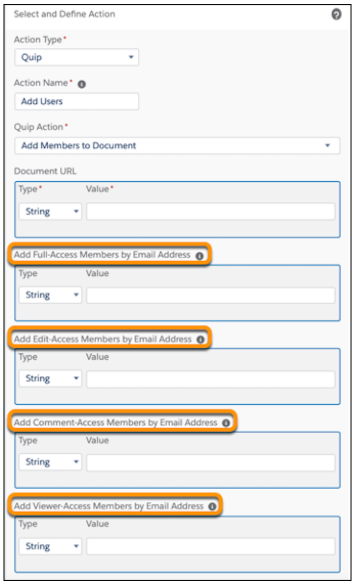
   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document or chat where you want to add members.
3. Enter the email addresses of the members you want to add based on the access level you want to grant.

You can enter up to 50 email addresses per access level. Emails must belong to Quip members in the same Quip site as the acting member.

Quip Access Level	Description
Full Access	Full-access members can view, comment on, edit, and share documents that they're added to.
Edit Access	Edit-access members can view, comment on, and edit documents that they're added to.
Comment Access	Comment-access members can view and comment on documents that they're added to.
View Access	View-access members can view documents that they're added to.

4. Save the action.



Select and Define Action

Action Type*: Quip

Action Name*: Add Users

Quip Action*: Add Members to Document

Document URL: Type*: String, Value*:

Add Full-Access Members by Email Address: Type: String, Value:

Add Edit-Access Members by Email Address: Type: String, Value:

Add Comment-Access Members by Email Address: Type: String, Value:

Add Viewer-Access Members by Email Address: Type: String, Value:

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Remove Document Members from a Process

Trigger a process to auto-remove users from a document when the collaboration is over.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Remove Members From Document**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document you want to manage access to.
3. Enter the email addresses of the members you want to remove from the document.

   You can enter up to 50 email addresses. Emails must belong to Quip members in the same Quip site as the acting member.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Copy Content from a Process (Retired)

Copy content from one slide deck to another.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Copy Content**, fill in the relevant fields to add the action to your process.

Warning: Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select **Slides** as your Document Type.
3. Enter the URL of the slide deck you want to copy.
4. Enter the slide number with the content you want to copy.
5. Enter the URL of the slide deck where you want to add content.
6. Enter the slide number where you want your copied content to appear.
7. Use the **Advanced** section to include anchor links instead of slide numbers.
8. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Edit a Document from a Process

Add content to an existing document.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Edit Document**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select **Document** as the Document Type.
3. Enter the URL of the document you want to edit.
4. Select the location in the document where you want to add content.

   To add content after or before a section or to replace a section, enter the document section anchor link.

5. Select the Content Type.
6. Enter the new content.
7. Optionally, select **Disable Extra Lines in Quip** to prevent Quip from automatically adding a blank line after each paragraph.
8. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Update a Template Section from a Process

Edit a section of a template using text detection. Update sections of cloned documents at scale.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

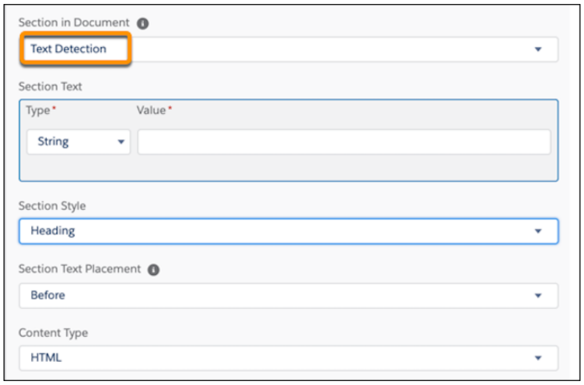
After you create a Quip action and select **Edit Document**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select **Document** as the Document Type.
3. Enter the URL of the document you want to edit.

4. To edit a document based on a section, select **After Section**, **Before Section**, or **Replace Section** as the location for your new content.
5. Use text detection to reference a document section by text. Under Section in Document, select **Text Detection**.



Section in Document: Text Detection

Section Text
Type*: String
Value*:

Section Style: Heading

Section Text Placement: Before

Content Type: HTML

6. Enter the section text you want to reference using text detection.
7. Under Section Style, select whether the section text is a heading, paragraph, or list.
8. Select the content type.
9. Enter your new content.
10. Save your action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Copy Content with Live Paste from a Process

Copy content from a source document and paste it with Live Paste in a new document. After you update the source content, set the content to automatically update in all documents that reference it.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

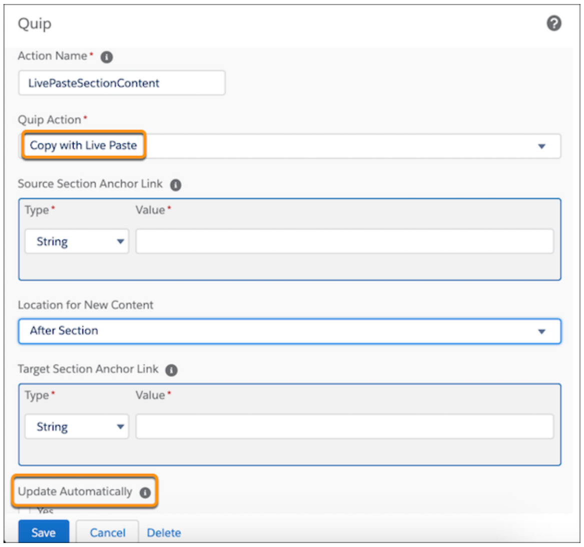
Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Copy with Live Paste**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the anchor links of the sections in the source document you want to copy with Live Paste. To copy content from multiple sections of the same document, enter anchor links and separate with commas. Your content appears in the order that the anchor links are entered. You can't copy content from multiple documents at the same time.



Quip

Action Name*: LivePasteSectionContent

Quip Action*: Copy with Live Paste

Source Section Anchor Link

Type*: String

Value*:

Location for New Content: After Section

Target Section Anchor Link

Type*: String

Value*:

Update Automatically

3. Select the location in the document where you want to paste your content. Live pasted content can appear at the end or beginning of a document, before or after a section, or can replace a document section.
4. To paste content at the beginning or end of a document, enter the target document URL. To paste content in a target document based on a section, enter the anchor link of the section where you want your copied content to appear.
5. To have content copied with Live Paste automatically update in the target document, select **Update Automatically**.
6. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Update Content Based on a Document Range from a Process

Edit or add content to a defined template section, called a document range, when something changes in Salesforce.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Document ranges are supported only in documents and templates. To see your highlighted document ranges, use a template.

After you create a Quip action and select **Edit Document** or **Copy with Live Paste**, fill in the relevant fields to add the action to your process.

1. To add new content to a document based on a document range, select the Edit Document action.
   a. Under Location for New Content, select a document range placement. You can choose to add content after a document range, before it, or you can replace it.

EDITIONS

Available in: Lightning Experience

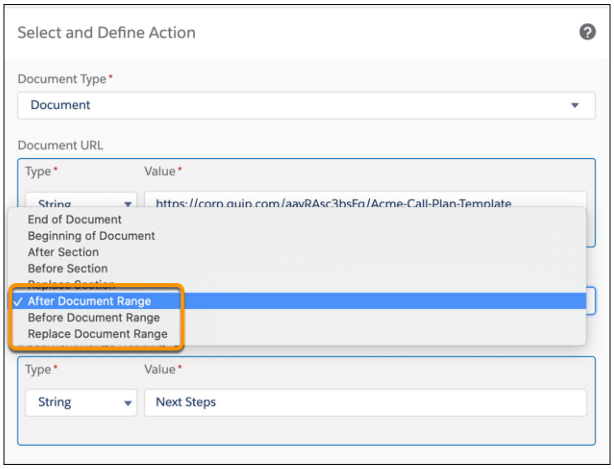
USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data



Select and Define Action

Document Type*: Document

Document URL

Type*: String — Value*: https://corp.quip.com/aavRAsc3bsFq/Acme-Call-Plan-Template

Type*: String — Value*: Next Steps

   b. Enter the document range heading. This is the heading text in your template that marks the start of a document range.

   c. Enter your new content and save the action.

2. To live paste existing content to a template, select the Copy with Live Paste action.

   a. Choose whether you want to copy content based on an anchor link or document range content type.

   b. Enter the URL of the template or anchor link you want to copy content from.

   c. To live paste content based on document range, select a document range placement. You can choose to paste your copied content after a document range, before it, or you can replace it.

   d. Enter the URL of the template where you want to paste your copied content.

   e. Enter the document range heading of the target template that you want to use to place your copied content.

   f. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Edit a Spreadsheet from a Process

Add data to an existing spreadsheet.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Edit Document**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select **Spreadsheet** as the Document Type.
3. Enter the URL of the spreadsheet you want to edit.
4. Select the location in the spreadsheet where you want to add content.
5. To add content after a section, before a section, or to replace a section, enter the Section Anchor Link.
6. Select **Row** or **Column** as the element type where you want to add content.
7. Enter the new content.
8. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Edit a Slide from a Process (Retired)

Insert a new slide or change an image in an existing slide deck.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Edit Document**, fill in the relevant fields to add the action to your process.

Warning: Quip is retiring slides on January 31, 2021. After this date, the Copy Content action in Process Builder and Flow Builder no longer works, and Slides isn't a valid document type for the Edit Document and Create Document actions. Tell Me More

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select **Slides** as the Document Type.
3. Enter the URL of the slide deck you want to edit.
4. Select **Insert New Slide** or **Change Image In Slide**.
5. Select the location in the slide deck where you want to add content.
6. To add content before or after a section, enter the slide number.

7. Use the **Advanced** section to include anchor links instead of slide numbers.
8. To add text to a slide, select **Text Layout** and add your content.
9. To add an image, select **Image Layout** and enter the URL of the image you want to add.
10. Save the action.

Lock Document Edits from a Process

Lock edits to mark a document as complete.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

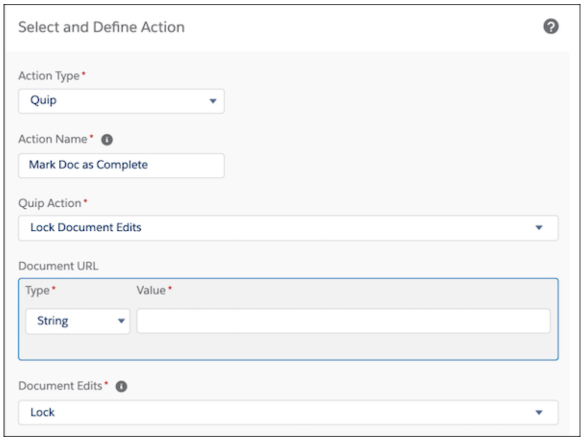
- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Lock Document Edits**, fill in the relevant fields to add the action to your process.


Select and Define Action

Action Type*: Quip

Action Name*: Mark Doc as Complete

Quip Action*: Lock Document Edits

Document URL

Type*: String

Value*:

Document Edits*: Lock

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document you want to lock.
3. To lock document edits, select **Lock**. To unlock document edits, select **Unlock**.

   Note: Only users with full access to a document can lock or unlock edits.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Lock Section Edits from a Process

To keep a document or template section safe from edits, lock it.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Lock Document Section Edits**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the anchor link URL of the section you want to lock.
3. To lock section edits, select **Lock**. To unlock section edits, select **Unlock**.

   Note: Only users with full access to a document can lock or unlock section edits.

4. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

Export a Document to a PDF from a Process

To mark a document as complete or to keep a document view-only for record keeping, export it to a PDF. You can choose to attach the PDF to a document or to a Salesforce record.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you create a Quip action and select **Export to PDF**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Optional. Enter the URL of the document you want to export to a PDF. To use a document housed in a URL field, set Type to **Field Reference**, and select the object's field.
3. Optional: To attach the PDF to a document, enter a target document URL. The PDF is added to the end of the document.
4. To attach the PDF to a Salesforce record, enter the record's Salesforce Organization ID and the Target Record ID. PDFs attached to a record are added to the record's Files component and Notes and Attachments component, and are visible to any user with access to the record.
5. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

SEE ALSO:

Automate Pricing Proposals with Flow Builder

Send a Message from a Process

Send a message in a chat room or in a document. Messages sent in a document appear as inline comments or in the document body.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created a Quip action and selected **Send Message in Document** or **Send Message in Chat**, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Enter the URL of the document or chat where you want to send your message.
3. To send a message in a specific section of the document body, enter the Document Section Anchor Link.
4. Enter the message you want to send.
5. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

Copy Comments from a Process

Copy comments from a template's source document to the newly-created target document.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. Create a Quip action in Process Builder and select **Copy Document**.
2. Under Advanced, select **Copy comments to new document**.
3. Save the action.

Note: To replace the existing Process Builder processes and Workflow Rules with flows before Process Builder and Workflow Rules reach end of support, see Quip Actions in Flow Builder.

EDITIONS

Available in: Lightning Experience

USER PERMISSIONS

To create, edit, or view processes:
- Manage Flow

  AND

  View All Data

Launch a Flow from a Process

Start an autolaunched flow from your process to automate complex business processes. Create flows to perform logic and have events trigger the flows via processes without writing code.

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To launch a flow from a process, you must create and activate the flow. The flow must be autolaunched.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For Flow, search for and select the flow that you want to launch from this process.

   Only active, autolaunched flows are available.

3. Optionally, click **Add Row** to set values for the flow's variables.

`Flow Variable`	Start typing the name of the flow variable whose value you want to set, or click ▾ to select a flow variable from the dropdown list. You can set values for any variables in the flow that allow input access. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of `null` is converted to an empty string. in the flow.
`Type`	Select the type of value that you want to set. For example, select **String** to manually enter the values for a Text collection variable, or select **Reference** to use the value of a record for a record variable.
`Value`	Set a value for the flow variable. • For collection variables, use the text entry field to specify a value. The value must match the collection variable's data type. • For record variables, use the field picker to select an ID field. The ID must correspond to a record whose object type matches the record variable's object. • For record collection variables, use the field picker to select a related list. The selected records' object type must match the record collection variable's object. For example, populate a record collection variable with all child contact records associated with the account that started the process. If the related list is empty when the flow tries to populate the record collection variable with field values from those records, the process fails.

4. Save the action.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Send an Email from a Process

Easily send an email from a process by using an email alert. Email alerts are configured outside of the Process Builder and contain the standard text, list of recipients, and template for an email.

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Before you begin:

- Make sure that the email alert you want to call from your process exists. If not, create the email alert on page 813.
- Understand the daily limits for emails sent from email alerts.

You can use only email alerts that are associated with the same object that the process is associated with. The record that started the process is used as the starting point for any merge fields that are used in the email alert.

After you've created an action and selected "Email Alerts" for the type, fill in the relevant fields to add the action to your process.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For `Email Alert`, type two or more letters to search for the email alert that you want to use to send an email.
3. Save the action.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Send a Custom Notification from a Process

Send customized notifications when important events occur. Alert an account owner if a new support case is logged while trying to close a deal, or send a notification for a workflow built entirely with custom objects. Add recipients and content to your custom notification, then add it to your process.

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Before you begin, make sure that the notification type you want to call from your process exists. If not, create a custom notification type.

After you've created an action and selected **Send Custom Notification** for the type, fill in the relevant fields to add the action to your process.

1. Enter an easily recognizable name for this action. The name appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.
2. Select a notification type.
3. Select a recipient category, and designate or find a recipient ID.
   - Current User — The user who initiated the record change, platform event, or process that triggered the process. This option is useful for confirmation notifications, such as a successful submission of a form.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

- Find User — The user who receives the notification each time this action is executed.
- User Field from a Record — A user referenced via UserId on the record that initiated the process or on a related record.
- Find Group — All users in the group that receives the notification each time this action is executed.
- Find Queue — All users in the queue that receives the notification each time this action is executed.
- Account Field from a Record — All users on the account team for an account referenced via AccountId on the record that initiated the process or on a related record. This option is available if you've enabled account teams for your org.
- Opportunity Field from a Record — All users on the opportunity team for an opportunity referenced via OpportunityId on the record that initiated the process or a related record. This option is available if you've enabled team selling for your org.
- Owner Field from a Record — An owner or queue referenced via OwnerId on the record that initiated the process or a related record. With this option, you can send a notification to all record owners, regardless of whether the owner is an individual owner or a queue.

4. Write a helpful notification title and body using text and merge fields.

   Note: The content of custom push notifications depends on the Display full content push notifications setting. If full content push notifications aren't enabled, only the notification title is sent.

5. Save the action.

Send a Survey Invitation from a Process

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Developer**, **Enterprise**, **Performance**, and **Unlimited** Editions.

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Send an email invitation containing the link to a particular survey question or to launch a survey.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you select **Send Survey Invitation** as the action, fill in the relevant fields.

1. Enter a name for this action.

   This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. Select an active survey.
3. Select a question or the survey link.

   Note: You can send email invitations for questions of the following types: Like or Dislike, Net Promoter Score (NPS), Rating, and Score.

4. Select the email template used to send the invitation.

   Important: The available templates depend on whether you choose to send a question or the survey link.

5. Select the recipient type.

   You can only send survey invitations to leads, contacts, and users in your org.

6. Select the recipient based on the object that's associated with the process.
7. Select your invitation settings.
8. Click **Save**.

Example: If you want to send an invitation to a case's contact, select Case as the object for the process, Contact as the recipient type, and Contact ID as the recipient.

SEE ALSO:

Customize the Survey Invitation Email Templates

Submit a Record for Approval from a Process

Submit the record that started the process for approval.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created an action and selected "Submit for Approval" for the type, fill in the relevant fields to add the action to your process.

Only the record that started the process is submitted. You can't submit any related records for approval.

1. Enter a name for this action. This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.
2. For `Approval Process`, indicate whether to submit the record through the default approval process or through a specific approval process.

   The process fails if:

   - The record is submitted to the default approval process, and there are no active approval processes for the record's object type.
   - The record is submitted to the default approval process, and it doesn't meet the criteria for any of the approval processes for the record's object type.
   - The record is submitted to a specific approval process, and it doesn't meet the entry criteria.
3. To submit the record to a specific approval process:
   1. Search for and select the approval process.
   2. Indicate whether to skip the entry criteria for the approval process.
4. For `Submitter`, identify who receives notifications about the approval request.

Value	Description
Current User	The user who triggered the process by creating or editing a record.
User Field from a Record	The user ID that's stored in a field value on the record that's being submitted for approval or another record.
Other User	A specific user in your organization.

   If the submitter isn't an allowed initial submitter on the approval process that runs, the process fails. Make sure that the initial submitters for the approval processes that are related to this object include all users who could trigger this process. For details about setting the initial submitters for an approval process, see Create an Approval Process with the Standard Wizard on page 833.

Any user with the "Modify All" permission to the object is allowed to submit a record for approval. They're permitted to submit the record, even if they aren't listed as initial submitter.

5. If necessary, enter submission comments. Don't reference merge fields or formula expressions.

   Submission comments appear in the approval history for the specified record. This text also appears in the initial approval request email if the template uses the `{!ApprovalRequest.Comments}` merge field.

6. Save the action.

Update Records from a Process

Update one or more records that are related to the record that started the process by manually entering values or by using the values from related records.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

> **Important:** Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

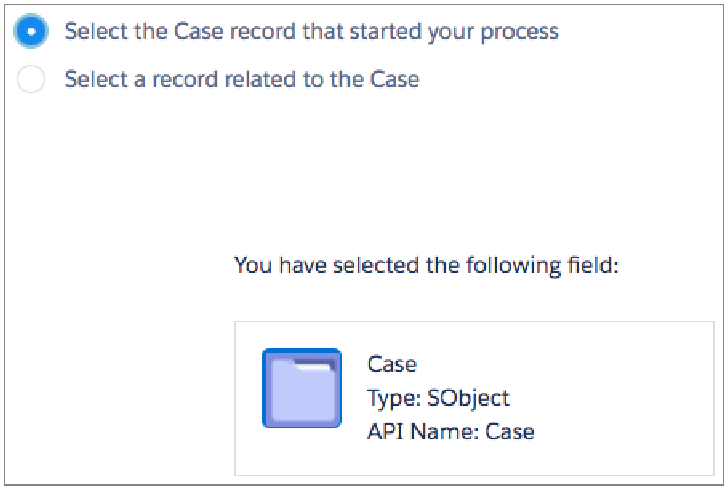
After you've created an action and selected "Update Records" for the action type, fill in the relevant fields to add the action to your process. The records' `Last Modified By` field is set to the user who started the process by creating or editing a record.

1. Enter a name for this action. This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.

2. For `Record Type`, select the record or records that you must update, and then click **Choose**.

   You can update only the record that started the process or records that are related to it. For example, you can reference *`[Case].ContactId`*, but not *`[Case].Contact.AccountId`*.

   - To update the record that started the process, click the appropriate radio button. For example, if your process is based on a case record, click next to **Select the Case record that started your process**.



● Select the Case record that started your process
○ Select a record related to the Case

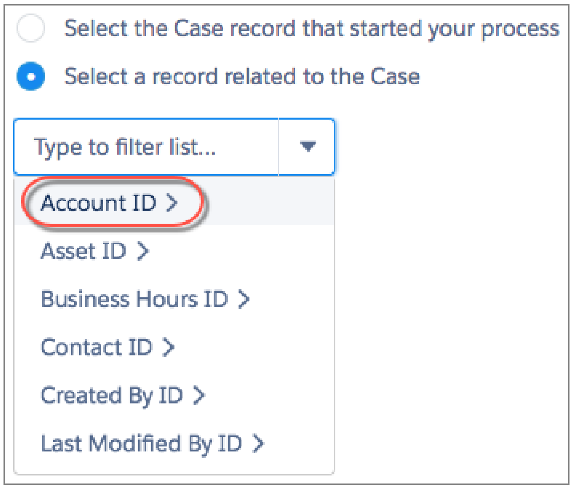
You have selected the following field:

Case
Type: SObject
API Name: Case

   - To update a record that's related to the record that started the process, click the appropriate radio button and select one of the field names in the dropdown list.

If you select a field that ends in "ID," you're selecting a single record. This field name corresponds to a lookup field on the original record. For details on lookup fields, see Custom Field Types.

For example, if a case record started the process and you select `Account Id`, this action updates the account that's associated with the case.

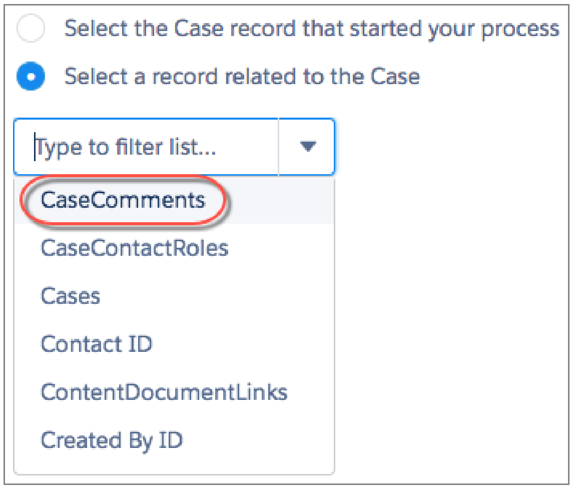


○ Select the Case record that started your process

● Select a record related to the Case

If you select a plural item that doesn't end in "ID," you're updating all the records of that object type that are related to the record that started the process. This plural item corresponds to child records of the original record, which can appear in a related list on the original record.

For example, if you select CaseComments, this action updates all the case comments that are related to the case.

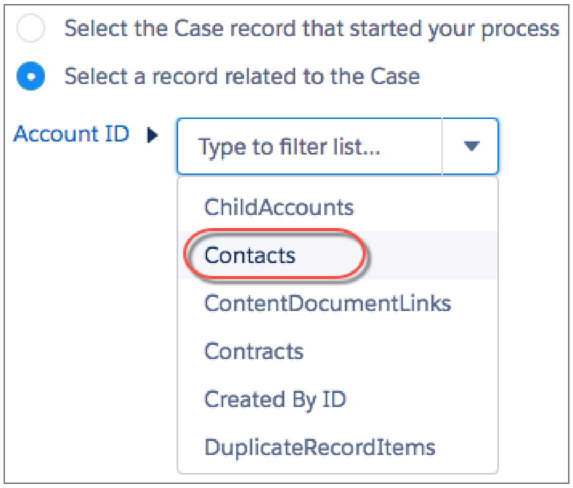


○ Select the Case record that started your process

● Select a record related to the Case

- To update fields on a related record, click a field with > next to it (ending in "ID") to access that record's fields.

  For example, let's say that, for a process that evaluates a case record (1), you want to update all contacts that are related to the case's parent account. Click **Account ID** > (2), then **Contacts** (3), and then **Choose**.



○ Select the Case record that started your process

● Select a record related to the Case

Account ID ▸ Type to filter list...

3. Optionally, specify conditions to filter the records you're updating. For example, if your process updates the status of a parent case, specify conditions so that you don't update the parent case if its status is set to On Hold.

   When you define conditions for updating records, you can't:

   - Reference a Long Text Area field
   - Reference a Rich Text field
   - Reference a child record's related fields.

     For example, you can reference *`[Case].ContactId`*, but not *`[Case].Contact.AccountId`*.

     When you define multiple filters, the filter logic usually defaults to `AND`. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with `OR`. For example, your filters check whether a case's Type equals Problem (1), Type equals Feature Request (2), and Escalated equals `true` (3). At run time, the filters are combined to `(1 OR 2) AND 3`.

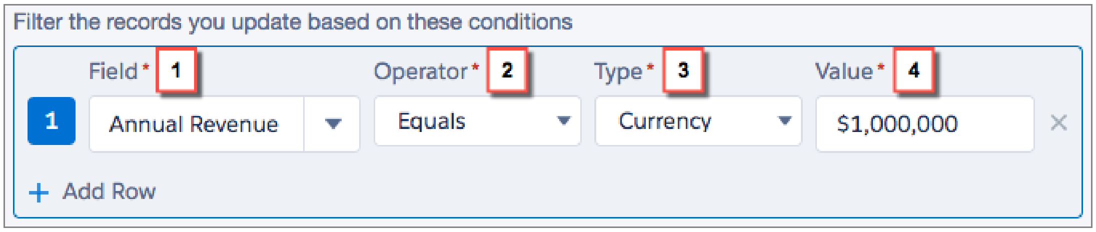
     If you're updating the record that started the process, Process Builder adds an implicit filter for you in the background: `[`***Object***`].Id equals myCurrentVariable.Id`. If you add filter criteria that set the record's ID to a value using the equals operator, at runtime the `[`***Object***`].Id equals` filters are combined using `OR` filter logic. For example, you update the case that started the process and add this filter: `[Case].Id equals 500D00000044XgV`. At runtime, your filter is combined with the implicit filter (`[Case].Id equals myCurrentVariable.Id`) with `OR`.

   a. Select **Updated records meet all conditions**.

   b. Set the conditions that you want to use to filter the updated records.

`Field`	Select the field whose value you want to evaluate.
`Operator`	The available operators depend on the field's data type.
`Type`	The available value types depend on the field' data type. See Process Builder Value Types on page 928 for details.
`Value`	Identify the value that you want to evaluate the field for.

For example, if your process updates account records, you can choose to update only accounts with an annual revenue (1) greater than (2) $1,000,000 (3).


Filter the records you update based on these conditions

Field*	Operator*	Type*	Value*
Annual Revenue	Equals	Currency	$1,000,000

4. Specify the new field values.

`Field`	Select the field whose value you want to set. To filter the dropdown list, type the name of the field. You can assign values to fields only on the record or records that you identified in the `Object` field. Use a separate Update Records action to update fields on related records.
`Type`	Select the type of value that you want to use. The available types depend on the field that you've selected.
`Value`	Set a value for the field. For example, if you select a Formula value type, click **Build a formula...** to create a formula value for the field.

5. Save the action.

Call Apex Code from a Process

Add customized functionality to your process by calling Apex from the process.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you've created an action and selected "Apex" for the type, fill in the relevant fields to add the action to your process.

Important: To use this action in a process, ask your developer to annotate the appropriate method with `@InvocableMethod`. For details, see "`InvocableMethod` Annotation" in the *Apex Developer Guide*.

The Apex class and the process are executed by the user whose action triggered the process.

1. Enter a name for this action. This text appears on the canvas and helps you differentiate this action from others in your process. The name truncates to fit on the canvas.
2. Choose an Apex class by entering the name of the class to filter results or select a class from the dropdown list.
3. If the class includes an invocable variable, you can manually enter values or reference field values from a related record.

   The value must match the variable's data type. You can set values for sObject and primitive type list variables only.

   - To set values for sObject variables and sObject list values, reference an object's related records, for example, all child contact records associated with the Account object that started the process.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

- To set a value for a primitive list variable (String, Integer, Time, and so on), select the String value type and enter a value in the text input field. You can't pass multiple values to lists.

4. Click **Save**.

> Note: If you define an Apex action in your process and then modify the Apex class by adding a standard field reference (for example, `User.Phone`), the Apex action is no longer visible in the process and must be added again.

Execute Actions for Multiple Criteria

Choose whether to stop or continue your process after specific criteria are met and associated actions execute.

> Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For each criteria node and associated action group, choose whether to stop the process after executing the actions or to continue the process and evaluate the next criteria node.

> Important: When a process continues to the next criteria node, it evaluates the values that the record had at the beginning of the process. For example:
>
> 1. The status of a case is New.
> 2. The case is edited.
> 3. The process evaluates Criteria 1. The conditions are met, so the process updates the case's status to Escalated.
> 4. The process evaluates Criteria 2 using the record values from step 2.
>
> If you want a process to react to changes that occur in the process, select the advanced option in the object node.

1. Make sure you've defined the next criteria and that your action group includes only immediate actions. You can't evaluate the next criteria when an action group contains scheduled actions.
2. To change what happens after actions execute, click **STOP** (1) or **EVALUATE THE NEXT CRITERIA** (2). Initially, each action group is set to stop after executing actions.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

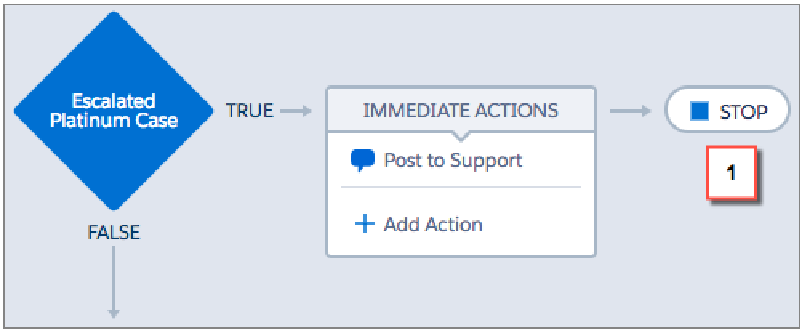
USER PERMISSIONS

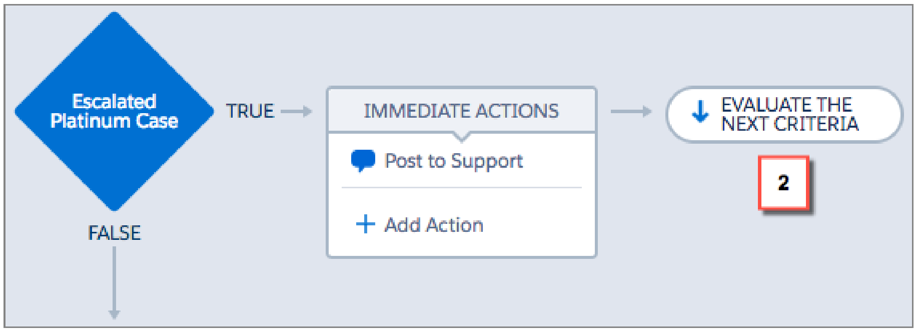
To create, edit, or view processes:

- Manage Flow

  AND

  View All Data





3. Save your changes, and your choice appears on the canvas.

SEE ALSO:

Reevaluate Records in the Process Builder

Process Management

Process Builder allows you to see and manage all your processes in one place.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To manage a process, from Setup, enter `Builder` in the Quick Find box, then select **Process Builder**.

From the process management page, you can:

- Create a process
- Edit a process
- Delete an inactive process
- See the status of your existing processes
- Sort your processes by name, description, object, last modified date, or status

When you open a process, you can:

- Clone the process
- Activate or deactivate the process
- Edit the process properties

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To manage processes:

- Manage Flow

  AND

  View All Data

From the list of paused flow interviews in Setup, you can:

- Monitor scheduled actions that haven't yet been executed
- Delete groups of scheduled actions that you no longer must wait for

Process Status

Each process has a status that determines whether the process can be edited, activated, or deleted.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Status	Description	Editable?
Active	The process has been activated. You can't edit an active process. However, you can clone it. Make any necessary changes to the cloned process and then activate it. Don't forget to deactivate the original process if appropriate.	No
Inactive	The process is inactive and can be activated.	Yes

Clone a Process

If you want to change an existing process, save a clone of that process. You can save the clone as either a new inactive process with its own version history, or as a new inactive version of the existing process.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create, edit, or view processes:

- Manage Flow

  AND

  View All Data

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To change an active process, you have a few options.

- Deactivate it, make changes, and then reactivate it.
- Clone it as an inactive version, make changes, and then activate the new version. The original version is automatically deactivated.
- Clone it as a new inactive process, make changes and then activate it. The original process isn't automatically deactivated, so consider whether it's appropriate for both processes to be active.

You can create up to 50 versions of a process, but only one version of a given process can be active.

1. From Setup, enter `Builder` in the `Quick Find` box, then select **Process Builder**.
2. Open the process or process version that you want to activate.
3. Click **Clone**.
4. You can create a version of the current process or a new process with its own version history.
5. Enter a name, API name, and description.
6. Click **Save**.

Activate a Process

Salesforce doesn't start using a new or revised process to evaluate records until you activate it.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After you activate a process, you can no longer edit it. However, you can click **Clone** to save the process as a new inactive process.

You can't activate a process unless it has:

- At least one defined criteria node
- At least one defined immediate or scheduled action

1. From Setup, enter `Builder` in the `Quick Find` box, then select **Process Builder**.
2. Open the process version that you want to activate.
3. Click **Activate**.

   If you activate a version of a process that already has an active version, the previously active version is automatically deactivated. To see that version later, refer to the process's version history.

After you've activated your process, consider creating or editing test records that will start the process to make sure it's working correctly. If you do, remember to delete those test records or return them to their previous values after you've confirmed that your process works as designed.

If you later want Salesforce to stop using a process to evaluate records as they're created or edited, open the active process and click **Deactivate**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To activate or deactivate processes:

- Manage Flow

  AND

  View All Data

  AND

  Customize Application

Delete a Process Version

If you no longer require a process version that you've defined, delete it.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To delete an active process, you must first deactivate it. You can't delete process versions with an Active status. If another process references your invocable process, you can't delete the invocable process. If a process has any scheduled actions, it can't be deleted until those pending actions have been executed or deleted.

1. In Setup, enter `Builder` in the Quick Find box, then select **Process Builder**.
2. Next to the appropriate process, click > to view all versions.
3. For the version that you want to delete, click **Delete**.

   If your process has only one version and you delete that version, the entire process is deleted.
4. Click **OK**.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To delete processes:

- Manage Flow

  AND

  View All Data

Monitor Your Processes' Pending Scheduled Actions

You can check which of your processes are waiting to execute scheduled actions.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. From Setup, enter *Flow* in the Quick Find box, then select **Paused And Failed Flow Interviews**

   Note: If Paused Flow Interviews isn't available as its own page, select **Flows** and scroll down to the list of paused interviews.

2. To see scheduled actions in the list of paused interviews, create a view.

   Note: We recommend displaying these fields.

   - `Flow API Name` or `Flow Name`—Contains the process name.
   - `Paused Date`—When the schedule started for the action group.
   - `Current Element`—Identifies the group of scheduled actions that the process is waiting to execute.

     The format of a `Current Element` value is `myWait_myRule_N`, where *N* is the number of the associated criteria and action group. For example, `myWait_myRule_2` indicates that the scheduled action is associated with the second criteria node in the process.

   - `Type`—Processes that are waiting to execute scheduled actions are of type Record Change Process.

SEE ALSO:

Delete Unexecuted Scheduled Actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To see unexecuted scheduled actions:

- View Setup and Configuration

Delete Unexecuted Scheduled Actions

If you no longer want to execute a process's scheduled actions, you can delete them from the list of paused flow interviews in Setup.

> Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. From Setup, enter `Flow` in the Quick Find box, then select **Paused Flow Interviews**.

   If Paused Flow Interviews isn't available as its own page, select **Flows** and scroll down to the list of paused interviews.

2. In the Flow API Name or Flow Name column, find the process whose scheduled actions you want to delete.
3. For each unexecuted group of scheduled actions that you want to delete, click **Del**, or click ▾ and select **Delete**.

SEE ALSO:

Monitor Your Processes' Pending Scheduled Actions

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To delete unexecuted scheduled actions:

- Manage Flow

Troubleshoot Processes

Use the error messages that appear in the Process Builder and the emails you receive when a process fails to help solve problems that arise when you're working with processes. When all else fails, look at the Apex debug logs for your processes.

> Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Common Reasons Why Processes Fail

Here are some common design problems that cause processes to fail.

Errors in the Process Builder

The API names for criteria nodes and actions are created in the background. When you create or update processes, you can see error messages that reference those names to help you identify specifically where the problem occurred.

What Happens When a Process Fails?

When a user does something that triggers a process, such as create a record, and the process fails, the user gets an error message. The error message includes the process name, error ID, and sometimes technical information that the user can give to you, the Salesforce admin. You can use the error ID to locate the detailed error email that is sent when the process failed.

Troubleshoot Processes with Apex Debug Logs

Use debug logs to find detailed information about your running processes after they finish running. For example, investigate why a process doesn't to trigger when a record meets the process's criteria, or explore the sequence of processes being executed.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Send Alerts When a Screen Flow Fails

To save time troubleshooting screen flows that fail, subscribe to the Flow Execution Error Event platform event. When a flow interview fails, Salesforce publishes a platform event message. In Process Builder, you can subscribe to the platform event and perform actions, such as posting to Chatter or sending custom notifications.

SEE ALSO:

Common Reasons Why Processes Fail

Common Reasons Why Processes Fail

Here are some common design problems that cause processes to fail.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

- A user creates or edits a person account. An Account-based process evaluates the record. The process's criteria node references an account field, for example, `[Account].Name Equals Acme`.
- The process references a field that hasn't been set. For example, you reference `[Contact].Account.Description` in your process. If the Account lookup field isn't set on the contact, the process fails because it doesn't know which account to reference.

  The workarounds for this issue depend on where the reference exists in the process.

  - If you reference it in filter conditions, add another filter condition that checks whether the lookup field is set. You can do the same workaround if it's referenced in a formula, for example, `[Contact].AccountId Is null False`.
  - Otherwise, consider making the fields required.

SEE ALSO:

Troubleshoot Processes

Errors in the Process Builder

What Happens When a Process Fails?

Errors in the Process Builder

The API names for criteria nodes and actions are created in the background. When you create or update processes, you can see error messages that reference those names to help you identify specifically where the problem occurred.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

API Name	Description
`myVariable_current.`*`field`*	*`field`* is the name of the field that's referenced. `myVariable_current` refers to the field values that the record had when it started the process. For example, `myVariable_current.Id` corresponds to the record's field value for `Id` when the record started the process.
`myVariable_old.`*`field`*	*`field`* is the name of the field that's referenced. `myVariable_old` refers to the most recent previous values of the record that started the process. For example, `myVariable_old.Id` corresponds to the record's field value for `Id` immediately before the record started the process.

Example:

```
The element has an invalid reference to "myVariable_current.AnnualRevenue".
```

`myVariable_current.AnnualRevenue` refers to the value for the field `AnnualRevenue` when the record started the process.

Note: Error or warning messages can refer to a "flow" instead of a "process". Those messages still apply to your process.

SEE ALSO:

Common Reasons Why Processes Fail

What Happens When a Process Fails?

When a user does something that triggers a process, such as create a record, and the process fails, the user gets an error message. The error message includes the process name, error ID, and sometimes technical information that the user can give to you, the Salesforce admin. You can use the error ID to locate the detailed error email that is sent when the process failed.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

The email includes the element that failed, the error message from that failure, and details about the criteria and actions that the process tried to execute. The subject line is `Error Occurred During Flow "`*`Process_Name`*`": `*`Error`*.

Example:

```
Error Occurred During Flow "Opportunity_Management": No
applicable approval process was found.
// The error occurred when the LeadConvertEmail process was
triggered.
An error occurred at element myRule_1_A1 (FlowActionCall).
No applicable approval process was found.
// The error occurred at the first action (A1) that's
associated with the
```

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

```
// first criteria node (myRule_1).

Flow Details
Flow Name: Opportunity_Management
Type: Workflow
Version: 3
Status: Active
Org: Acme (00DR00000000o82)
// The user triggered version 3 of the Opportunity_Management process.

Flow Interview Details
Interview Label: Opportunity_Management-3_Opportunity
Current User: Madison Rigsby (0051a000000qJXL)
Start time: 2/2/2017 11:21 AM
Duration: 0 seconds
// The process was triggered by user Madison Rigsby.

How the Interview Started
Madison Rigsby (0051a000000qJXL) started the flow interview.
Some of this flow's variables were set when the interview started.
myVariable_old = 0061a00000D3ibfAAB
myVariable_current = 0061a00000D3ibfAAB

ASSIGNMENT: myVariable_waitStartTimeAssignment
{!myVariable_waitStartTimeVariable} Equals {!Flow.CurrentDateTime}
Result
{!myVariable_waitStartTimeVariable} = "2/2/2017 11:21 AM"

DECISION: myDecision
Executed this outcome: myRule_1
Outcome conditions: and
1. {!myVariable_current.StageName} (Proposal/Price Quote) Equals Proposal/Price Quote
Logic: All conditions must be true (AND)
// The first criteria node (myRule_1) checks whether the opportunity’s StageName
// is “Proposal/Price Quote”. It is, so the process moves on to execute the associated
// actions.

SUBMIT FOR APPROVAL: myRule_1_A1
Inputs:
objectId = {!myVariable_current.Id} (0061a00000D3ibfAAB)
comment = null
Error Occurred: No applicable approval process was found.
// The process tries to execute the first associated action.
// The action fails because no approval processes exist that
// the record can be submitted to. Maybe the org doesn’t include
// any active Opportunity approval processes. Or maybe it does, but the
// record doesn't meet the entry criteria for any of them.
Salesforce Error ID: 593281227-1030 (-1996259564)
```

SEE ALSO:

Select Flow and Process Error Email Recipients

Common Reasons Why Processes Fail

Send Alerts When a Screen Flow Fails

Troubleshoot Processes with Apex Debug Logs

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Use debug logs to find detailed information about your running processes after they finish running. For example, investigate why a process doesn't to trigger when a record meets the process's criteria, or explore the sequence of processes being executed.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Tip: Make sure that your filters include FINER events in the WORKFLOW category. For details, see Debug Log Levels.

When using debug logs to troubleshoot a process, consider the following.

- Processes created in the Process Builder appear as flows and workflow rules in debug logs. The generated names have some resemblance to the process names, but they don't map one-to-one.
- Record change processes appear as flows of type Workflow. Invocable processes appear as flows of type InvocableProcess. Event processes appear as flows of type CustomEvent.
- Immediate actions that are executed in a block are strung together in the flow. If one action fails in the middle, then the following actions aren't executed.
- Scheduled actions are executed after a `FLOW_WAIT` element. The actions are executed similarly to immediate actions after the process resumes.
- `WF_CRITERIA_BEGIN` and `WF_CRITERIA_END` refer to the workflow rule criteria that are always set to true and not the criteria defined in your process.
- Here's how elements in the Process Builder correspond to flow debug events.

Process Builder Element	Flow Debug Event
Criteria	`FLOW_RULE_...`
Create a Record	`FLOW_ELEMENT_...`
Update Records	`FLOW_ELEMENT_...`
Post to Chatter	`FLOW_ACTIONCALL_...`
Submit for Approval	`FLOW_ACTIONCALL_...`
Email Alerts	`FLOW_ACTIONCALL_...`
Flows	`FLOW_ACTIONCALL_...`

Process Builder Element	Flow Debug Event
Processes	`FLOW_ACTIONCALL_...`
Apex	`FLOW_ACTIONCALL_...`
Schedule	`FLOW_WAIT_...`

Example: **Debugging Immediate Actions**

This example covers a process with an immediate Post to Chatter action.

Here's what you can tell from this first snippet.

- A lead named "Madison Rigby" triggers the process.
- The name of the process is `Hello_World`. The number appended to the name is the process version's ID: `301R000000009n0`.
- The process is set to trigger when a record is created (`ON_CREATE_ONLY`).

```
10:11:26.594 (595241802)|EXECUTION_STARTED
10:11:26.594 (595255829)|CODE_UNIT_STARTED|[EXTERNAL]|Workflow:Lead
10:11:26.594 (685753138)|WF_RULE_EVAL_BEGIN|Workflow
10:11:26.594 (686312384)|WF_CRITERIA_BEGIN|
   [Lead: Ms. Madison Rigsby 00QR0000001HqC4]|Hello_World301R000000009n0|
   01QR00000000Nz8|ON_CREATE_ONLY|0
```

In this snippet, the process compares the record's current values to the values it had before it was changed. `myVariable_current` contains all the record's current field values. `myVariable_old` contains all the field values of the record immediately before it was changed. In this example,`myVariable_old` has no values (null), because the process is evaluating a newly created lead.

```
10:11:26.594 (688919502)|WF_FORMULA|
   Formula:ENCODED:[treatNullAsNull]true|Values:
10:11:26.594 (689128428)|WF_CRITERIA_END|
   true
10:11:26.594 (695758445)|WF_SPOOL_ACTION_BEGIN|
   Workflow
10:11:26.594 (714823342)|WF_ACTION|
   Flow Trigger: 1;
10:11:26.594 (714900811)|WF_RULE_EVAL_END
10:11:26.594 (719777561)|WF_FLOW_ACTION_BEGIN|
   09LR000000005Td10:11:26.594 (720281142)|WF_FLOW_ACTION_DETAIL|
   09LR000000005Td|[Lead: Ms. Madison Rigsby 00QR0000001HqC4]|Id=09LR000000005Td|
   CurrentRule:Hello_World301R000000009n0 (Id=01QR00000000Nz8)
10:11:26.722 (722465931)|FLOW_CREATE_INTERVIEW_BEGIN|
   00DR00000000o82|300R00000004PQB|301R000000009n0
10:11:26.722 (740702983)|FLOW_CREATE_INTERVIEW_END|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|Hello World
10:11:26.594 (748183550)|WF_FLOW_ACTION_DETAIL|
   Param Name: myVariable_current,
   Param Value: ENCODED:{![treatNullAsNull]{!ID:this}},
   Evaluated Param Value: {Entity type: Lead, id: 00QR0000001HqC4MAK}|
   Param Name: myVariable_old,
   Param Value: {!old},
   Evaluated Param Value: null
```

When the process started:

- This instance of the process starts with the FLOW_START_INTERVIEW_BEGIN event.
- Each FLOW_START_INTERVIEW_LIMIT_USAGE event displays the usage of a given limit when the process started. In this example, the transaction hasn't done anything that counts toward a limit.
- A handful of variables are set. The process uses these variables to perform logic later.
  - myVariable_old is set to nothing because the record didn't exist before this transaction.
  - myVariable_current is set to the current values of the lead record.
  - myVariable_waitStartTimeVariable is set to the current time.

```
10:11:26.750 (750700361)|FLOW_START_INTERVIEWS_BEGIN|1
10:11:26.750 (751285739)|FLOW_START_INTERVIEW_BEGIN|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   Hello World
10:11:26.750  (751341782)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   SOQL queries: 0 out of 100
10:11:26.750 (751367432)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   SOQL query rows: 0 out of 50000
10:11:26.750 (751384035)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   SOSL queries: 0 out of 20
10:11:26.750 (751397896)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   DML statements: 0 out of 150
10:11:26.750 (751412225)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   DML rows: 0 out of 10000
10:11:26.750 (751427529)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   CPU time in ms: 0 out of 15000
10:11:26.750 (751472968)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Heap size in bytes: 0 out of 6000000
10:11:26.750 (751490226)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Callouts: 0 out of 100
10:11:26.750 (751505266)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Email invocations: 0 out of 10
10:11:26.750 (751519128)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Future calls: 0 out of 50
10:11:26.750 (751533892)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Jobs in queue: 0 out of 50
10:11:26.750 (751547542)|FLOW_START_INTERVIEW_LIMIT_USAGE|
   Push notifications: 0 out of 10
10:11:26.750 (752380627)|FLOW_VALUE_ASSIGNMENT|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   myVariable_old|
10:11:26.750 (754872639)|FLOW_VALUE_ASSIGNMENT|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   myVariable_current|
   {LastModifiedDate=2018-02-28 18:11:26, Company=Acme Wireless, Email=null,
   HasOptedOutOfFax=false, Latitude=null, MobilePhone=null, Industry=Apparel,
   CreatedById=005R0000000J01RIAS, Street=null, PhotoUrl=null,
   ConvertedOpportunityId=null, MasterRecordId=null,
   LastModifiedByID=005R0000000J01RIAS, Status=Contacted, IsDeleted=false,
   ConvertedAccountId=null, IsConverted=false, HasOptedOutOfEmail=false,
   LastViewedDate=null, City=null, Longitude=null, LeadSource=External Referral,
   CreatedByID=005R0000000J01RIAS, GeocodeAccuracy=null, State=null,
   CreatedDate=2018-02-28 18:11:26, Country=null, Id=00QR0000001HqC4MAK,
   LastName=Rigsby, AnnualRevenue=500000.0, Jigsaw=null, EmailBouncedDate=null,
   Description=null, ConvertedDate=null, DoNotCall=false, Rating=null,
   PostalCode=null, Website=null, LastReferencedDate=null, NumberOfEmployees=5,
   Salutation=Ms., ConvertedContactId=null, OwnerId=005R0000000J01RIAS,
   Phone=null, EmailBouncedReason=null, FirstName=Madison, IsUnreadByOwner=true,
   Title=null, SystemModstamp=2018-02-28 18:11:26, LastActivityDate=null,
   Fax=null, LastModifiedById=005R0000000J01RIAS,
   LastTransferDate=2018-02-28 18:11:26, JigsawContactId=null}
10:11:26.750 (755116990)|FLOW_ELEMENT_BEGIN|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   FlowAssignment|myVariable_waitStartTimeAssignment
10:11:26.750 (755457410)|FLOW_ASSIGNMENT_DETAIL|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   myVariable_waitStartTimeVariable|ASSIGN|2/28/2018, 10:11 AM
10:11:26.750 (756105710)|FLOW_VALUE_ASSIGNMENT|

2416dcc6212273331b3d50a38a161dd464e3e-7fdd|myVariable_waitStartTimeVariable|2018-02-28T18:11:27Z
10:11:26.750 (756182849)|FLOW_ELEMENT_END|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   FlowAssignment|myVariable_waitStartTimeAssignment
```

The process evaluates the first criteria.

In debug logs, a FLOW_RULE_DETAIL event represents a process criteria node. myRule_1 corresponds to the first criteria node in the process. Because the result of myRule_1 is true, the process executes the actions associated with the first criteria.

```
10:11:26.750 (757306870)|FLOW_ELEMENT_BEGIN|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   FlowDecision|myDecision
10:11:26.750 (757582110)|FLOW_RULE_DETAIL|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   myRule_1|true
10:11:26.750 (757616076)|FLOW_VALUE_ASSIGNMENT|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   myRule_1|true
10:11:26.750 (757683580)|FLOW_ELEMENT_END|
   2416dcc6212273331b3d50a38a161dd464e3e-7fdd|
   FlowDecision|myDecision
```

In this snippet, the immediate actions for the first criteria are executed. In the name myRule_1_A1, "A1" indicates that this element corresponds to the first action in the action group, which creates a task. The FLOW_BULK_ELEMENT_LIMIT_USAGE events indicate that the action increased the transaction's usage count toward two limits: the number of DML statements issued and the number DML rows processed.

```
10:11:26.750 (1898050716)|FLOW_ELEMENT_BEGIN|
   68211d9d9f918ee32db47d21247161de215ce5-7d38|
   FlowRecordCreate|myRule_1_A1
10:11:26.750 (1898121764)|FLOW_ELEMENT_DEFERRED|
   FlowRecordCreate|myRule_1_A1
10:11:26.750 (1898261705)|FLOW_ELEMENT_END|
   68211d9d9f918ee32db47d21247161de215ce5-7d38|
   FlowRecordCreate|myRule_1_A1
10:11:26.750 (1345712687)|FLOW_START_INTERVIEW_END|
   68211d9d9f918ee32db47d21247161de215ce5-7d38|Hello World
10:11:26.750 (1898350543)|FLOW_BULK_ELEMENT_BEGIN|
   FlowRecordCreate|myRule_1_A1
10:11:26.750 (1928183118)|FLOW_BULK_ELEMENT_DETAIL|
   FlowRecordCreate|myRule_1_A1|1
10:11:26.750 (2267557291)|FLOW_VALUE_ASSIGNMENT|
   68211d9d9f918ee32db47d21247161de215ce5-7d38|
   myRule_1_A1|true
10:11:26.750 (2267878414)|FLOW_BULK_ELEMENT_LIMIT_USAGE|
   1 DML statements, total 1 out of 150
10:11:26.750 (2267929106)|FLOW_BULK_ELEMENT_LIMIT_USAGE|
   1 DML rows, total 1 out of 10000
10:11:26.750 (2268002776)|FLOW_BULK_ELEMENT_END|
   FlowRecordCreate|myRule_1_A1|1|370
```

Then the process finishes.

```
10:11:27.977 (1978733709)|FLOW_START_INTERVIEWS_END|1

10:11:27.989 (1989764561)|WF_FLOW_ACTION_END|09LR000000005Td
10:11:27.989 (1998560773)|WF_ACTIONS_END| Flow Trigger: 1;
10:11:27.989 (1998600044)|CODE_UNIT_FINISHED|Workflow:Lead
10:11:27.989 (2000437095)|EXECUTION_FINISHED
```

Example: Debugging Scheduled Actions

Scheduled actions are logged separately from immediate actions. After the scheduled time occurs, an automated process executes the scheduled actions. However, the actions are still executed as the user who originally caused the process to run. The log uses coordinated universal time (UTC) instead of the user's time zone.

This example walks you through a debug log for a process with a scheduled Create a Record action.

Any events that start with `FLOW_WAIT_` provide information about a process schedule. `myWait_myRule_`***`int`*** always indicates a schedule, where *`int`* identifies which criteria node the schedule is associated with.

In this snippet:

- The schedules that are associated with the first criteria node (`myWait_myRule_1`) are evaluated.
- The defined time for the first schedule has passed (`myWaitEvent_myWait_myRule_1_event_0`).
- `FLOW_WAIT_RESUMING_DETAIL` indicates that the interview is resumed so that the process can execute its scheduled actions.
- The `myVariable_current` variable is updated with the latest values from the record that started the process originally.

```
10:21:35.461 (1461109547)|FLOW_BULK_ELEMENT_BEGIN|
   WaitInfo|myWait_myRule_1
10:21:35.461 (1467206801)|FLOW_WAIT_EVENT_RESUMING_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWait_myRule_1|myWaitEvent_myWait_myRule_1_event_0|DateRefAlarmEvent
10:21:35.461 (1467428864)|FLOW_WAIT_RESUMING_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWait_myRule_1|0FoRM0000004C9I
10:21:35.461 (1503485017)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWaitEvent_myWait_myRule_1_event_0|true
10:21:35.461 (1509382975)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myVariable_current|{Id=00QRM000003abIU2AY, IsDeleted=false,
   MasterRecordId=null, Salutation=null, FirstName=Another, LastName=Lead,
   Title=null, Company=Acme, Street=null, City=null, State=null, PostalCode=null,
   Country=null, Latitude=null, Longitude=null, GeocodeAccuracy=null, Phone=null,
   MobilePhone=null, Fax=null, Email=null, Website=null, PhotoUrl=null,
   Description=null, LeadSource=Advertisement, Status=New, Industry=null,
   Rating=null, AnnualRevenue=null, NumberOfEmployees=null, InternalSource=null,
   OwnerId=005RM000001cEmFYAU, HasOptedOutOfEmail=false, IsConverted=false,
   ConvertedDate=null, ConvertedAccountId=null, ConvertedContactId=null,
   ConvertedOpportunityId=null, IsUnreadByOwner=false,
   CreatedDate=2018-03-01 18:12:05, CreatedById=005RM000001cEmFYAU,
   LastModifiedDate=2018-03-01 18:12:05, LastModifiedById=005RM000001cEmFYAU,
   SystemModstamp=2018-03-01 18:12:05, LastActivityDate=null, DoNotCall=false,
   CreatedByID=005RM000001cEmFYAU, LastModifiedByID=005RM000001cEmFYAU,
   CampaignId=null, CampaignMemberStatus=null, HasOptedOutOfFax=false,
   LastViewedDate=null, LastReferencedDate=null,
   LastTransferDate=2018-03-01 18:12:05, Jigsaw=null, JigsawContactId=null,
   ConnectionReceivedDate=null, ConnectionSentDate=null, EmailBouncedReason=null,
   EmailBouncedDate=null}
10:21:35.461 (1512457819)|FLOW_BULK_ELEMENT_END|
   WaitInfo|myWait_myRule_1|0|47
```

In this snippet, the process makes sure that the record's date field isn't null. Specifically, it checks the date field that's referenced in the schedule.

```
10:21:35.461 (1514489368)|FLOW_ELEMENT_BEGIN|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowDecision|myPostWaitDecision_myWaitEvent_myWait_myRule_1_event_0
10:21:35.461 (1528928534)|FLOW_RULE_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myPostWaitRule_myWaitEvent_myWait_myRule_1_event_0|true
10:21:35.461 (1529027007)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myPostWaitRule_myWaitEvent_myWait_myRule_1_event_0|true
10:21:35.461 (1529230456)|FLOW_ELEMENT_END|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowDecision|myPostWaitDecision_myWaitEvent_myWait_myRule_1_event_0
```

Now to execute the actions associated with the schedule. First up is `..._myRule_1_event_0_SA1`.

- `myRule_1` corresponds to the first criteria node
- `event_0` corresponds to the first schedule associated with the criteria
- `SA1` corresponds to the first action in the schedule.

The action creates a record. With the `FLOW_BULK_ELEMENT_LIMIT_USAGE` events, we see that action increased the transaction's usage count toward two limits: the number of DML statements issued and the number DML rows processed.

```
10:21:35.461 (1529433132)|FLOW_ELEMENT_BEGIN|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1
```

```
10:21:35.461 (1529526210)|FLOW_ELEMENT_DEFERRED|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1
10:21:35.461 (1529619300)|FLOW_ELEMENT_END|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1
10:21:35.461 (1534801023)|FLOW_BULK_ELEMENT_BEGIN|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1
10:21:35.461 (1681358347)|FLOW_BULK_ELEMENT_DETAIL|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1|1
10:21:35.461 (1963485392)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWaitEvent_myWait_myRule_1_event_0_SA1|true
10:21:35.461 (1973349443)|FLOW_BULK_ELEMENT_LIMIT_USAGE|
   1 DML statements, total 1 out of 150
10:21:35.461 (1973886332)|FLOW_BULK_ELEMENT_LIMIT_USAGE|
   1 DML rows, total 1 out of 10000
10:21:35.461 (1974083134)|FLOW_BULK_ELEMENT_END|
   FlowRecordCreate|myWaitEvent_myWait_myRule_1_event_0_SA1|1|429
```

This snippet displays some internal logic that Process Builder performs for you. The process uses a variable to note that it has executed the action for this schedule, so that it doesn't accidentally duplicate the action.

```
10:21:41.527 (7529131090)|FLOW_ELEMENT_BEGIN|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowAssignment|myWaitEvent_myWait_myRule_1_event_0_postWaitExecutionAssignment
10:21:41.527 (7529875281)|FLOW_ASSIGNMENT_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWaitEvent_myWait_myRule_1_event_0_postActionExecutionVariable|ASSIGN|true
10:21:41.527 (7529943822)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWaitEvent_myWait_myRule_1_event_0_postActionExecutionVariable|true
10:21:41.527 (7530040052)|FLOW_ELEMENT_END|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   FlowAssignment|myWaitEvent_myWait_myRule_1_event_0_postWaitExecutionAssignment
```

Then the process evaluates whether to execute any of the other schedules. Notice that the conditions are no longer met for `..._event_0`. Because of the variable assignment in the previous snippet, the process doesn't re-execute the actions associated with that schedule.

There's only one schedule, so the process finishes.

```
10:21:41.527 (7530094566)|FLOW_ELEMENT_BEGIN|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   WaitInfo|myWait_myRule_1
10:21:41.527 (7530148328)|FLOW_ELEMENT_DEFERRED|
   WaitInfo|myWait_myRule_1
10:21:41.527 (7530225216)|FLOW_ELEMENT_END|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   WaitInfo|myWait_myRule_1
10:21:41.527 (7530291079)|FLOW_BULK_ELEMENT_BEGIN|
   WaitInfo|myWait_myRule_1
10:21:41.527 (7530832531)|FLOW_WAIT_EVENT_WAITING_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWait_myRule_1|myWaitEvent_myWait_myRule_1_event_0|DateRefAlarmEvent|false
10:21:41.527 (7530895796)|FLOW_WAIT_WAITING_DETAIL|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWait_myRule_1|0|
10:21:41.527 (7530968776)|FLOW_VALUE_ASSIGNMENT|
   2ef1ba5afce469a1e74b7b869161e25400a2-7f29|
   myWaitEvent_myWait_myRule_1_event_0|false
10:21:41.527 (7531068544)|FLOW_BULK_ELEMENT_END|
   WaitInfo|myWait_myRule_1|0|1
```

SEE ALSO:

Troubleshoot Processes

Send Alerts When a Screen Flow Fails

To save time troubleshooting screen flows that fail, subscribe to the Flow Execution Error Event platform event. When a flow interview fails, Salesforce publishes a platform event message. In Process Builder, you can subscribe to the platform event and perform actions, such as posting to Chatter or sending custom notifications.

Important: Starting in Summer '23, you can't create new processes. You can still activate, deactivate, and edit any existing processes. To migrate existing processes, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. Define the process properties on page 935 to start when a platform event message is received.
2. Configure a process trigger for a platform event on page 938.
3. Add the process criteria on page 939.
4. Create a Chatter post on page 945, or send a custom notification on page 968.

EDITIONS

Available in: both Salesforce Classic (not available in all orgs) and Lightning Experience

Available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

SEE ALSO:

Create a Process

Troubleshoot Processes

Platform Events Developer Guide: FlowExecutionErrorEvent

Workflow Rules

Workflow rules let you automate standard internal procedures and processes to save time across your org. A workflow rule is the main container for a set of workflow instructions. These instructions can always be summed up in an if/then statement.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For example: If it's raining, then bring an umbrella.

Workflow rules can be broken into two main components.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Criteria: the "if" part of the "if/then" statement. In other words, what must be true of the record for the workflow rule to execute the associated actions.

- Actions: the “then” part of the “if/then” statement. In other words, what to do when the record meets the criteria.

In the raining example, the criteria is “it’s raining” and the action is “bring an umbrella”. If the criteria isn’t met (“it isn’t raining”), then the action isn’t executed (“you don’t bring an umbrella”).

When a record meets all the criteria for a workflow rule, that rule’s actions are executed. Familiarize yourself with the automated actions that are available for workflow.

Create a Workflow Rule
Automate your organization’s standard process by creating a workflow rule.

Workflow Limits
Salesforce limits the number of total and active rules in your org, the number of time triggers and actions per rule. It also processes a limited number of daily emails and hourly time triggers.

Workflow Considerations
Learn the intricacies of workflow rules and workflow actions before you begin working with them.

Workflow Rule Examples
Looking for ideas on how workflow rules can help streamline your business? Check out these examples.

Monitor Pending Workflow Actions
When a workflow rule that has time-dependent actions is triggered, use the workflow queue to view pending actions and cancel them if necessary.

Workflow Terminology
These terms are used when describing workflow features and functionality.

SEE ALSO:

Choose Which Salesforce Flow Feature to Use

Create a Workflow Rule

Automate your organization’s standard process by creating a workflow rule.

Important: Starting in Winter ’23, you can’t create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Watch a Demo: Creating a Workflow Rule (Salesforce Classic)

1. Set the Criteria for Your Workflow Rule
   Get started with creating a workflow rule by selecting the object the rule relates to and configuring its criteria.
2. Add Automated Actions to Your Workflow Rule
   After you’ve set the criteria for your workflow rule, identify what to do when that criteria are met.
3. Identify Your Salesforce Org’s Default Workflow User
   Select a `Default Workflow User` that you want Salesforce to display with a workflow rule when the user that triggered the rule isn’t active.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:

- Customize Application

4. Activate Your Workflow Rule
   Salesforce doesn't trigger a workflow rule until you activate it.

SEE ALSO:

Workflow Considerations

Workflow Rule Examples

Set the Criteria for Your Workflow Rule

Get started with creating a workflow rule by selecting the object the rule relates to and configuring its criteria.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:
- Customize Application

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Note:

- If you have a workflow action that updates a field on a related object, that target object isn't the one that's associated with the workflow rule.
- To create workflow rules based on new case comments or incoming email messages that automatically update fields on their associated cases, choose Case Comment or Email Message. See Workflow Considerations for more information.
- To create a site usage rule, choose one of the following:
  - `Organization` (for monthly page views allowed and monthly page views used fields)
  - `Site` (for site detail, daily bandwidth and request time, monthly page views allowed, and other fields)
  - `User License` (for the monthly logins allowed and monthly logins used fields)

  The Organization and Site objects are only available if Salesforce Sites is enabled for your organization. The User License object isn't dependent on sites, and is only available if you have Customer Portals or partner portals enabled for your organization.
- This release contains a beta version of the workflow on the User object that is production quality but has known limitations.

`Evaluate the rule when a record is:`	**Description**
`created`	Evaluate the rule criteria each time a record is created. If the rule criteria is met, run the rule. Ignore all updates to existing records. With this option, the rule never runs more than one time per record.
`created, and every time it's edited`	Evaluate the rule criteria each time a record is created or updated. If the rule criteria is met, run the rule. With this option, the rule repeatedly runs every time a record is edited as long as the record meets the rule criteria.

Evaluate the rule when a record is:	Description
	If you select this option, you can't add time-dependent actions to the rule.
created, and any time it’s edited to subsequently meet criteria	(Default) Evaluate the rule criteria each time a record is created or updated. • For a new record, run the rule if the rule criteria is met. • For an updated record, run the rule only if the record is changed from not meeting the rule criteria to meeting the rule criteria. With this option, the rule can run multiple times per record, but it doesn’t run when the record edits are unrelated to the rule criteria. For example, suppose that for an opportunity record to meet the rule criteria, the opportunity probability must be greater than 50%. If you create an opportunity with a probability of 75%, the workflow rule runs. If you edit that opportunity by changing the probability to 25%, the edit doesn't cause the rule to run. If you then edit that opportunity by changing the probability from 25% to 75%, the edit causes the rule to run. With this last edit, the rule runs, because the record is changed from not meeting the rule criteria to meeting the rule criteria.

1. From Setup, enter *Workflow Rules* in the Quick Find box, then select **Workflow Rules**.
2. Click **New Rule**.
3. Choose the object to which you want this workflow rule to apply.
4. Click **Next**.
5. Give the rule a name and description.
6. Set the evaluation criteria. For example:

Option	Description
Evaluate the rule when a record is created	Evaluate the rule criteria each time a record is created. If the rule criteria is met, run the rule. Ignore all updates to existing records. With this option, the rule never runs more than one time per record.
Evaluate the rule when a record is created, and every time it’s edited	Evaluate the rule criteria each time a record is created or updated. If the rule criteria is met, run the rule. With this option, the rule repeatedly runs every time a record is edited as long as the record meets the rule criteria. If you select this option, you can't add time-dependent actions to the rule.
Evaluate the rule criteria each time a record is created, and any time it’s edited to subsequently meet criteria	(Default) Evaluate the rule criteria each time a record is created or updated. For a new record, run the rule if the rule criteria is met. For an updated record, run the rule only if the record is changed from not meeting the rule criteria to meeting the rule criteria. With this option, the rule can run multiple times per record, but it doesn’t run when the record edits are unrelated to the rule criteria.

Option	Description
	For example, suppose that for an opportunity record to meet the rule criteria, the opportunity probability must be greater than 50%. If you create an opportunity with a probability of 75%, the workflow rule runs. If you edit that opportunity by changing the probability to 25%, the edit doesn't cause the rule to run. If you then edit that opportunity by changing the probability from 25% to 75%, the edit causes the rule to run. With this last edit, the rule runs, because the record is changed from not meeting the rule criteria to meeting the rule criteria.

7. Enter your rule criteria. For example:
    - Choose `criteria are met` and select the filter criteria that a record must meet to trigger the rule. For example, set the filter to "Opportunity: Amount greater than 5000" if you want opportunity records with an amount greater than $5,000 to trigger the rule. If your organization uses multiple languages, enter filter values in your individual language. You can add up to 25 filter criteria, of up to 255 characters each.
8. Enter your rule criteria. For example:
    - Choose `criteria are met` and select the filter criteria that a record must meet to trigger the rule. For example, set the filter to "Opportunity: Amount greater than 5000" if you want opportunity records with an amount greater than $5,000 to trigger the rule. If your organization uses multiple languages, enter filter values in your individual language. You can add up to 25 filter criteria, of up to 255 characters each.
    - Choose `formula evaluates to true` and enter a formula that returns a value of "True" or "False." Salesforce triggers the rule if the formula returns "True."
9. Click **Save & Next**.

Example: Examples of useful workflow formulas include:

- If the number of filled positions equals the number of total positions on a job, update the `Job Status` field to "Filled."
- If mileage expenses associated with visiting a customer site are 35 cents per mile and exceed a $1,000 limit, automatically update the `Approval Required` field to "Required."
- If a monthly subscription-based opportunity amount is greater than $10,000, create a task for an opportunity owner to follow up 60 days after the opportunity is closed.

The `$Label` variable isn't supported in workflow rule formulas. Also, some functions aren't available in workflow rule formulas.

Tip: You can use merge fields for directly related objects in workflow rule formulas.

SEE ALSO:

Workflow Considerations

Add Automated Actions to Your Workflow Rule

After you've set the criteria for your workflow rule, identify what to do when that criteria are met.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:

- Customize Application

❗ Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Add an Immediate Action to Your Workflow Rule

Immediate actions, like their name suggests, are executed as soon as the workflow rule finishes evaluating the record.

Add a Time-Dependent Action to Your Workflow Rule

Time-dependent actions are executed at a specific time, such as 10 days before a record's close date. When that specific time passes, the workflow rule reevaluates the record to make sure that it still meets the rule criteria. If the record does, the workflow rule executes those actions.

SEE ALSO:

Identify Your Salesforce Org's Default Workflow User

Set the Criteria for Your Workflow Rule

Add an Immediate Action to Your Workflow Rule

Immediate actions, like their name suggests, are executed as soon as the workflow rule finishes evaluating the record.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:

- Customize Application

❗ Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

For details on each action type, see Automated Actions .

1. Open a workflow rule.
2. In the Immediate Workflow Actions section, click **Add Workflow Action**.
3. Select one of the options to create an action or select an existing one.

SEE ALSO:

Add Automated Actions to Your Workflow Rule

Add a Time-Dependent Action to Your Workflow Rule

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:

- Customize Application

Time-dependent actions are executed at a specific time, such as 10 days before a record's close date. When that specific time passes, the workflow rule reevaluates the record to make sure that it still meets the rule criteria. If the record does, the workflow rule executes those actions.

> Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Time-dependent actions and time triggers are complex features. As you work with time-dependent actions and time triggers, keep in mind their considerations.

If you plan on configuring workflow rules that have time-dependent actions, specify a default workflow user. Salesforce associates the default workflow user with a workflow rule if the user who initiated the rule is no longer active.

1. Open a workflow rule.
2. In the Time-Dependent Workflow Actions section, click **Add Time Trigger**.

   > Note: You can't add a time trigger if:
   >
   > - The evaluation criteria is set to `Evaluate the rule when a record is: created, and every time it's edited`.
   > - The rule is activated.
   > - The rule is deactivated but has pending actions in the workflow queue.

3. Specify a number of days or hours before or after a date that's relevant to the record, such as the date the record was created.

   If the workflow rule is still active and valid when this time occurs, the time trigger fires the workflow action.

4. Save your time trigger.
5. In the section for the time trigger you created, click **Add Workflow Action**.
6. Select one of the options to create an action or select an existing one.
7. Click **Done**.

SEE ALSO:

Add Automated Actions to Your Workflow Rule

Considerations for Time-Dependent Actions and Time Triggers

Identify Your Salesforce Org's Default Workflow User

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Select a `Default Workflow User` that you want Salesforce to display with a workflow rule when the user that triggered the rule isn't active.

User Permissions Needed	
To edit process automation settings:	Customize Application
To create, update, and delete flow list views:	Manage Flow

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

If your organization uses time-dependent actions in workflow rules, you must designate a default workflow user. When the user who triggered the rule isn't active, Salesforce displays the username of the default workflow user in the `Created By` field for tasks, the `Sending User` field for email, and the `Last Modified By` field for field updates. Salesforce doesn't display this username for outbound messages. If a problem occurs with a pending action, the default workflow user receives an email notification.

When workflow email alerts approach or exceed certain limits, Salesforce sends a warning email to the default workflow user or—if the default workflow user isn't set—to an active Salesforce admin.

1. From Setup, enter *`Process Automation Settings`* in the `Quick Find` box, then select **Process Automation Settings**.
2. For `Default Workflow User`, select a user.
3. Save your changes.

SEE ALSO:

Daily Allocations for Email Alerts

Associate Actions with Workflow Rules or Approval Processes

Associate actions that have already been created in your organization with a workflow rule and approval processes.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To select existing actions:
- Customize Application

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. To associate existing workflow actions with a workflow rule:
   a. From Setup, enter *`Workflow Rules`* in the `Quick Find` box, then select **Workflow Rules**.
   b. Select the workflow rule.
   c. Click **Edit** in the Workflow Actions section.
   d. Click **Add Workflow Action** in either the Immediate Workflow Actions or Time-Dependent Actions section, depending on when you want the action to occur, and choose **Select Existing Action**.
   e. Select the type of action to associate with the workflow rule.
   f. Select the actions in the **Available Actions** box and use the right arrow to move them to the **Selected Actions** box. If necessary, select the left arrow to remove actions from the **Available Actions** box.
   g. Save your changes.
2. To associate existing workflow actions with an approval process:
   a. From Setup, enter *`Approval Processes`* in the `Quick Find` box, then select **Approval Processes**.
   b. Click the name of an approval process.
   c. To have the action occur during the initial submission, final approval, final rejection , or recall, click **Add Existing** in the Initial Submission Actions, Final Approval Actions, Final Rejection Actions, or Recall Actions section.

d. To have the action occur during the approval steps, click **Show Actions** in the Approval Steps section, then click **Add Existing** in the Approval, Rejection, or Recall Actions section. See Add an Existing Automated Action to Your Approval Process on page 842

e. Select the type of action you want to associate with the approval process. The **Available Actions** box lists all existing actions of the selected type.

f. Enter the name of a specific action in the text field and click **Find**.

g. Select the actions in the **Available Actions** box that you want to associate with the approval process, and use the right arrow to move the actions to the **Selected Actions** box. If necessary, select the left arrow to remove actions from the **Available Actions** box.

h. Save your changes.

SEE ALSO:

Manage Automated Actions in Workflow Rules

Define a Flow Trigger for Workflow (Pilot)

Create a flow trigger so that you can launch a flow from workflow rules. With flow triggers, you can automate complex business processes—create flows to perform logic, and have events trigger the flows via workflow rules—without writing code. For example, your flow looks up and assigns the relevant entitlement for a case. Create a flow trigger to launch the flow whenever a case is created, so that all new cases are automatically set with a default entitlement.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To view workflow rules and actions:

- View Setup and Configuration

To create or change workflow rules and actions:

- Customize Application

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

To get started using flow triggers, from Setup, enter `Flow Triggers` in the Quick Find box, then select **Flow Triggers**. Before you begin:

- Create and activate the autolaunched flow that you want this workflow action to launch.
- Create the workflow rule that you plan to add this workflow action to.
- Understand the special behavior and limitations of flow triggers. See Flow Trigger Considerations (Pilot) on page 1010.

Complete these steps to create a flow trigger.

1. From Setup, enter `Flow Triggers` in the Quick Find box, then select **Flow Triggers**.
2. Click **New Flow Trigger**.
3. Select the same object as the workflow rule, and then click **Next**.
4. Configure the flow trigger.

Field	Description
`Name`	Name of the flow trigger.
`Unique Name`	Enter a unique name to refer to this component in the API. The **Unique Name** field can contain only underscores and alphanumeric characters. It must be unique within the selected object type,

Field	Description
	begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
`Protected Component`	Reserved for future use.
`Flow`	Unique name of the autolaunched flow that this workflow action launches.
`Set Flow Variables`	Whether to pass values into the flow's variables.

5. If you select `Set Flow Variables`, specify their names and values.

   Click **Set Another Value** to set up to

Field	Description
`Name`	Select the name of the flow variable. Only variables that allow input access can be selected.
`Value`	For a flow variable, you can: • Enter a literal value. • Click 🔍 🔍, select a field, and click **Insert**. For a record variable, you can: • Click 🔍 🔍, select a record, and click **Insert**. To help you distinguish between records and fields, all record options are marked with an asterisk (*) and appear at the top of each list. • To use the current values of the record that was created or edited to cause the workflow rule to fire, enter *`{!this}`*. • To use the most recent previous values of the record that was edited to cause the workflow rule to fire, enter *`{!old}`*. In other words, `{!old}` identifies the same record as `{!this}` but uses the record's values from immediately before it was edited to cause the workflow rule to fire. • If the record was newly created, `{!old}` is `null`. • Unlike `{!this}`, `{!old}` can't be selected by clicking 🔍 🔍. Manually enter *`{!old}`* in the Value column.

6. To put the flow trigger in test mode, select `Administrators run the latest flow version`.

   When selected and an admin triggers the workflow rule, the flow trigger launches the latest version of the flow. For all other users, the flow trigger always launches the active version of the flow.

   The same values are passed into the flow variables whether the flow trigger launches the active or latest flow version.

7. Click **Save**.

Don't forget to associate the flow trigger to a workflow rule.

SEE ALSO:

Flow Trigger Considerations (Pilot)

Activate Your Workflow Rule

Salesforce doesn't trigger a workflow rule until you activate it.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

To activate a workflow rule, click **Activate** on the workflow rule detail page. Click **Deactivate** to prevent a rule from triggering or if you want to edit the time-dependent actions and time triggers that are associated with the rule.

You can deactivate a workflow rule at any time. However, if you deactivate a rule that has pending actions, Salesforce completes those actions as long as the record that triggered the rule isn't updated.

Note:

- You can't delete a workflow rule that has pending actions in the workflow queue. Wait until pending actions are processed, or use the workflow queue to cancel the pending actions.
- You can't add time-dependent workflow actions to active workflow rules. Deactivate the workflow rule first, add the time-dependent workflow action, and reactivate the rule.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

USER PERMISSIONS

To create or change workflow rules and actions:

- Customize Application

SEE ALSO:

Set the Criteria for Your Workflow Rule

Workflow Limits

Salesforce limits the number of total and active rules in your org, the number of time triggers and actions per rule. It also processes a limited number of daily emails and hourly time triggers.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Per-Org Limit	Value
Total rules across objects (Applies to any combination of workflow, assignment, auto-response, and escalation rules, *active* and *inactive*.)	2,000
Total rules per object	500

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Per-Org Limit	Value
(Applies to any combination of workflow, assignment, auto-response, and escalation rules, *active* and *inactive*.)	
Active rules per object (Applies to any combination of *active* workflow, assignment, auto-response, and escalation rules, as well as record change processes.)	50
Time triggers per workflow rule[1]	10
Immediate actions per workflow rule[1]	40
Time-dependent actions per time trigger	40
Workflow time triggers per hour	1,000
Flow trigger workflow actions: flow variable assignments[2]	25 (N/A in Professional Edition)
Combined total of these automations that start or resume based on a record's field value. • Resume events that are defined in active flows • Groups of scheduled actions that are defined in active processes • Time triggers that are defined in active workflow rules • Inactive flow interviews that are resumed	20,000

[1] The immediate actions and each time trigger can have:

[2] The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Daily Allocations for Email Alerts

The daily allocation for emails sent through email alerts is 1,000 per standard user license per org—except for free Developer Edition and trial orgs, where the daily workflow email allocation is 15. The overall org allocation is 2,000,000. This allocation applies to emails sent through email alerts in workflow rules, approval processes, flows, processes, or REST API. Single emails sent to external email addresses are also limited, and how those limits are enforced depends on when your org was created.

Daily Allocations for Email Alerts

The daily allocation for emails sent through email alerts is 1,000 per standard user license per org—except for free Developer Edition and trial orgs, where the daily workflow email allocation is 15. The overall org allocation is 2,000,000. This allocation applies to emails sent through email alerts in workflow rules, approval processes, flows, processes, or REST API. Single emails sent to external email addresses are also limited, and how those limits are enforced depends on when your org was created.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

After your org has reached its daily workflow email allocation:

- Any emails in the workflow queue not sent that day are discarded. Salesforce doesn't try to resend them later.
- If a workflow rule with an action and an email alert is triggered, only the email action is blocked.
- Final approval, final rejection, approval, rejection, and recall email actions are blocked.
- An error message is added to the debug log.

These items don't count against the workflow email allocation:

- Approval notification emails
- Task assignment notifications
- Lead assignment rules notifications
- Case assignment rules notifications
- Case escalation rules notifications
- Salesforce Sites usage alerts

The allocation restriction is based on activity in the 24-hour period starting and ending at midnight GMT. Adding or removing a user license immediately adjusts the allocation's total. If you send an email alert to a group, every recipient in that group counts against your daily workflow email allocation.

Single Email Limits

Each licensed org can send single emails to a maximum of 5,000 external email addresses, or recipients, per day. A day is based on Greenwich Mean Time (GMT).

Sending emails to internal email recipients doesn't count toward the org daily limit.

- For orgs created before Spring '19, the org daily limit is enforced only for emails sent via Apex and Salesforce APIs, except for REST API.
- For orgs created in Spring '19 and later, the org daily limit is also enforced for email alerts, simple email actions, Send Email actions in flows, and REST API.
- Each user can send emails from the email composer to a maximum of 250 external email recipients per hour.

In Developer Edition orgs and orgs evaluating Salesforce during a trial period, each user can send emails to a maximum of 50 recipients per day, and each single email can have up to 15 recipients.

Allocation Alerts

When workflow email alerts approach or exceed certain allocations, Salesforce sends a warning email to the default workflow user or—if the default workflow user isn't set—to an active Salesforce admin.

When...	Salesforce Sends...	Warning Email Includes...
An email alert isn't sent because the number of recipients exceeds the allocation for a single email	A warning email for each unsent email alert	The unsent email alert's content and recipients
The org reaches 90% of the allocation of emails per day	One warning email	The allocation and the org's usage
The org reaches 90% of the allocation of workflow emails per day	One warning email	The allocation and the org's usage
An email alert isn't sent because the org reaches the allocation of emails per day	A warning email after every 100 attempted email alerts over the allocation	The allocation and the org's usage
An email alert isn't sent because the org reaches the allocation of workflow emails per day	A warning email after every 100 attempted email alerts over the allocation	The allocation and the org's usage
The org reaches the daily allocation for single emails sent to external email addresses	One warning email	The allocation and the org that exceeded the allocation

SEE ALSO:

Salesforce Help: Standard User Licenses

Workflow Considerations

Learn the intricacies of workflow rules and workflow actions before you begin working with them.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

- Each workflow rule applies to a single object.
- If you have workflow rules on converted leads and want to use cross-object field updates on the resulting accounts and opportunities, you must enable the lead setting `Require Validation for Converted Leads`.
- If the custom object is deleted, workflow rules on custom objects are automatically deleted.
- The order that individual actions and types of actions are executed in isn't guaranteed. Field update actions are executed first, followed by other actions.
- To create workflow rules that update case fields based on new case comments or incoming email messages, choose Case Comment or Email Message from the `Select Object` dropdown list. Email Message is only available if Email-to-Case or On-Demand Email-to-Case is enabled. You can only create email message workflow rules for field updates, and case comment workflow rules

for field updates, email alerts, and outbound messages. For example, you can create a workflow rule so that an email marked as `Is Incoming` changes its case's `Status` from Closed to New.

- Changes you make to records while using Connect Offline are evaluated by workflow rules when you synchronize.
- Salesforce processes rules in this order.
  - Validation rules
  - Assignment rules
  - Auto-response rules
  - Workflow rules (with immediate actions)
  - Escalation rules
- If a lookup field references a record that is deleted, Salesforce clears the value of the lookup field by default. Or you can choose to prevent record deletions if they're in a lookup relationship.
- If you create workflow rules to replace any Apex triggers, make sure to delete those Apex triggers when you activate the equivalent workflow rules. Otherwise, Apex triggers and workflow rules both fire and cause unexpected results, such as overwritten field updates or redundant email messages.
- When an Account record's owner field is changed, processes and workflows defined on the child object don't get triggered to run.

When Do Workflow Rules Get Triggered?

- Workflow rules can be triggered any time a record is saved or created, depending on your rule criteria. Rules created after saving records don't affect those records retroactively.
- Workflow rules are triggered when a standard or custom object in a master-detail or lookup relationship is reparented, even if the object's evaluation criteria is set to `Evaluate the rule when a record is: created, and any time it's edited to subsequently meet criteria`.
- Saving or creating records can trigger more than one rule.
- Workflow rules only trigger on converted leads if validation and triggers for lead convert are enabled in your Salesforce org.
- Workflow rules trigger automatically and are invisible to the user. Alternatively, approval processes allow users to submit records for approval.
- If your organization uses multiple languages, enter filter values in your individual language. You can add up to 25 filter criteria, of up to 255 characters each.

  When you use picklists to specify filter criteria, the selected values are stored in your org's default language. If you edit or clone existing filter criteria, first set the `Default Language` on the Company Information page to the same language that was used to set the original filter criteria. Otherwise, the filter criteria no longer matches your picklist values and returns inaccurate results.
- If you use record types in your workflow rule criteria whose labels have been translated using the translation workbench, the translated label value doesn't trigger the workflow rule. Workflow criteria evaluate the primary label value and ignore the translated value. To avoid this problem, set the workflow criteria to evaluate the main record type label value by entering it manually in the `Value` field.
- Campaign statistic fields, such as individual campaign statistics and campaign hierarchy statistics, can't trigger workflow rules.
- If its condition references a field that doesn't have a value, a workflow rule isn't triggered. For example, if a User-based workflow rule checks "Role not equal to CEO", the rule isn't triggered for a user without an assigned role. Instead of conditions, use a formula to check that the field is either null or set to something other than "CEO":

  ```
  UserRoleId == null || UserRole.Name != "CEO"
  ```
- The following actions don't trigger workflow rules.

  - Mass replacing picklist values
  - Using the option to replace a picklist value while deleting the current value.
  - Mass updating address fields
  - Mass updating divisions
  - Changing the territory assignments of accounts and opportunities
  - Converting leads to person accounts
  - Deactivating Self-Service Portal, Customer Portal, or Partner Portal users
  - Converting state, country, and territory data from the State and Country/Territory Picklists page in Setup
  - Changing state and country/territory picklists using AddressSettings in the Metadata API

Workflow Rule Limitations

- You can't package workflow rules with time triggers.
- You can't create outbound messages for workflow rules on junction objects.

Tip: Use the Developer Console to debug workflow rules. The Developer Console lets you view debug log details and information about workflow rules and actions. For example, you can view the name of the user who triggered the workflow rule and the name and ID of the record being evaluated.

Workflow for the User Object (Beta)
You can create workflow rules and actions for the User object. You can, for example, send welcome emails to new employees or sync user data with a third-party service using outbound message actions.

Considerations for Time-Dependent Actions and Time Triggers
When creating time-dependent actions and time triggers for workflow rules, consider these factors.

Flow Trigger Considerations (Pilot)
Flow trigger workflow actions have special behaviors and limitations.

SEE ALSO:

Set the Criteria for Your Workflow Rule

Workflow for the User Object (Beta)

You can create workflow rules and actions for the User object. You can, for example, send welcome emails to new employees or sync user data with a third-party service using outbound message actions.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Note: This release contains a beta version of workflow on the User object that is production quality but has known limitations. To provide feedback and suggestions, go to IdeaExchange.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Example Use Cases

For the User object, you can set up workflow rules to:

- Send welcome email messages with training resources to newly created users by using email alert actions.
- Send emails when users change roles or are deactivated by using email alert actions.
- Deactivate temporary employees after a specified period by using field update actions.
- Sync user data with third-party systems by using outbound messages actions.

Merge Field Types for the User Object

To use merge fields from user records in email templates, select from the following merge field types:

- User Fields—Use these merge fields to represent the sending user. Merge fields named {!User.`field_name`} return values from the user record of the person who created or updated the record that triggered the workflow rule.
- Workflow Target User Fields—Use these merge fields only in email templates for workflow rules on the User object. Merge fields named {!Target_User.`field_name`} return values from the user record that was created or updated to trigger the workflow rule.

Beta Limitations for Workflow on the User Object

Understand these limitations before you create workflow rules or workflow actions for the User object.

- Tasks aren't supported as workflow actions for the User object.
- When setting the workflow rule criteria, you can't select `Current User` fields using the picklists. You can, however, use a formula to set the rule criteria and include fields from the current user. In the formula editor, click **Insert Field**, select `$User`, select the field, and click **Insert**.
- Remember that custom validation rules run *before* workflow rules are executed. Refer to Triggers and Order of Execution in the *Apex Developer Guide*.

SEE ALSO:

Workflow Considerations

Considerations for Time-Dependent Actions and Time Triggers

When creating time-dependent actions and time triggers for workflow rules, consider these factors.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Defining Time Triggers

- When defining a time trigger, use standard and custom date and date/time fields defined for the object. Specify time using days and hours. The valid range is 0–999 days or hours.
- You can modify existing time triggers by adding or removing actions.

  Note: Removing all the actions from a time trigger doesn't remove the trigger. Empty triggers are still queued and count against your hourly workflow time trigger limit. To remove scheduled time triggers, delete them from the workflow queue.

Time Trigger Processing

- Time-dependent actions aren't executed independently. They're grouped into a single batch that starts executing within one hour after the first action enters the batch.

  Note: Actual execution can be delayed based on service availability.

- Apex triggers that fire as a result of time-dependent actions can get executed in a single batch or independently. Follow these best practices:
  - In case they fire independently–Ensure that your Apex logic is scoped for a single scheduled action. For example, don't use Apex static variables to communicate state across Apex code triggered by different scheduled actions.
  - In case they fire in a single batch, be aware of how the combination of your time-dependent actions and Apex triggers impacts your Apex governor limits.
- Salesforce evaluates time-based workflow on the organization's time zone, not the user's. Users in different time zones can see differences in behavior.
- Salesforce doesn't necessarily execute time triggers in the order they appear on the workflow rule detail page. Workflow rules list time triggers that use the `Before` field first, followed by time triggers that use the `After` field.
- If you set the workflow rule evaluation criteria to `Evaluate the rule when created, and every time it's edited`, Salesforce doesn't display time-dependent action controls on the workflow rule edit page.
- If you change a date field that is referenced by an unfired time trigger in a workflow rule that has been evaluated, Salesforce recalculates the unfired time triggers associated with the rule. For example, if a workflow rule is scheduled to alert the opportunity owner 7 days before the opportunity close date, and the close date is set to 2/20/2011, Salesforce sends the alert on 2/13/2011. If the close date is updated to 2/10/2011 and the time trigger hasn't fired, Salesforce reschedules the alert for 2/3/2011. If Salesforce recalculates the time triggers to a date in the past, Salesforce triggers the associated actions shortly after you save the record.
- If a workflow rule has a time trigger set for a time in the past, Salesforce queues the associated time-dependent actions to start executing within one hour. For example, if a workflow rule on opportunities is configured to update a field 7 days before the close date, and you create an opportunity record with the close date set to today, Salesforce starts to process the field update within an hour after you create the opportunity.
- Time-dependent actions remain in the workflow queue only as long as the workflow rule criteria are still valid. If a record no longer matches the rule criteria, Salesforce removes the time-dependent actions queued for that record.

  For example, an opportunity workflow rule can specify:
  - A criteria set to "Opportunity: Status not equals to Closed Won, Closed Lost"
  - An associated time-dependent action with a time trigger set to 7 days before the opportunity close date

  If a record that matches the criteria is created on July 1 and the `Close Date` is set to July 30, the time-dependent action is scheduled for July 23. However, if the opportunity is set to "Closed Won" or "Closed Lost" before July 23, the time-dependent action is removed from the queue.

- Salesforce ignores time triggers that reference null fields.
- If the record is updated and the evaluation criteria is set to `Evaluate the rule when a record is: created, and any time it's edited to subsequently meet criteria`, time-dependent actions can automatically be queued again. Using the previous example, if the opportunity status is changed from Closed Lost to Prospecting and the workflow rule evaluation criteria is `Evaluate the rule when a record is: created, and any time it's edited to subsequently meet criteria`, Salesforce reevaluates the time triggers and adds the appropriate actions to the workflow queue.
- Deleting a record that has pending actions removes the pending actions from the workflow queue. You can't restore the actions, even if you undelete the record.

- If the evaluation criteria is set to `Evaluate the rule when a record is: created`, the workflow rule evaluates its time triggers only one time. If the record that fired the rule changes to no longer meet the evaluation criteria, Salesforce removes the pending actions from the queue and never reapplies the rule to the record.
- You can deactivate a workflow rule at any time. If the rule has pending actions in the workflow queue, editing the record that triggered the rule removes the pending actions from the queue. If you don't edit the record, the pending actions are processed even though the rule has been deactivated.
- Time-dependent actions aren't executed for a reevaluated workflow rule in the following situations:
  - The reevaluated workflow rule's immediate actions cause the record to no longer meet the workflow rule criteria.
  - An Apex `after` trigger that is executed as a result of a workflow or approvals action causes the record to no longer meet the workflow rule criteria.
- Configuring a task's `Due Date` to "Rule Trigger Date" sets time triggers and workflow task due dates based on the date that the workflow time trigger's action is executed. For example, if the task due date is "Rule Trigger Date plus 10 days" and the time trigger is executed on January 1, Salesforce sets the task due date to January 11.
- You can add a new active workflow rule with time triggers in a change set and deploy it. You can only change time triggers on a workflow rule in a change set if it's inactive. The rule must be activated in the destination organization manually or through another change set that only activates workflow rules and makes no time trigger changes.

  For example, let's say you have an inactive workflow rule in your destination organization, and your change set contains an active workflow rule with the same name and new or different time triggers. The deployment fails because it activates the workflow rule first and then tries to add or remove the time triggers.

  Note: You must add time-dependent actions manually when including a workflow rule in a change set. The **View/Add Dependencies** function doesn't detect time-dependent actions.

Using Time-Dependent Workflow with Leads

- You can't convert a lead that has pending actions.
- If Validation and Triggers from Lead Convert is enabled, existing time-based workflow actions on leads aren't triggered during lead conversion.
- If a campaign member based on a lead is converted before the completion of the time-based workflow actions associated with it, Salesforce still performs the time-based workflow actions.

Limitations

- Time triggers don't support minutes or seconds.
- Time triggers can't reference the following:
  - `DATE` or `DATETIME` fields containing automatically derived functions, such as `TODAY` or `NOW`.
  - Formula fields that include related-object merge fields.
- Salesforce limits the number of time triggers an organization can execute per hour. If an organization exceeds the limits for its Edition, Salesforce defers the execution of the additional time triggers to the next hour. For example, if an Unlimited Edition organization has 1,200 time triggers scheduled to execute between 4:00 PM and 5:00 PM, Salesforce processes 1,000 time triggers between 4:00 PM and 5:00 PM and the remaining 200 time triggers between 5:00 PM and 6:00 PM.
- You can't archive a product or price book that has pending actions.
- If time-based workflow actions exist in the queue, you can't add or remove time triggers or edit trigger dates without deleting the actions first. Because the deleted records can't be restored, carefully consider the implications of editing the workflow rules before you proceed. If you decide to edit the workflow rules, deactivate the workflow that you want to edit, edit the rules as needed, and

then save your changes. For information about finding and deleting time-based workflow actions in the queue, see Monitor Pending Workflow Actions on page 1021.

You also can't add or remove time triggers if:

- The workflow rule is active.
- The workflow rule is deactivated, but has pending actions in the queue.
- The workflow rule evaluation criteria is set to `Evaluate the rule when a record is: created, and every time it's edited`.
- The workflow rule is included in a package.

SEE ALSO:

Add Automated Actions to Your Workflow Rule

Identify Your Salesforce Org's Default Workflow User

Flow Trigger Considerations (Pilot)

Flow trigger workflow actions have special behaviors and limitations.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Understand these considerations before you create flow triggers or add them to workflow rules.

- Flow triggers are available only for workflow rules. You can't use them as actions elsewhere, for example, in approval processes.
- Flow triggers are available on most—but not all—objects that are supported by workflow rules. You can see the list of supported objects when you create a flow trigger. From Setup, enter *`Flow Triggers`* in the `Quick Find` box, then click **Flow Triggers**.
- Only active, autolaunched flows can be launched by flow triggers. However, if a flow trigger is in test mode, admins run the latest flow version while other users run the active flow version.
- Flows that are launched from workflow rules are run in system context, which means that user permissions, field-level security, and sharing rules aren't considered during flow execution.
- If a flow trigger fails at run time, the user who created or edited the record to meet the workflow rule criteria isn't able to save the record. To troubleshoot run time issues, see the flow action events in the `Workflow` category of debug logs, which show the flow version and the values passed into flow variables.
- A flow trigger can set the values of up to 25 variables in the flow, with the following limitations.
  - Flow triggers can't use multi-select picklist fields to set flow variables.
  - When a flow trigger uses a currency field to set a flow variable, only the amount is passed into the flow. Any currency ISO code or locale information is ignored. If your organization uses multiple currencies, the flow trigger uses the amount in the currency of the record that contains the specified currency field.
  - Flow triggers can't pass values into record collection variables in flows.
- Always keep one version of the flow active if it's referenced by an active workflow rule's flow trigger.
- After you activate a workflow rule using the flow trigger, don't modify or add a version of the flow to include screens or other elements that violate the run restrictions for an autolaunched flow. If you modify a flow to no longer autolaunch, it can't be launched by flow triggers. To work around this situation, you can save the non-autolaunched flow as a new flow and change the new flow to become autolaunched. Then update the flow triggers to launch the new flow.

- Flow triggers aren't available as time-dependent workflow actions. You can add flow triggers to workflow rules only as immediate workflow actions.
- When the system executes a workflow rule with multiple flow triggers, those flows aren't run in any particular order.
- In a transaction, flow triggers are executed after all workflow field updates, including any Apex triggers and standard validations that are executed as a result of those workflow field updates. After executing flow triggers, the system executes escalation rules.
- Flows that are launched from workflow rules are governed by the per transaction limits already enforced by Apex.
- When flows are launched from workflow rules that are triggered by bulk loads or imports, the flows' data manipulation language (DML) operations are executed in bulk to reduce the number of calls required and to optimize system performance. The execution of any of the following flow elements qualifies as a DML operation: Create Records, Update Records, or Delete Records.

  For example, suppose that you use Data Loader or the Bulk API to update 50 records, and those updates meet the criteria of a workflow rule with a flow trigger action. In response, the system executes 50 instances of the flow within the same transaction. Each instance of a running flow is called an interview. The system attempts to execute each DML operation across all the interviews in the transaction at the same time. Suppose that five of those interviews are executing the same branch of the flow, which has an Update Records element called "SetEntitlement." The system waits for all five interviews to reach that element, and then executes all five record updates in bulk.
- Flow triggers aren't available in change sets.
- Flow triggers aren't packageable.

Workflow Rule Examples

Looking for ideas on how workflow rules can help streamline your business? Check out these examples.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

Important: Where possible, we changed noninclusive terms to align with our company value of Equality. We maintained certain terms to avoid any effect on customer implementations.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

- Business Processes
  - Follow Up Before Contract Expires
  - Follow Up when Platinum Contract Case Closes
  - Assign Credit Check for New Customer
  - Notify Account Owner About New, High-Priority Cases
  - Set a Default Entitlement for Each New Case
  - Update Shipment Status if Shipment is Delayed
  - Automatically Activate New Users
- Cross-Object Processes
  - Notify Sales VP About Cases Filed for Top Accounts
  - Set Default Opportunity Name
  - Set Target Resolution Date for Cases
  - Update Application Record when Candidate Accepts Job

- Deal Management
  - Track Closed Opportunities
  - Override Default Opportunity Close Date
  - Report Lost Opportunities
  - Report Unassigned Leads
  - Send Alert if Quote Line Item Discount Exceeds 40%
- Notifications
  - Notify Key People About Account Owner Changes
  - Set Reminder for Contact Birthday
  - Set Reminder for High-Value Opportunity Close Date
  - Notify Account Owner of Updates by Others

Follow Up Before a Contract Expires

Object	Contract
Description	Email a reminder to the renewal manager 20 days before a contract's end date.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(Contract: Status equals Activated)`
Immediate Actions	None
Time-Dependent Actions	20 Days Before Contract: End Date—`Email Alert:` Email a reminder to the renewal manager to confirm whether the client wants an extension.

Follow Up When a Platinum Contract Case Closes

This example assumes that a `Contract Type` custom picklist is used to identify the contract level on cases and that the picklist contains the Platinum value.

Object	Case
Description	If the customer has a platinum contract agreement, email a feedback request to the case contact 7 days after a high-priority case has been closed.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria

Rule Criteria (Filter)	Run this rule if the following criteria are met.
	`(Case: Priority equals High) and`
	`(Case: Closed equals True) and`
	`(Case: Contract Type equals Platinum)`

Immediate Actions	None

Time-Dependent Actions	7 Days After Case: Date/Time Closed—`Email Alert:` Email a feedback request to the case contact.

Assign Credit Check for a New Customer

This example assumes that a `New Customer` custom field is on opportunities.

Object	Opportunity
Description	Assign the Accounts Receivable (AR) department a task to check the credit of a potential customer 15 days before the opportunity close date if the amount is greater than $50,000.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria

Rule Criteria (Filter)	Run this rule if the following criteria are met.
	`(Opportunity: Amount greater than 50000) and`
	`(Opportunity: Closed equals False) and`
	`(Opportunity: New Customer equals True)`

Immediate Actions	None
Time-Dependent Actions	15 Days Before Opportunity: Close Date—`Task:` Create a task for users in the Accounts Receivable role to run a credit check.

Notify Account Owner About New, High-Priority Cases

This example assumes that a Service Level Agreement custom picklist called SLA identifies the agreement level on accounts and contains the Platinum value.

Object	Case
Description	Notify the account owner when a high-priority case is created for accounts with a platinum SLA.
Evaluation Criteria	Evaluate the rule when a record is: created

Rule Criteria (Filter)	Run this rule if the following criteria are met.
	`(Case: Priority equals High) and`
	`(Account: SLA equals Platinum)`

Immediate Actions	`Email Alert:` Email the details of the high-priority case to the account owner.
Time-Dependent Actions	None

Set a Default Entitlement for Each New Case

This example assumes that an active, autolaunched flow looks up the relevant entitlement based on the account, asset, or contact associated with the new case and updates the case with the entitlement name.

The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Object	Case
Description	Set a default entitlement on each new case.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(Case: Status not equal to Closed)`
Immediate Actions	`Flow Trigger:` Look up and assign the relevant entitlement to the case. Pass the account, asset, or contact associated with the new case into the relevant flow variable to enable the entitlement lookup. Pass the case ID into the relevant flow variable to enable the case update.
Time-Dependent Actions	None.

Update Shipment Status If Shipment Is Delayed

Object	Shipment
Description	Update the `Shipment Status` field to Delayed if a shipment has exceeded the expected delivery date and hasn't reached the customer.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(Shipment: Status not equal to Delivered)`
Immediate Actions	None
Time-Dependent Actions	1 day after Shipment: Expected Delivery Date—`Field Update`: Change `Shipment Status` field to Delayed on Shipment record.

Automatically Activate New Users

Object	User
Description	Make sure that each new user is active so that the user can log in to Salesforce.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(User: Active equals False)`
Immediate Actions	`Field Update`: Set `Active` to True.
Time-Dependent Actions	None.

Notify Sales VP About Cases Filed for Top Accounts

This workflow rule is for sales VPs who want to know about cases filed for top accounts. Top accounts are determined by size and revenue.

Object	Case
Description	Notify sales VP about cases filed for top accounts.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following criteria are met. `AND(Account.AnnualRevenue > 500000, Account.NumberOfEmployees > 5000)`
Immediate Actions	`Email Alert:` Notify VP about cases for large accounts.
Time-Dependent Actions	None

Set Default Opportunity Name

The opportunity naming convention for some companies is *`Account Name: Opportunity Name`*. To automate the default name of each opportunity in your org, create the following workflow rule.

Object	Opportunity
Description	Enforce opportunity naming convention.
Evaluation Criteria	Evaluate the rule when a record is: created, and every time it's edited
Rule Criteria (Filter)	Run this rule if the following criteria are met. `NOT(CONTAINS( Name, Account.Name ))`
Immediate Actions	`Field Update`: Set opportunity name to the following formula. `Account.Name & ": " & Name`
Time-Dependent Actions	None

Set Target Resolution Date for Cases

This example sets a case resolution date based on the value of a field on the associated account. It uses a custom picklist field on accounts called `Support Level`, which has three values: Basic, Standard, and Premium. It also has a custom date field on cases called `Target Resolution Date`.

Use the following three workflow rule examples to set the target resolution date of a case based on the support level for the related account.

Set Resolution Date for Basic Support

Object	Case
Description	Set the case target resolution date for accounts that have basic support level to 30 days from today.

Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following formula is true. `ISPICKVAL(Account.Support_Level__c , "Basic")`
Immediate Actions	`Field Update`: Set the `Target Resolution Date` to Today() + 30.
Time-Dependent Actions	None

Set Resolution Date for Standard Support

Object	Case
Description	Set the case target resolution date for accounts that have standard support level to 14 days from today.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following formula is true. `ISPICKVAL(Account.Support_Level__c , "Standard")`
Immediate Actions	`Field Update`: Set the `Target Resolution Date` to Today() + 14.
Time-Dependent Actions	None

Set Resolution Date for Premium Support

Object	Case
Description	Set the case target resolution date for accounts that have premium support level to 5 days from today.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following formula is true. `ISPICKVAL(Account.Support_Level__c , "Premium")`
Immediate Actions	`Field Update`: Set the `Target Resolution Date` to Today() + 5.
Time-Dependent Actions	None

Update Application Record When Candidate Accepts Job

This workflow rule closes the Application record when a candidate accepts the job. Cross-object field updates to the main record are supported between custom objects in a main detail relationship.

Object	Candidate
Description	Change the `Application Status` field to Closed for the custom Application object when the `Candidate Status` field for the custom Candidate object changes to Accepted.

Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(Candidate: Status equals Accepted)`
Immediate Actions	`Field Update:` Change the `Application Status` field to Closed on parent Application record.
Time-Dependent Actions	None

Track Closed Opportunities

This example assumes that a Closed Opportunities record type provides additional information to certain profiles. For information on record types, see Tailor Business Processes to Different Record Types Users.

Object	Opportunity
Description	Change the record type of closed-won opportunities.
Evaluation Criteria	Evaluate the rule when a record is: created, and every time it's edited

Rule Criteria (Filter)	Run this rule if the following criteria are met.
	`(Opportunity: Closed equals True) and`
	`(Opportunity: Stage equals Closed Won)`

Immediate Actions	`Field Update:` Set the record type to Closed Opportunities.
Time-Dependent Actions	None

Override the Default Opportunity Close Date

Object	Opportunity
Description	Override the default close date from the close of the quarter to 6 months after the opportunity is created.
Evaluation Criteria	Evaluate the rule when a record is: created
Rule Criteria (Filter)	Run this rule if the following criteria are met. `(Opportunity: Closed equals False)`
Immediate Actions	`Field Update:` Use the following formula to set the opportunity close date to 6 months after the creation date. `DATE( YEAR(TODAY()) , (MONTH(TODAY()) + 6), DAY(TODAY()))`
Time-Dependent Actions	None

Report Lost Opportunities

Object	Opportunity
Description	Notify the VP of sales when a deal is lost if the stage was Proposal/Price Quote and the amount was greater than $1 million.
Evaluation Criteria	Evaluate the rule when a record is: created, and every time it's edited
Rule Criteria (Filter)	Run this rule if the following formula is true. `AND( ISCHANGED(StageName), ISPICKVAL(PRIORVALUE(StageName) , "Proposal/Price Quote"), ISPICKVAL(StageName,"Closed Lost"), (Amount >1000000))`
Immediate Actions	`Email Alert:` Notify the VP of sales role that the deal was lost.
Time-Dependent Actions	None

Report Unassigned Leads

This example assumes that all unassigned leads are placed in an unassigned leads queue by a leads assignment rule.

Object	Lead
Description	Ensure that unassigned leads are tracked in a timely manner by notifying the manager if a lead isn't accepted in 2 days.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria
Rule Criteria (Filter)	Run this rule if the following criteria are met. `Lead Owner equals Unassigned Lead Queue`
Immediate Actions	None
Time-Dependent Actions	2 Days After Lead: Last Modified Date—`Email Alert:` Notify the manager role that the queue has unassigned leads that are older than 2 days.

Send Alert If Quote Line Item Discount Exceeds 40%

Object	Quote Line Item
Description	Ensure that an email alert is sent if a sales rep applies a quote line item discount that exceeds 40%.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria
Rule Criteria (Filter)	Run this rule if the following criteria are met. `Quote Line Item: Discount is greater than 40`
Immediate Actions	`Email Alert:` Notify the manager role that the quote line item discount exceeds 40%.

Time-Dependent Actions	None

Notify Key People About Account Owner Changes

Object	Account
Description	Notify key people in the sales department when the owner of an account changes if the account's annual revenue is greater than $1 million.
Evaluation Criteria	Evaluate the rule when a record is: created, and every time it's edited
Rule Criteria (Filter)	Run this rule if the following formula is true. `AND( ISCHANGED(OwnerId), AnnualRevenue > 1000000 )`
Immediate Actions	`Email Alert:` Notify the person in the sales operations role of the change in account ownership.
Time-Dependent Actions	None

Set Reminder for Contact Birthday

This example assumes that a `Next Birthday` custom formula field uses the following formula to calculate the date of the contact's next birthday on contact records.

```
IF(MONTH(Birthdate) > MONTH(TODAY()),DATE(YEAR(TODAY()),MONTH(Birthdate),DAY(Birthdate)),
IF(MONTH(Birthdate) < MONTH(TODAY()),DATE(YEAR(TODAY())+1,MONTH(Birthdate),DAY(Birthdate)),
IF(DAY(Birthdate) >= (DAY(TODAY())),DATE(YEAR(TODAY()),MONTH(Birthdate),DAY(Birthdate)),
DATE(YEAR(TODAY())+1,MONTH(Birthdate),DAY(Birthdate)))))
```

Object	Contact
Description	Send an email to the contact 2 days before the contact's birthday.
Evaluation Criteria	Evaluate the rule when a record is: created

Rule Criteria (Filter)	Run this rule if the following formula is true.
	`(Contact: Birthdate not equal to null) and`
	`(Contact: Email not equal to null)`

Immediate Actions	None
Time-Dependent Actions	2 Days Before Contact: Next Birthday—`Email Alert:` Send a birthday greeting to the contact's email address.

Set Reminder for High-Value Opportunity Close Date

Object	Opportunity

Description	Remind the opportunity owner and senior management when the close date is approaching for an opportunity that has an amount greater than $100,000. Create a follow-up task for the opportunity owner if the deal is still open when the close date passes.
Evaluation Criteria	Evaluate the rule when a record is: created, and anytime it's edited to subsequently meet criteria

Rule Criteria (Filter)	Run this rule if the following criteria are met.
	`(Opportunity: Amount greater than 100000) and`
	`(Opportunity: Closed equals False)`

Immediate Actions	None
Time-Dependent Actions	• 30 Days Before Opportunity: Close Date—`Email Alert:` Notify the opportunity owner that 30 days remain. • 15 Days Before Opportunity: Close Date—`Email Alert:` Notify the opportunity owner that 15 days remain. • 5 Days After Opportunity: Close Date—`Task:` Create a follow-up task for the opportunity owner to update the deal. `Email Alert:` Notify senior management to involve executives.

Notify Account Owner of Updates by Others

Object	Account
Description	Notify the account owner when someone else updates the account if the account's annual revenue is greater than $1 million.
Evaluation Criteria	Evaluate the rule when a record is: created, and every time it's edited
Rule Criteria (Filter)	Run this rule if the following formula is true. `AND( (LastModifiedById <> OwnerId), (AnnualRevenue > 1000000) )`
Immediate Actions	`Email Alert:` Notify the account owner that someone else has updated the account.
Time-Dependent Actions	None

SEE ALSO:

Workflow Rules

Set the Criteria for Your Workflow Rule

Monitor Pending Workflow Actions

When a workflow rule that has time-dependent actions is triggered, use the workflow queue to view pending actions and cancel them if necessary.

EDITIONS

Available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, **Developer**, and **Database.com** Editions

Workflow tasks and email alerts aren't available in **Database.com**

USER PERMISSIONS

To manage the workflow queue:

- Modify All Data

❗ Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

1. From Setup, enter *`Time-Based Workflow`* in the `Quick Find` box, then select **Time-Based Workflow**.
2. To view all pending actions for any active workflow rules, click **Search**. Or to view only the pending actions that match the criteria, set the filter criteria and click **Search**.

   The filter options are:

   - **Workflow Rule Name**: The name of the workflow rule.
   - **Object**: The object that triggered the workflow rule. Enter the object name in the singular form.
   - **Scheduled Date**: The date the pending actions are scheduled to occur.
   - **Create Date**: The date the record that triggered the workflow was created.
   - **Created By**: The user who created the record that triggered the workflow rule.
   - **Record Name**: The name of the record that triggered the workflow rule.

   The filter isn't case-sensitive.

To cancel pending actions:

- Select the box next to the pending actions you want to cancel.
- Click **Delete**.

Workflow Terminology

These terms are used when describing workflow features and functionality.

Important: Starting in Winter '23, you can't create new workflow rules. You can still activate, deactivate, and edit any existing workflow rules. To migrate existing workflow rules, use the Migrate to Flow tool on page 894. For new automations, create flows in Flow Builder on page 16.

EDITIONS

Available in: both Lightning Experience and Salesforce Classic

Flow triggers are available in: Salesforce Classic

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Outbound messages available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Email alerts are available in: **Essentials**, **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Workflow Rule

A workflow rule sets workflow actions into motion when its designated conditions are met. You can configure workflow actions to execute immediately when a record meets the conditions in your workflow rule, or set time triggers that execute the workflow actions on a specific day. If a workflow action hasn't executed yet, you can view and modify it in the workflow queue.

Workflow Action

A workflow action, such as an email alert, field update, outbound message, or task, fires when the conditions of a workflow rule are met.

Email Alert

Email alerts are actions that send emails, using a specified email template, to specified recipients. Workflow alerts can be sent to any user or contact, as long as they have a valid email address.

Field Update

A field update is an action that automatically updates a field with a new value.

Flow

A *flow* is an application that can execute logic, interact with the Salesforce database, call Apex classes, and collect data from users. You can build flows by using Flow Builder.

Flow Trigger

A *flow trigger* is a workflow action that launches a flow. With flow triggers, you can automate complex business processes—create flows to perform logic, and have events trigger the flows via workflow rules—without writing code.

The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use Flow Builder to create a record-triggered flow, or use Process Builder to launch a flow from a process.

Outbound Message

An outbound message sends information to a designated endpoint, like an external service. Outbound messages are configured from Setup. You must configure the external endpoint and create a listener for the messages using SOAP API.

INDEX

E

Einstein Next Best Action, NBA 766
Einstein Next Best Action, NBA, Strategy Builder 749
Einstein Next Best Action, Strategy Builder, Troubleshoot 791

F

Flow
- delivering to users 186
- delivering to users, external 216
- delivering to users, internal 196
- launching from processes 219
- process action 219
- sharing 186, 196

N

NBA, Einstein Next Best Action, strategy builder, elements 793
NBA, Einstein Next Best Action, Strategy Builder, Expressions 782
nba, einstein next best action, strategy builder, recommendations 752–753, 758, 776, 779
Next Best Action, Setup, Implementation 750
Next Best Action, Strategy Builder, Action Strategies 781
Next Best Action, Strategy Builder, Manage Strategies 789
Next Best Action, Strategy Builder, Platform Status Alert Event 787
Next Best Action, Strategy Builder, Tour the Interface 779

S

Strategy Builder Branch Merge Element; Next Best Action 805
Strategy Builder Branch Selector Element; Next Best Action 805
Strategy Builder Enhance Element; Next Best Action 794
Strategy Builder Filter Element; Next Best Action 801
Strategy Builder First Non-empty Branch Element; Next Best Action 806
Strategy Builder Generate Element; Next Best Action 797
Strategy Builder Limit Reoffers Element; Next Best Action 802
Strategy Builder Load Element; Next Best Action 800
Strategy Builder Map Element; Next Best Action 803
Strategy Builder Sort Element; Next Best Action 804

V

Voice
- create permission set 199