Visual Workflow lets you automate business processes by building flows and distributing them to the right users or systems. 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 the Cloud Flow Designer.

For example, use Visual Workflow to script calls for a customer support center or to generate real-time quotes for a sales organization.

You can also build autolaunched flows. An autolaunched 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.

Visual Workflow has three different aspects.

- **Design**: Create flows using the Cloud Flow Designer. This drag-and-drop tool lets you diagram flow structure and configure how it runs, without writing any code.
- **Manage**: Once you create a flow, you can manage it in Salesforce, edit its properties, activate, deactivate, delete, or run it at will.
- **Run**: Flow users can then run the active flow from a custom button, tab, link, or directly from the flow URL. If you’ve enabled them to do so, systems can run active flows through an Apex class or a flow trigger workflow action.

**IN THIS SECTION:**

- Flow Building Blocks
- Use combinations of elements, connectors, and resources to build flows.

- What’s the Difference Between Workflow and Visual Workflow?
- Despite their similar names, Workflow and Visual Workflow are separate Salesforce features for automating business processes.

- Visual Workflow Limits and Considerations
- When designing, managing, and running flows, consider the permissions, use limits, and data issues.

**SEE ALSO:**

- Flow Building Blocks
• 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

What’s the Difference Between Workflow and Visual Workflow?

Despite their similar names, Workflow and Visual Workflow are separate Salesforce features for automating business processes.

Workflow
Workflow enables you to set up workflow rules. A workflow rule identifies what kinds of record changes or additions trigger specified workflow actions, such as sending email alerts and updating record fields.

Workflow rules and actions are associated with a specific object (and can cross objects only to update fields on a related master record).

Visual Workflow
Visual Workflow enables you to create flows, which are triggered by users rather than events. Unlike Workflow, which always executes rules and actions behind the scenes, Visual Workflow offers screens for displaying and collecting information from the user running the flow.

Note: You can, however, participate in a pilot program that enables workflow rules to execute flows behind the scenes. The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are
using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

Flows aren’t tied to any one object. They can look up, create, update, and delete records for multiple objects.

In case you’re wondering, the “visual” in Visual Workflow refers to the ability to visually build flows using the Cloud Flow Designer. Its drag-and-drop user interface lets you build flows without writing any code.

SEE ALSO:
Visual Workflow Overview

Visual Workflow Limits and Considerations

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

IN THIS SECTION:

General Flow Limits
When using Visual Workflow, keep flow limits and Apex governor limits in mind.

Flow Design Considerations
When you design flows, keep certain guidelines in mind.

Flow Administration Considerations
When managing flows, consider the administration and activation limits.

Flow Run Time Considerations
When running flows, keep certain limits and guidelines in mind.

Flow Accessibility Considerations
Visual Workflow is 508-compliant with a few exceptions.

SEE ALSO:
Cloud Flow Designer Overview

General Flow Limits
When using Visual Workflow, keep flow limits and Apex governor limits in mind.

<table>
<thead>
<tr>
<th>Limit</th>
<th>Limitations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of versions per flow</td>
<td>50</td>
</tr>
<tr>
<td>Maximum number of executed elements at run time</td>
<td>2000</td>
</tr>
<tr>
<td>Maximum number of active flows and processes per organization</td>
<td>500</td>
</tr>
<tr>
<td>Maximum number of flows and processes per organization</td>
<td>1000</td>
</tr>
<tr>
<td>Maximum number of flow interviews or groups of scheduled actions (from processes) that are waiting at one time</td>
<td>30,000</td>
</tr>
<tr>
<td>Maximum number of flow interviews that are resumed or groups of scheduled actions that are executed per hour</td>
<td>1000</td>
</tr>
</tbody>
</table>

SEE ALSO:
Cloud Flow Designer Overview

Editions

Available in:
- Enterprise
- Performance
- Unlimited
- Developer
Maximum number of relative time alarms defined in flow versions or schedules based on a field value in processes 20,000

**Apex Governors and Limits for Visual Workflow**

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.

<table>
<thead>
<tr>
<th>Description</th>
<th>Per-Transaction Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total number of SOQL queries issued (Record Update, Record Delete, Record Lookup, and Fast Lookup element executions)</td>
<td>100</td>
</tr>
<tr>
<td>Total number of records retrieved by SOQL queries (across all Record Update, Record Delete, Record Lookup, and Fast Lookup elements executed in all interviews in the transaction)</td>
<td>50,000</td>
</tr>
<tr>
<td>Total number of DML statements issued (Record Create, Record Update, Record Delete, Fast Create, Fast Update, and Fast Delete executions)</td>
<td>150</td>
</tr>
<tr>
<td>Total number of records processed as a result of DML statements</td>
<td>10,000</td>
</tr>
</tbody>
</table>

Autolaunched flows are part of the larger transaction through which they were launched. For example, flows launched from a process are executed with the 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 as soon as a flow interview begins to wait 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.

SEE ALSO:
- Visual Workflow Overview
- Visual Workflow Limits and Considerations

**Flow Design Considerations**

When you design flows, keep certain guidelines in mind.

**Deleting Variables**

If you delete an sObject variable or sObject collection variable, any variable assignments that use the deleted variable are set to **null**.
Referring to Blank Fields or Resources
If you leave any field or resource value blank, that value is null at run time. To treat a text value as an empty string instead of null, set it to {!$GlobalConstant.EmptyString}.

Setting the Record Type
To set the record type for a record, use the ID of the record type. Look up the record type by its name and then store its ID in the flow.

For example, use a Record Lookup element to find the RecordType 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.

IN THIS SECTION:

Cloud Flow Designer Considerations
When you create a flow in the Cloud Flow Designer, familiarize yourself with its limitations and behaviors. For example, it supports a handful of locales and can’t open flows from managed packages.

Large Flow Guidelines
Business processes can be complex. When your flow is too large for the canvas, control the zoom, search in the Explorer tab, or collapse the left side panel.

Multi-Select Choice Fields Limitations
Multi-select checkboxes and multi-select picklist fields enable flow users to select multiple choices in a screen field. At run time, the value of each multi-select field is a concatenation of the user-selected choice values, separated by semicolons.

Flow Formula Limitations
When you create a formula resource or add validation to a screen input field, understand the limitations of formulas in Visual Workflow.

Time-based Flow Limitations
Before you design flows that contain one or more Wait elements, understand the limitations and guidelines.

SEE ALSO:
Create a Flow
Flow Operators
Visual Workflow Limits and Considerations
Cross-Object Field References in Flows

Cloud Flow Designer Considerations
When you create a flow in the Cloud Flow Designer, familiarize yourself with its limitations and behaviors. For example, it supports a handful of locales and can’t open flows from managed packages.

• At run time, time zones for date/time values can differ from what you see in the Cloud Flow Designer. During run time, date/time values reflect the running user’s time zone settings in Salesforce. In the Cloud Flow Designer, date/time values reflect the time zone set on your computer. The Cloud Flow Designer appends the GMT offset to your date/time value.

• The Cloud Flow Designer doesn’t support UTF-8 encoding for text in user input fields.

• The Cloud Flow Designer contains embedded fonts for all locales it supports. The supported locales are:
  – English (US)
  – French (France)
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 the Cloud Flow Designer.

- The Cloud Flow Designer can't open flows that are installed from managed packages.
- Don't enter the string `null` as the value of a text field in the Cloud Flow Designer.
- The Cloud Flow Designer has access to information that exists when you open it. If you modify data or metadata in your organization and need to refer to it in a flow, close and reopen the Cloud Flow Designer. For example, if you add a custom field or modify an Apex class with the Cloud Flow Designer open, close and reopen the Cloud Flow Designer.
- The Cloud Flow Designer uses the permissions and locale assigned to the current user.
- 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 name as the old decision.
  - Takes the unique name of the old decision, appended with "_switch".
  - Has an outcome labeled "True". This outcome's unique 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".

**Large Flow Guidelines**

Business processes can be complex. When your flow is too large for the canvas, control the zoom, search in the Explorer tab, or collapse the left side panel.

**Zoom**

To zoom in and out of your flow, use the + and - buttons on the right side of the canvas.

**Search in the Explorer tab**

Looking for a specific element or resource? Search for it in the Explorer tab.

- To find an element with a specific name, type in the search box.
To find all instances of a certain element or resource, click the magnifying glass and select the type.

Once you find the right resource in the Explorer tab, see which elements are using the resource. In the Description pane, click the Usage tab.

Once you find the right element in the Explorer, find that element in your canvas. Hover over the element, and click the magnifying glass.

The element is highlighted in green in your canvas.

If the element wasn’t in view, the Cloud Flow Designer automatically scrolls to show the element.

**Collapse the left side panel**
To hide the Palette, Resources, and Explorer tabs from your view, click the left arrow next to the side panel. That way, you get even more space in the canvas.
Multi-Select Choice Fields Limitations

Multi-select checkboxes and multi-select picklist fields enable flow users to select multiple choices in a screen field. At run time, the value of each multi-select field is a concatenation of the user-selected choice values, separated by semicolons.

When using multi-select choice fields in a Screen element, consider the following:

- At run time, all semicolons in the selected choices' stored values are removed when added to the multi-select field value.
- You can populate multi-select choice fields with only flow-specific choices and dynamic choices. You can't directly populate multi-select picklists in flows with the values of picklist or multi-select picklist fields in the Salesforce database.
- A multi-select choice field can have only one default value.
- A dynamic choice resource can be configured to assign field values from a user-selected record to variables in the flow. When a multi-select choice field uses a dynamic choice, only values from the last record that the user selects are stored in the flow variables. If multiple multi-select choice fields on one screen use the same dynamic choice, the first multi-select choice field on the screen determines the flow variable assignments.
- To ensure the correct run time behavior of flow conditions that reference multi-select choice fields:
  - Configure a stored value for each choice that you use in multi-select choice fields.
  - Don’t use the same choice in multiple multi-select choice fields on the same screen.

SEE ALSO:

Customize Choice Fields on a Flow Screen
Flow Formula Limitations

When you create a formula resource or add validation to a screen input field, understand the limitations of formulas in Visual Workflow.

- Flow formulas can't contain more than 3,000 characters.
- A formula returns `null` if:
  - 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.

- These functions aren’t supported in a flow formula.
  - `GETRECORDIDS`
  - `IMAGE`
  - `INCLUDE`
  - `ISCHANGED`
  - `ISNEW`
  - `PARENTGROUPVAL`
  - `PREVGROUPVAL`
  - `PRIORVALUE`
  - `REQUIRE SCRIPT`
  - `VLOOKUP`

For a complete list of operators and functions for building formulas in Salesforce, see “Formula Operators and Functions Overview” in the Salesforce Help.

- In a flow, the `CONTAINS` function checks all of the 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 an sObject 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 of the characters in the field.

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

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

SEE ALSO:

- Add a Formula to a Flow
- Flow Resources
Time-based Flow Limitations

Before you design flows that contain one or more Wait elements, understand the limitations and guidelines.

- After you deactivate a flow or flow version, the associated waiting interviews continue as usual. You can’t delete a flow or flow version if it has associated waiting interviews.
- An interview can execute only one event path per Wait element. After one of its events is processed, the remaining events are removed from the queue.
- An organization can process up to 1000 events per hour. When an event is processed, the interview that it’s associated with is resumed and any other events for that interview are removed from the queue. If an organization exceeds this limit, Salesforce processes the remaining events in the next hour.

For example, an organization has 1,200 events scheduled to be processed between 4:00 PM and 5:00 PM. Salesforce processes 1000 events between 4:00 PM and 5:00 PM and the additional 200 events between 5:00 PM and 6:00 PM.

- An organization can have up to 30,000 interviews waiting at a given time.
- If the user who started the interview is deactivated when Salesforce tries to execute an event path, the interview fails to resume.

Transactions and Waiting Interviews

A transaction ends as soon as a flow interview begins to wait 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.

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. Any actions that fire as a result of those grouped interviews are also executed in that transaction. This behavior can cause you to exceed your Apex governor limits if the resumed interview executes DML operations or SOQL queries through:

- Flow elements such as Record Create or Fast Lookup
- Apex Plug-in elements
- Apex triggers
- Immediate workflow actions

For details on Apex governor limits, see General Flow Limits on page 3.

If a Wait element precedes any flow elements that execute DML operations or SOQL queries:

- Ensure that your flows don’t perform more DML operations or SOQL queries between Wait elements than the Apex governor limits allow.
- 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 waited are successful.
- If a fault connector 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 connector 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.

**General Alarms Limitations**

• Alarms don’t support minutes or seconds.

• If an interview is waiting for an event that’s set for a time in the past, Salesforce resumes the interview within one hour.

  For example, a flow is configured to email an opportunity owner seven days before the close date. An interview is started for an opportunity with the close date set to today. Salesforce resumes the interview within an hour.

**Absolute Time Alarms Limitations**

• Absolute time alarms are evaluated based on the time zone of the user who created the flow.

**Relative Time Alarms Limitations**

• Relative time alarms are evaluated based on the organization’s time zone.

• Across all of your flow versions, your organization can have up to 20,000 defined relative time alarms.

• Alarms can’t reference the following:
  – **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 relative time alarm in a waiting interview, Salesforce recalculates the 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 relative time alarm and the interview resumes on 2/3/2014.

  – The close date is updated to a date in the past. Result: Salesforce recalculates the relative time alarm and resumes the interview shortly after you save the record.

• If a relative time alarm references a null date field when the interview executes the Wait element, Salesforce resumes the interview within an hour.

• If a relative time alarm references a date field that’s that has a non-null value when the flow interview executes the Wait element and it’s updated to **null** before the alarm is processed, Salesforce resumes the interview within an hour after the date field is updated.

• If a waiting interview has a relative time alarm and the referenced record or object is deleted, the alarm is removed from the queue. If the interview has no other events to wait for, the interview is deleted.

• You can’t archive a product or price book that’s referenced in a relative or absolute time alarm in a waiting interview.

• **Lead Convert Limitations**
  – You can’t convert a lead that has associated relative time alarms in waiting interviews.
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 waiting interview that’s associated with that record finishes, Salesforce still executes the interview.

SEE ALSO:
Flow Design Considerations
Visual Workflow Limits and Considerations
Operators in Flow Conditions
Configure a Flow to Wait for Events

Flow Administration Considerations

When managing flows, consider the administration and activation limits.

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 any paused or waiting interviews, it can’t be deleted until those interviews are finished or deleted. Flows that have never been activated can be deleted immediately.

Flow Properties
The properties for a given flow’s versions automatically match the active version’s properties by default. In other words, if you have three versions and you activate version 2, Salesforce updates the properties for versions 1 and 3 to match version 2. However, if you edit the properties for an inactive version, that version’s properties are no longer automatically updated to match the active version.

The flow’s active (or latest) version determines the flow’s type. For example, if a flow’s active version contains a screen, its type is Flow. It can’t be implemented through a system-based method, like the Process Builder.

IN THIS SECTION:
Installed Flow Considerations
Keep these considerations in mind when you distribute, upgrade, or remove a flow that you installed from a package.

SEE ALSO:
Manage Your Flows
Visual Workflow Limits and Considerations
Installed Flow Considerations

Keep these considerations in mind when you distribute, upgrade, or remove a flow that you installed from a package.

- The Cloud Flow Designer can’t open flows that are installed from managed packages.
- If you install a package that contains multiple flow versions in a fresh destination organization, only the latest flow version is deployed.
- If you install a flow from an unmanaged package that has the same name but a different version number as a flow in your organization, 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 organization, the package install fails. You can’t overwrite a flow.

Status

An active flow in a package is active after it’s installed. The previous active version of the flow in the destination organization 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 organization 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 organization, deactivate it and then uninstall the package.
- You can’t delete flow components from Managed - Beta package installations in development organizations.
- 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.

SEE ALSO:

Packaging Considerations for Flows

Flow Run Time Considerations

When running flows, keep certain limits and guidelines in mind.

- Be careful when testing flows that contain delete elements. Even if the flow is inactive, it triggers the delete operation.
- At run time, time zones for date/time values can differ from what you see in the Cloud Flow Designer. During run time, date/time values reflect the running user’s time zone settings in Salesforce.
- 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.

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.

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 variables of type Currency can accept only numeric values, so the flow fails at run time. Instead, pass the record’s ID to a flow Text variable with a URL parameter. Then in the flow, use the ID to look up that record’s value for the currency field.

If a flow is resumed from the Paused Flow Interviews item in Salesforce1, validation isn’t performed on Currency or Number input fields when the value exceeds 18 digits.

SEE ALSO:
   Test a Flow
   Visual Workflow Limits and Considerations

Flow Accessibility Considerations

Visual Workflow is 508-compliant with a few exceptions.

Clicking the next or previous buttons doesn’t change the title of the screen, so you may not realize you’re on a new page.

There are no labels on radio buttons. Screen readers can’t distinguish between questions.

Questions without defined prompts can read incorrectly.

Errors are not noted when reading the fields.

SEE ALSO:
   Visual Workflow Limits and Considerations
CHAPTER 2  Cloud Flow Designer Overview

The Cloud Flow Designer lets you design flows without writing any code.

Watch a Demo: Visual Workflow Cloud Flow Designer Overview

For a collection of useful resources, including videos and sample flows, open the Cloud Flow Designer and click Get Started.

IN THIS SECTION:

Requirements for the Cloud Flow Designer
To use the Cloud Flow Designer, you need an up-to-date browser and Adobe® Flash® Player.

Tour the Cloud Flow Designer User Interface
Before you use the Cloud Flow Designer to design flows, understand the tool’s main components.

Manage Elements, Resources, and Connectors in a Flow
Customize your flow by adding, editing, or removing elements, resources, and connectors.

Search Within a Flow
As a flow grows and becomes more complex, it becomes more challenging to find things within it. The Cloud Flow Designer offers tools for quickly finding flow elements and resources.

Search Within the Palette
As you add more flows, actions, and Apex classes to your organization, it becomes more challenging to find a specific item in the Palette. You can, however, search in the Palette to quickly find the right element for your flow.

SEE ALSO:

Flow Elements
Flow Resources

Requirements for the Cloud Flow Designer

To use the Cloud Flow Designer, you need an up-to-date browser and Adobe® Flash® Player.

We recommend:
- Windows® Internet Explorer®, versions 8 through 11, Google® Chrome™, or Mozilla® Firefox®. Internet Explorer 6 and 7 are not supported.
- Adobe® Flash® Player version 10.1 and later. The minimum version required to run the Cloud Flow Designer is 10.0.
- A minimum browser resolution of 1024x768.
Tour the Cloud Flow Designer User Interface

Before you use the Cloud Flow Designer to design flows, understand the tool’s main components.

**EDITIONS**

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

---

**Button Bar (1)**

Use the button bar to manage your flow as you build it.

- **Run** runs the most recent saved version of the flow that you have open. If the flow contains subflow elements, each subflow runs the active version of its referenced flow. If the referenced flow has no active version, then the subflow element runs the latest version of its referenced flow.
- The status indicator on the right side displays whether:
  - The flow is active or not
  - The latest changes to the flow are saved or not
  - There are any warnings or errors in the last saved version of the flow

To see a list of the warnings or errors, click the indicator.

---

**Canvas (2)**

The canvas is the working area, where you build a flow by adding elements. As you add elements to the canvas and connect them together, you see a visual diagram of your flow.

---

**Palette Tab (3)**

Add new elements, like Screens and Record Creates, to your flow from the Palette tab.
Resources Tab (4)
Create new resources, like a variable or formula, to use in your flow from the Resources tab.

Explorer Tab (5)
The Explorer tab is a library of all elements and resources that you’ve added to the flow.

SEE ALSO:
Flow Properties
Manage Elements, Resources, and Connectors in a Flow
Search Within the Palette
Search Within a Flow

Manage Elements, Resources, and Connectors in a Flow
Customize your flow by adding, editing, or removing elements, resources, and connectors.

<table>
<thead>
<tr>
<th>Add</th>
<th>Edit</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element</td>
<td>Drag from the Palette tab and drop it on to the canvas.</td>
<td>Double-click, or hover over it and click 🔄.</td>
</tr>
<tr>
<td>Resource</td>
<td>From the Resources tab, double-click.</td>
<td>From the Explorer tab, double-click or hover over it and click 🔄.</td>
</tr>
<tr>
<td>Connector</td>
<td>Click the node at the bottom of an element on the canvas and drag a line anywhere onto the target element.</td>
<td>n/a</td>
</tr>
</tbody>
</table>

SEE ALSO:
Flow Elements
Flow Resources
Flow Connectors
Search Within a Flow

As a flow grows and becomes more complex, it becomes more challenging to find things within it. The Cloud Flow Designer offers tools for quickly finding flow elements and resources.

Open the flow in the Cloud Flow Designer. Then use one or more of the following options to find an element or resource in the flow.

- On the Explorer tab, enter search text. The Explorer tab displays only the elements and resources whose properties contain the entered text.
- Click \( \text{🔍} \) to filter the Explorer tab contents to one type of element or resource. To remove the filter, click \( \text{🔍} \) and select **SEARCH ALL**.
- Select **Highlight Results on Canvas** to dim all visible elements on the canvas other than the Explorer search or filter results.
- Zoom in and out as desired using the controls near the top right corner of the canvas area.
- To see the location of an Explorer item on the canvas, complete one of the following procedures.
  - If the Explorer item is a canvas-visible element or a screen field:
    1. Hover over the item on the Explorer tab.
    2. Click \( \text{🔍} \).
  - If the Explorer item is a resource that doesn’t appear on the canvas:
    1. Click the item on the Explorer tab.
    2. Click the Usage tab in the Description pane.
    3. Hover over an element listed on the Usage tab.
    4. Click its \( \text{🔍} \). The canvas shifts to display the element and momentarily highlights it.

SEE ALSO:
- **Tour the Cloud Flow Designer User Interface**
Search Within the Palette

As you add more flows, actions, and Apex classes to your organization, it becomes more challenging to find a specific item in the Palette. You can, however, search in the Palette to quickly find the right element for your flow.

From the Palette tab, use the following options to find a specific item.

- Next to 🔍, enter search text. The Palette displays only the items that contain the entered text.
- To filter the Palette tab contents to one type of element, click ⚒️ and select what you want to see.
- To remove the filter, click ⚒️ and select SEARCH ALL.

SEE ALSO:
- Tour the Cloud Flow Designer User Interface

EDITIONS

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

USER PERMISSIONS

To open, edit, or create a flow in the Cloud Flow Designer:
- “Manage Force.com Flow”
Once you understand the process that you want to automate, design a flow in the Cloud Flow Designer for that process.

Tip: Before you start creating your flow, plan it out. It’s much easier to automate a business process by using Visual Workflow when you fully understand the details of your business process.

If you’re new to the Cloud Flow Designer, we recommend walking through one or more of the sample flow tutorials in the Cloud Flow Designer Workbook. They’re a great way to learn about the tool and discover how it works.

2. Drag the appropriate elements onto the canvas.
   Tip: If you’re not sure which element you need for a node, add a Step element as a placeholder until you figure it out. You can always replace the Step later.
3. Connect the elements together so that it’s clear what the order of the elements is.
4. Identify which element the flow should start with when it runs.
5. Save any changes that you made to the flow.
6. Test the flow to make sure it’s working as you expect it to.
7. Activate the flow so that users can run it.
8. Distribute the flow to the appropriate users.

SEE ALSO:
- Manage Your Flows
- Flow Design Considerations
- Flow Accessibility Considerations
- Flow Building Blocks

Flow Elements

Each element represents an action that the flow can execute. Examples of such actions include reading or writing Salesforce data, displaying information and collecting data from flow users, executing business logic, or manipulating data.

In the Cloud Flow Designer, the canvas and Explorer tab display the elements that exist in the flow. The Palette tab displays the available element types that you can add to the flow by dragging them onto the canvas.
<table>
<thead>
<tr>
<th>Flow Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apex</td>
<td>Calls an Apex class’s invocable method.</td>
</tr>
<tr>
<td>Apex Plug-in</td>
<td>Calls an Apex class that implements the <code>Process.Plugin</code> interface. If you used the Tag property in the <code>PluginDescribeResult</code> class, the Apex class appears under a customized section. Otherwise, it appears under the Apex Plug-ins section.</td>
</tr>
<tr>
<td>Assignment</td>
<td>Sets or changes values in variables, collection variables, sObject variables, and sObject collection variables.</td>
</tr>
<tr>
<td>Decision</td>
<td>Evaluates conditions to determine which flow path to take.</td>
</tr>
<tr>
<td>Email Alert</td>
<td>Sends an email by using a workflow email alert to specify the email template and recipients.</td>
</tr>
<tr>
<td>Fast Create</td>
<td>Creates Salesforce records using the field values from an sObject collection variable. Or creates one Salesforce record using the field values from an sObject variable.</td>
</tr>
<tr>
<td>Fast Delete</td>
<td>Deletes Salesforce records using the ID values from an sObject collection variable. Or deletes one Salesforce record using the ID value from an sObject variable.</td>
</tr>
<tr>
<td>Fast Lookup</td>
<td>Queries Salesforce records to assign their field values to an sObject collection variable. Or queries one Salesforce record to assign its field values to an sObject variable.</td>
</tr>
<tr>
<td>Fast Update</td>
<td>Updates Salesforce records using the field values from an sObject collection variable. Or updates one Salesforce record using the field values from an sObject variable.</td>
</tr>
</tbody>
</table>
| Loop         | Iterates through a collection variable. For each item in this collection variable, the Loop element:  
1. Assigns the item’s values to a loop-specific variable.  
2. Executes the loop-specific flow path, which contains elements that use the loop-specific variable. For example, manipulate the variable’s values and then assign the updated values to another collection variable that can be referenced later in the flow. |
| Post to Chatter | Posts a message to the feed for a specific record, user, or Chatter group. |
| Quick Action | Calls an object-specific or global quick action. |
| Record Create | Creates one Salesforce record using field values that you specify individually. You can assign these values from flow resources, such as variables and screen fields. |
| Record Delete | Deletes Salesforce records that meet the filter criteria you specify. |
| Record Lookup | Queries one Salesforce record that meets the filter criteria you specify. Then assigns the record’s field values to separate, individual flow variables or individual fields on sObject variables. |
| Record Update | Updates one Salesforce record using field values that you specify individually. You can assign these values from flow resources, such as variables and screen fields. |
| Screen       | Displays a screen to the user who is running the flow. The screen contains fields for user input or for displaying information to the user. Fields and choices that you create for the screen are available as resources that you can reference elsewhere in the flow. |
| Send Email   | Sends an email by using the flow to specify the subject, body, and recipients. |
| Step         | Serves as a temporary placeholder while you design a flow. You can convert Step elements into Screen elements. |
### Flow Element Description

<table>
<thead>
<tr>
<th>Flow Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Submit for Approval</td>
<td>Submits one Salesforce record for approval.</td>
</tr>
<tr>
<td>Subflow</td>
<td>Calls another flow in the organization.</td>
</tr>
<tr>
<td>Wait</td>
<td>Waits for one or more defined events to occur.</td>
</tr>
</tbody>
</table>

SEE ALSO:
- Flow Resources
- Cloud Flow Designer Overview

### Invoke Apex Code from a Flow

The Cloud Flow Designer comes with a lot of functionality, but sometimes your flow needs to do more than the default elements allow. In that case, call an Apex class from your flow by using one of two flow elements: Apex Plug-in and Call Apex.

Developers have two options when they’re trying to make an Apex class available for a flow.

⚠️ Tip: We recommend using the `@InvocableMethod` annotation instead of the `Process.Plugin` interface.

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.

<table>
<thead>
<tr>
<th>Apex data type support</th>
<th>Process.Plugin Interface</th>
<th>@InvocableMethod Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Doesn’t support:</td>
<td>Doesn’t support:</td>
</tr>
<tr>
<td></td>
<td>• Blob</td>
<td>• Generic Object</td>
</tr>
<tr>
<td></td>
<td>• Collection</td>
<td>• Generic sObject</td>
</tr>
<tr>
<td></td>
<td>• sObject</td>
<td>• Sets</td>
</tr>
<tr>
<td></td>
<td>• Time</td>
<td>• Maps</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Enums</td>
</tr>
</tbody>
</table>

The Cloud Flow Designer doesn’t support mapping an Apex method’s input or output parameters to an sObject collection variable.

<table>
<thead>
<tr>
<th>Bulk operations</th>
<th>Not supported</th>
<th>Supported</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Element name in the Cloud Flow Designer</th>
<th>Process.Plugin Interface</th>
<th>@InvocableMethod Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class name or the value of the <code>name</code> property.</td>
<td>Class name</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reusability</th>
<th>Process.Plugin Interface</th>
<th>@InvocableMethod Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classes with this interface implemented are available in flows</td>
<td>Classes with this annotation implemented are available in:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Flows</td>
</tr>
</tbody>
</table>
Create a Flow

<table>
<thead>
<tr>
<th>Process.Plugin Interface</th>
<th>@InvocableMethod Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Processes</td>
<td></td>
</tr>
<tr>
<td>• Rest API</td>
<td></td>
</tr>
</tbody>
</table>

**Section in the Cloud Flow Designer**
- Apex Plug-in or the value of the `tag` property.
- Apex

**More Details in the Force.com Apex Code Developer’s 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.

```java
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;
    }
}
```

This class implements the `Process.Plugin` interface.

```java
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
```

---

23
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.

IN THIS SECTION:

Add an Apex Plug-In Element to a Flow
Use an Apex Plug-in element to call an Apex class that has the Process.Plugin interface implemented.

Add a Call Apex Element to a Flow
Use a Call Apex element to call an Apex class that contains an invocable method.

SEE ALSO:
Flow Elements

Add an Apex Plug-In Element to a Flow

Use an Apex Plug-in element to call an Apex class that has the Process.Plugin interface implemented.

Tip:
- Apex classes appear in the palette as Apex plug-ins only if the Process.Plugin interface has been implemented.
- If you have many plug-ins in your organization, use the tag property. The class appears under a special section header in the palette. Otherwise, the class appears with all the other Apex plug-ins.
- 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 much easier to implement. To see a complete comparison between the interface and the annotation, see Invoke Apex Code from a Flow on page 22.

1. From Setup, click Create > Workflow & Approvals > Flows
2. In the palette, next to , type the name of the Apex class that you want to call.
3. Drag that element onto the canvas.
4. Enter the general settings for the element.
### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>

| Description       | Appears after you click Add Description. |

5. On the **Inputs** tab, assign values to the Apex input parameters. Required parameters are automatically listed as targets.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Target</td>
<td>Input parameter whose value you want to set.</td>
</tr>
</tbody>
</table>
| Source            | Value to assign to the input parameter. Options:  
  - Select an existing flow resource, such as a variable, constant, or user input.  
  - Select CREATE NEW to create a flow resource.  
  - Manually enter a literal value or merge field. |

6. On the **Outputs** tab, assign values from Apex output parameters to variables in the flow.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Output parameter whose value you want to get.</td>
</tr>
<tr>
<td>Target</td>
<td>Flow variable whose value you want to set.</td>
</tr>
</tbody>
</table>

7. Click **OK**.

**Note:** If this element fails for some reason, the flow takes the fault connector path if you’ve defined one.

**SEE ALSO:**
- [Invoke Apex Code from a Flow](#)
- [Customize What Happens When a Flow Fails](#)
- [Add Connectors to Define Flow Paths](#)
- [Cross-Object Field References in Flows](#)
Add a Call Apex Element to a Flow

Use a Call Apex element to call an Apex class that contains an invocable method.

Important: To use this element to call an Apex class from a flow, ask your developer to annotate one of the class's methods with @InvocableMethod. For details, see “InvocableMethod Annotation” in the Force.com Apex Code Developer’s Guide.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. Find the class that you want to call.
   a. In the palette, click CALL APEX.
      The palette displays only Apex classes in your organization that contain an invocable method.
   b. Optionally, type three or more letters to search for a specific class.
3. Drag the desired class onto the canvas.
4. Enter general settings, so that you can differentiate this element from other elements in the flow.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB.</td>
</tr>
<tr>
<td></td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>

Description Appears after you click Add Description.

5. For each input parameter, manually enter a value, enter a merge field, select an existing flow resource, or create a flow resource.
6. To assign a value from the method’s outputs to variables in your flow, open the Outputs tab.
   The flow assigns the values to the specified variables when the method is executed.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Output parameter whose value you want to get.</td>
</tr>
<tr>
<td>Target</td>
<td>Flow variable whose value you want to set.</td>
</tr>
</tbody>
</table>

7. Click OK.
Note: If the element fails for some reason, the flow uses the fault connector path if you defined one.

SEE ALSO:
- Invoke Apex Code from a Flow
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

Add an Assignment Element to a Flow

Use the Assignment element to modify the value of a variable. For example, you can dynamically change the value of a customer’s credit score depending on how the customer answers questions in the flow.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Assignment element onto the canvas.
3. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>

4. Specify the variable assignments.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Variable whose value you want to change.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available operators depend on the data type selected for the Variable. For details, see Operators in Flow Assignment Elements on page 96.</td>
</tr>
</tbody>
</table>
| Value         | The Variable and Value in the same row must have compatible data types. Options:  
- Select an existing flow resource, such as a variable, constant, or user input.  
- Select CREATE NEW to create a flow resource.  
- Manually enter a literal value or merge field. |
5. Click OK.

Note: Every assignment you add to a flow is also added to the Explorer tab.

SEE ALSO:
- Flow Elements
- Operators in Flow Assignment Elements
- Add Connectors to Define Flow Paths
- Flow Resources
- Cross-Object Field References in Flows

Add a Decision Element to a Flow

Use the Decision element to 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. For example, use a Decision element to determine whether:

- To give customers a return shipping address (because an item is definitely faulty) or instructions on how to resolve the problem
- To offer a customer a loan or not (based on results of a credit scoring formula)

Tip: Configure your flow to do different things based on which option a user selected for a screen’s drop-down list. To do so, add a decision after the screen to create the branches of the flow based on the choices available in that drop-down list. Then you can represent each choice in your decision and connect it to a branch of your flow.

To add or configure a Decision element:

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Decision element onto the canvas.
3. Enter the general settings for the element.

### Field Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>
4. Create the outcomes for the decision. For each outcome:
   a. Enter a Name and Unique Name.
   b. Set up the conditions.
      If all of an outcome’s conditions evaluate to true, the flow takes that outcome’s path. For details, see Define Flow Conditions on page 74.

   To reorder outcomes, drag them up and down.

5. To rename the path that the flow takes when none of the other outcome conditions are met, click [Default Outcome].

6. Click OK.

After you set up your outcomes, connect your Decision element to other elements on the canvas and assign an outcome to each connector.

SEE ALSO:
Flow Elements
Operators in Flow Conditions
Add Connectors to Define Flow Paths
Cross-Object Field References in Flows

Add a Fast Create Element to a Flow

Use a Fast Create element to create a Salesforce record from a single sObject variable or to create multiple Salesforce records from an sObject collection variable.

For example, you could take a collection of new cases and use a Fast Create element to create records for each case in the collection. Make sure that your flow populates the sObject variable or collection with all required field values before executing the Fast Create element.

To create a single record with field values from regular variables and other flow resources, such as constants, formulas, and screen fields, use Record Create.

1. From Setup, click Create > Workflow & Approvals > Flows.

2. From the Palette, drag the Fast Create element onto the canvas.

3. Enter a Name and Unique Name for the element.

   Note: The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

4. If you haven’t already added a description, click Add Description to do so.

5. Select the sObject variable or collection that you want to use to create the record or multiple records.
The object types must match, and each ID field must not have a value.
If you haven’t created the variable yet, you can do that by expanding the CREATE NEW section of the drop-down list.

6. Click OK.

If you used an sObject variable to create a single record, the sObject variable’s ID field is updated with the new record’s ID value. If you used an sObject collection to create multiple records, the ID field of each collection item is updated with its matching new record ID value.

**Note:** If the Fast Create fails at run time, the flow takes the fault connector path if you’ve defined one.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Add an SObject Variable to a Flow
- Add an SObject Collection to a Flow

**Add a Fast Delete Element to a Flow**

Use a Fast Delete element to delete records in Salesforce.

You can delete a single record based on the ID value of an sObject variable that you’ve defined. A Fast Delete can also find and delete a set of records based on the ID field values of an sObject collection that you’ve defined. Make sure that the sObject variable or collection is populated with ID values before using the Fast Delete element.

To delete one or more records that meet filter criteria specified by regular variables and other flow resources, such as constants, formulas, and screen fields, use Record Delete.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Fast Delete element onto the canvas.
3. Enter a Name and Unique Name for the element.
   **Note:** The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
4. If you haven’t already added a description, click Add Description to do so.
5. Select the sObject variable or collection that you want to use to find the record or records to delete.
   The sObject variable or collection must have the same object type as the record type you’re deleting. That is, if you’re deleting several records of type account, use an sObject collection of type account.
6. Click OK.

**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 organization the moment the flow executes the delete element.

- Deleted records are sent to the Recycle Bin and remain there for 15 days before they are permanently deleted.
- Flows can delete records that are in an approval process.

Note: If the Fast Delete fails for some reason, the flow takes the fault connector path if you've defined one.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Add an SObject Variable to a Flow
- Add an SObject Collection to a Flow

## Add a Fast Lookup Element to a Flow

Use a Fast Lookup element to find a Salesforce record and store the values from its fields in a single sObject variable that you define. You can also use a Fast Lookup element to find a set of records and store their field values into an sObject collection that you define.

For example, you could use a Fast Lookup element to:

- Input (or read) a bar code from a product tag. Use the code to find out the product name or description from the database.
- Look up a city to list all customers in that city.
- Look up customer records to verify transactions on a particular day.

To get a single record and store specified field values in regular variables and sObject variables, use Record Lookup.

1. From Setup, click **Create > Workflow & Approvals > Flows.**
2. From the Palette, drag the Fast Lookup element onto the canvas.
3. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB.</td>
</tr>
</tbody>
</table>

The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Description  Appears after you click **Add Description.**
4. Select the object to identify which record type to look up.

5. Specify the filter criteria for selecting the record from the database.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Field for filtering records.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available filter operators depend on the data type selected for Field. For details, see Operators in Flow Record Filters on page 109.</td>
</tr>
</tbody>
</table>
| Value         | The Field and Value in the same row must have the same data type. Options:  
  - Select an existing flow resource, such as a variable, constant, or user input.  
  - Select CREATE NEW to create a flow resource.  
  - Manually enter a literal value or merge field. |

Make sure that your filter criteria sufficiently narrows the search. The Fast Lookup element returns only the first record from the filtered results if you use an sObject variable to contain the results. Use an sObject collection variable to contain the values from multiple results.

6. To sort the filtered results before any records are selected:
   a. Select Sort results by.
   b. Select the field to sort by. Only sortable fields appear in the drop-down list.
   c. Select the sort order, either ascending or descending.

7. Select or create a variable to contain the values returned from the filters.
   - To contain the field values for the first returned record, use an sObject variable.
   - To contain the field values for all of the returned records, use an sObject collection variable.

8. Select Assign null to the variable if no records are found to have the variable set to null if the Fast Lookup doesn’t return any records. The default is to leave the sObject variable or collection values unchanged.

9. Select the record fields to save into the sObject variable or collection. Only the values of fields you select are saved. Unselected values are set to null in the sObject variable or collection.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Record’s field whose value you want to assign to a variable.</td>
</tr>
<tr>
<td>Variable</td>
<td>Select an existing flow variable, or select CREATE NEW to create a variable.</td>
</tr>
</tbody>
</table>

10. Click OK.
Note: If the Fast Lookup fails for some reason, the flow takes the fault connector path if you’ve defined one.

SEE ALSO:
- Flow Elements
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Add an SObject Variable to a Flow
- Add an SObject Collection to a Flow
- Cross-Object Field References in Flows

Add a Fast Update Element to a Flow

Use a Fast Update element to update one Salesforce record based on an sObject variable or multiple Salesforce records based on an sObject collection variable.

For example: you’re designing flows for a call center. To automatically update Salesforce with data collected from callers, such as new addresses or product preferences, use a Fast Update element. Have your flow populate the sObject variable or collection before using the Fast Update element. Then make sure that the sObject variable or sObject values within the collection contain the ID for the records that are being updated.

To update one or more records with field values from regular variables and other flow resources, such as constants, formulas, and screen fields, use Record Update.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Fast Update element onto the canvas.
3. Enter a Name and Unique Name for the element.
   Note: The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
4. If you haven’t already added a description, click Add Description to do so.
5. Select the sObject variable or collection that you want to use to update the record or records.
   The sObject variable or collection must be the same object type as the records being updated. That is, if you’re updating several account records, use an sObject collection of type account. This practice ensures that the fields in the records match the fields in the sObject variable or collection.
   If you haven’t created the variable yet, you can do that by expanding the CREATE NEW section of the drop-down list.
6. Click OK.
Note: If the Fast Update fails for some reason, the flow takes the fault connector path if you’ve defined one.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Add an SObject Variable to a Flow
- Add an SObject Collection to a Flow

Add a Loop Element to a Flow

Use a loop to examine every item in a collection and execute actions on each item’s field values using other elements within the loop. A collection is a list of items that contain field values from Salesforce records.

Loops examine one item in a collection at a time. A loop uses an sObject variable, referred to as a loop variable, to contain the values for the current item in the collection. Once the loop finishes examining an item, it copies the field values for the next item into the loop variable. Then the loop examines those values. The loop variable must have the same object type as the collection. For example, if your collection contains field values from accounts, your loop variable must also be of type Account.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. From the Palette, drag the Loop element onto the canvas.
3. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
</tbody>
</table>
| Unique Name    | Automatically populated if empty when you fill out the Name field and press TAB.  

The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

4. In the Assignments section, select the collection you want to loop through.

If you haven’t created the variable yet, you can do that by expanding the CREATE NEW section of the drop-down list.

5. Select the order to loop through the collection.

Ascending begins at the start of the collection and moves to the end, while Descending begins at the end and moves to the start.
6. Select the sObject variable to use as the loop variable. Use the CREATE NEW section of the drop-down list to create resources, if necessary.

7. Click **OK**.

After you add a Loop element and the elements that you want the loop to include, from the Loop element:

- Add a “Next element” connector to determine which element to execute first when a new item’s values are copied into the loop variable.
- Add an “End of loop” connector to determine which flow element to execute after the loop has processed all the items in the collection.

SEE ALSO:
- Sample Flow That Loops Through a Collection
- Add Connectors to Define Flow Paths
- Flow Elements
- Add an SObject Collection to a Flow

### Sample Flow That Loops Through a Collection

This example demonstrates how to use sObject variable collections and loops to transfer ownership of accounts from one user to another. The flow already has the required user IDs.

First, create an Account-based sObject collection variable called `collAcctJSmith` and populate it with all account records that John Smith owns.

Then create a loop that iterates through the collection. For each item in the collection, the loop does the following:

1. Assigns the collection item to the loop variable.
2. Evaluates whether the account has more than 10,000 employees.
3. If the account has more than 10,000 employees, assigns Madison’s user ID to the `OwnerId` field in the loop variable.
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.
5. Adds the loop variable’s values as a new item in a second collection called `collReassignedAccts`.

Finally, create a Fast Update element to update the accounts in `collReassignedAccts` with the new `OwnerId` after the loop finishes iterating through the collection.
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 created a similar flow by using Record Update elements, you would use:

- One Record Update 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 Record Update 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).

Add a Record Create Element to a Flow

Use the Record Create element to create a record in Salesforce by using values from your flow.

For example, a user enters the name and address into the flow. Verify that a matching user exists by using the Record Lookup element. If a matching contact doesn’t exist, create a record for that user by using the Record Create element.

To create a single record with all field values from one sObject variable, or multiple records with all field values from an sObject collection, use Fast Create.

1. From Setup, click **Create > Workflow & Approvals > Flows**.
2. From the Palette, drag the Record Create element onto the canvas.
3. Enter a **Name** and **Unique Name** for the element.

   **Note:** The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

4. If you haven’t already added a description, click **Add Description** to do so.
5. Select the object for which you want to create a record.
6. Select the field or fields from the object that you want to populate, then assign a value or resource to the field. The data types must match.

In the Value column, you can manually enter literal or merge field values. You can also select pre-defined flow values (from variables, constants or user inputs, for example). Use the CREATE NEW section of the drop-down list to create resources, if necessary.

7. Optionally, assign the new record’s ID to a variable so you can use it or refer to it later in the flow. The variable you choose must be of type TEXT. If you haven’t created the variable yet, you can do that by expanding the CREATE NEW section of the drop-down list.

8. Click OK.

Important: Check your object definition to ensure that all required fields are populated with values; otherwise the record create fails at run time. If the record create does fail, the flow takes the fault connector path if you’ve defined one.

SEE ALSO:
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Flow Elements

Add a Record Delete Element to a Flow

Use a Record Delete element to delete records in Salesforce. You can delete a single record or multiple records, depending on how you set your filter criteria.

To delete a single record identified by the ID in one sObject variable, or delete multiple records identified by the IDs in an sObject collection, use Fast Delete.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Record Delete element onto the canvas.
3. Enter a Name and Unique Name for the element.

Note: The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

4. If you haven’t already added a description, click Add Description to do so.
5. Select the object whose records you want to delete.
6. Add rows of filter criteria to determine which records to delete. Configure at least one filter.

In the Value column, you can manually enter literal or merge field values. You can also select pre-defined flow values (from variables, constants or user inputs, for example). Use the CREATE NEW section of the drop-down list to create resources, if necessary.

7. Click OK.

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 organization the moment the flow executes the delete element.
- Deleted records are sent to the Recycle Bin and remain there for 15 days before they are permanently deleted.
- Flows can delete records that are in an approval process.

Note: If the record delete fails for some reason, the flow takes the fault connector path if you've defined one.

SEE ALSO:
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Flow Elements

Add a Record Lookup Element to a Flow

Use a Record Lookup element to find a Salesforce record and store the values from its fields in your flow.

For example, you could use a Record Lookup element to:
- Input (or read) a bar code from a product tag. Use the code to find out the product name or description from the database.
- Look up item details to check your stock for availability.
- Look up a customer record to verify a caller's identity.

Use a Fast Lookup element to find:
- A single record and store specified field values in one sObject variable, or
- Multiple records and store specified field values in an sObject collection

To add and configure a Record Lookup element:

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Record Lookup element onto the canvas.
3. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Appears after you click Add Description.</td>
</tr>
</tbody>
</table>

4. Select the object to identify which records to look up.
5. Specify the filter criteria for selecting the record from the database.
Make sure that your filter criteria sufficiently narrows the search. This element returns only the first record from the filtered results, which aren’t retrieved from the database in any particular order.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Field for filtering records.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available filter operators depend on the data type selected for Field. For details, see Operators in Flow Record Filters on page 109.</td>
</tr>
</tbody>
</table>
| Value         | The Field and Value in the same row must have the same data type. Options:  
  - Select an existing flow resource, such as a variable, constant, or user input.  
  - Select CREATE NEW to create a flow resource.  
  - Manually enter a literal value or merge field. |

6. To sort the filtered results before the first record is selected:
   a. Select Sort results by.
   b. Select the field to sort by. Only sortable fields appear in the drop-down list.
   c. Select the sort order, either ascending or descending.

7. Assign the field values from the returned record to variables in the flow.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Record’s field whose value you want to assign to a variable.</td>
</tr>
<tr>
<td>Variable</td>
<td>Select an existing flow variable, or select CREATE NEW to create a variable.</td>
</tr>
</tbody>
</table>

8. Click OK.

Note: If the record lookup fails for some reason, the flow takes the fault connector path if you've defined one.

SEE ALSO:
- Operators in Flow Record Filters
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows
Add a Record Update Element to a Flow

Use a Record Update element to update a set of records in Salesforce with variable, constant, input, or other values from your flow. You can update a single record, or multiple records, depending on how you set your filter criteria.

For example: to automatically update Salesforce with data collected from customers, such as new addresses or product preferences, use a Record Update element.

Use Fast Update to update:

- A single record with all field values from one sObject variable, or
- Multiple records with all field values from an sObject collection

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Record Update element onto the canvas.
3. Enter the general settings for the element.

4. Select the object for which you want to update a record.
5. Narrow down the number of records that the flow updates by specifying filter criteria.

**Important:** Configure at least one filter, or the flow updates all the records for the object.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Field for filtering records.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available filter operators depend on the data type selected for Field. For details, see Operators in Flow Record Filters on page 109.</td>
</tr>
<tr>
<td>Value</td>
<td>The Field and Value in the same row must have the same data type. Options:</td>
</tr>
<tr>
<td></td>
<td>- Select an existing flow resource, such as a variable, constant, or user input.</td>
</tr>
<tr>
<td></td>
<td>- Select CREATE NEW to create a flow resource.</td>
</tr>
<tr>
<td></td>
<td>- Manually enter a literal value or merge field.</td>
</tr>
</tbody>
</table>
6. Identify which fields to update on the records that meet your filter criteria, as well as the new values.

<table>
<thead>
<tr>
<th><strong>Column Header</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Record’s field whose value you want to update.</td>
</tr>
<tr>
<td>Variable</td>
<td>The updated value for the selected field. Options:</td>
</tr>
<tr>
<td></td>
<td>• Select an existing flow resource, such as a variable, constant, or user input.</td>
</tr>
<tr>
<td></td>
<td>• Select CREATE NEW to create a flow resource.</td>
</tr>
<tr>
<td></td>
<td>• Manually enter a literal value or merge field.</td>
</tr>
</tbody>
</table>

7. Click **OK**.

ℹ️ **Note:** If the record update fails for some reason, the flow takes the fault connector path if you’ve defined one.

SEE ALSO:
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Flow Elements

### Invoke an Object-Specific or Global Quick Action from a Flow

Have the flow call an object-specific or global quick action that’s already been configured in your organization. For example, your organization has an object-specific action that creates a case record on an account. The flow calls that action at run time and uses input assignments to transfer data from the flow to the action.

The unique name for each object-specific action is prefixed with the object it’s associated with. The unique name for each global action has no prefix.

1. From Setup, click **Create > Workflow & Approvals > Flows** and open a new or existing flow.
2. Find the action that you want to call.
   a. In the Palette, click ⌃ and select **QUICK ACTIONS**. The Palette displays only the object-specific and global actions in your organization.
   b. Optionally, type three or more letters to search for a specific action.
3. Drag the desired action onto the canvas.
4. Enter general settings, so that you can differentiate this action from other actions in the flow.

<table>
<thead>
<tr>
<th><strong>Field</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
</tbody>
</table>
Create a Flow

Post to Chatter from a Flow

**Field** | **Description**
---|---
Unique Name | Automatically populated if empty when you fill out the Name field and press TAB.
The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

Description | Appears after you click **Add Description**.

5. For object-specific actions, enter a value for **Related Record ID** to represent the ID of the record from which the action executes.
For example, the action creates a case that is associated with a given account. Assign the ID for that account to **Related Record ID**.

6. For each input parameter, enter a literal value, enter a merge field, select an existing flow resource, or create a flow resource.
Required parameters appear by default and can’t be removed. The action layout determines which parameters are required. 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.

7. Click **OK**.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

**Post to Chatter from a Flow**

Use a Post to Chatter element to post a message to a specified feed, such as to a Chatter group or a case record. The Post to Chatter element allows the message to contain mentions and topics, but only text posts are supported.

Use the input assignments to reference data from the flow to create a Chatterpost. Use the output assignments to use data from the Chatter post in the flow.

1. From Setup, click **Create > Workflow & Approvals > Flows**, and open a new or existing flow.
2. In the Palette, type **Post** next to **Q**.
3. Drag the Post to Chatter element onto the canvas.
4. Enter the general settings for the element.

**Field** | **Description**
---|---
Name | Helps you identify the element on the canvas.

**EDITIONS**

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

**USER PERMISSIONS**

To open, edit, or create a flow in the Cloud Flow Designer:
- “Manage Force.com Flow”
5. For each of the following input parameters, enter a literal value, enter a merge field, select an existing flow resource, or create a flow resource.

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Message               | The text that you want to post. Must be a string of no more than 5,000 characters.  
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]. |
| Target Name or ID     | Reference to the user, Chatter group, or record whose feed you want to post to. External users and groups in a community aren’t supported.  
• 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 |
| 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. |

6. To assign the post’s ID to a variable or sObject variable in your flow:
   a. Select Outputs.  
   b. For the Source drop-down list, select Feed Item ID.  
   c. For Value, select a flow variable or sObject variable.  
The flow stores the post’s ID in the specified variable when the post is created.
7. Click **OK**.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

## Add a Screen Element to a Flow

The Screen element can be used to set up a form to capture user input, provide choice selections, or display information to your users.

Once you've added a Screen element to your flow, configure it with the following steps.

### IN THIS SECTION:

1. **Configure a Flow Screen's General Settings**
   - Customize a Screen element’s Name, Unique Name, description, navigation, and help text by using the General Info tab.

2. **Add Fields to a Flow Screen**
   - Add input, choice, and output fields to your Screen element by using the Add a Field tab. You can also remove and reorder the fields that you’ve added.

3. **Customize Fields in a Flow Screen**
   - To configure a screen field, select it in the preview pane to open the Field Settings tab. The options on the Field Settings tab vary based on the field type.

SEE ALSO:
- Flow Elements
- Add Connectors to Define Flow Paths
Configure a Flow Screen’s General Settings

Customize a Screen element’s Name, Unique Name, description, navigation, and help text by using the General Info tab.

1. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Appears after you click Add Description.</td>
</tr>
</tbody>
</table>

2. Specify whether you want to restrict the screen’s navigation options.

At run time, the system automatically determines which buttons are relevant, depending on whether any screens precede or follow the screen in the flow path. To restrict the screen from displaying either the Previous or the Finish button, use the drop-down list. Options:

- **No navigation restrictions**—(Default) The system displays all relevant navigation buttons on the screen.
- **Don’t show Previous button**—Select this option if revisiting the previous screen triggers an action that must not be repeated, such as a credit card transaction.
- **Don’t show Finish button**—Select this option if you need the user to go back to a previous screen to continue or complete the flow.

For example, suppose the flow prompts the user to enter information to identify an existing contact. The flow then looks up the user-entered information in the database. If no matching contact is found, the flow displays a screen to tell the user to go back and try again. For that screen, select Don’t show Finish button.

3. Let flow users pause from the screen.

   a. Select Show Pause button.

   b. Customize the paused message.

   This message appears when the user clicks Pause from this screen. Once a user pauses a flow interview, only that user or an administrator can resume the interview.

4. Offer flow users help for the screen.

   a. Click to expand the Help Text section.

   b. Enter the text that you want flow users to see when they click Help for this form.

   c. Use the drop-down menu to add merge fields to your help text.

   d. Click to switch between the plain text editor and the rich text editor.
Using the rich text editor saves the content as HTML.

If you allow users to pause interviews of this flow, consider:

- Customizing the interview label, and
- Adding the Paused Flow Interviews components to the Home page layout for relevant users.

SEE ALSO:
Add a Screen Element to a Flow

Add Fields to a Flow Screen

Add input, choice, and output fields to your Screen element by using the Add a Field tab. You can also remove and reorder the fields that you’ve added.

- To add a field, double-click the field type or drag it to the preview pane.
- To remove a field, hover over the field and click 
- To reorder the fields in the preview pane, click and drag them up and down.

SEE ALSO:
Add a Screen Element to a Flow

Customize Fields in a Flow Screen

To configure a screen field, select it in the preview pane to open the Field Settings tab. The options on the Field Settings tab vary based on the field type.

IN THIS SECTION:
Customize User Input Fields on a Flow Screen
After you add a user input field to a Screen element, use the Field Settings tab to customize the field.

Customize Choice Fields on a Flow Screen
After you add a choice field to a Screen element, use the Field Settings tab to customize the field.
Customize Display Text on a Flow Screen

After you add a display text field to a flow screen, use the Field Settings tab to customize the field.

SEE ALSO:
- Add a Screen Element to a Flow
- Cross-Object Field References in Flows

Checkbox Input Fields Overview

A checkbox screen field is of type boolean. Use a checkbox screen field to offer flow users a yes-or-no choice.

For example:
- Whether to opt into a marketing campaign.
- Whether to receive a follow-up call after a purchase or case resolution.
- Whether an important policy is understood.

In this case, you can validate that the user selects the checkbox before proceeding to the next screen.

SEE ALSO:
- Customize User Input Fields on a Flow Screen
- Global Constants in Flows
- Multi-Select Choice Fields Limitations

Customize User Input Fields on a Flow Screen

After you add a user input field to a Screen element, use the Field Settings tab to customize the field.

To set up a text box, long text area, number, currency, date, date/time, password, or checkbox:
1. In the screen’s preview pane, click an input field.
2. Configure the input field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Displays to the left of the field.</td>
</tr>
<tr>
<td></td>
<td>To format the label, click [ ]</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated based on Label.</td>
</tr>
<tr>
<td></td>
<td>A unique name is limited to underscores and alphanumeric characters. It must</td>
</tr>
<tr>
<td></td>
<td>begin with a letter, not include spaces, not end with an underscore, and not</td>
</tr>
<tr>
<td></td>
<td>contain two consecutive underscores.</td>
</tr>
<tr>
<td>Input Type</td>
<td>Automatically populated based on the type of input field you selected.</td>
</tr>
</tbody>
</table>
The data type of the default value must be compatible with the field’s data type. For example, a checkbox’s default value must be of type boolean.

Available for only Currency and Number input fields. The scale is the maximum number of digits allowed to the right of the decimal point. This number can’t exceed 17. If you leave this field blank or set to zero, only whole numbers are displayed when your flow runs.

Select if the user must enter a value before moving on to the next screen.

3. Add validation to verify what the user enters.
4. Provide flow users with help for the field.
   a. Click to expand the Help Text section.
   b. Enter the text that you want flow users to see when they click next to the field.
5. Click OK.

IN THIS SECTION:

Validate Users’ Inputs with Flow Formulas
Just like with regular validation rules, you can validate what users enter in flow screens.

SEE ALSO:
Chatbox Input Fields Overview
Customize Fields in a Flow Screen
Add a Screen Element to a Flow

Validate Users’ Inputs with Flow Formulas
Just like with regular validation rules, you can validate what users enter in flow screens.

- If you reference a formula resource, the formula must have a data type of Boolean.
- If you manually enter a formula expression, it must evaluate to TRUE or FALSE.
- If the expression evaluates to TRUE, the input is valid. If the expression evaluates to FALSE, the error message is displayed to the user.
- If the user leaves the field blank and the field isn’t required, no validation is performed.

When you configure a screen input field:

1. In the Input Validation section, select Validate.
2. Enter a Boolean formula expression to define the values allowed for the field.

Note:
- The formula expression must return a Boolean value.
- If the formula expression evaluates to TRUE, the input is valid.
- If the formula expression evaluates to FALSE, the error message is displayed to the user.
3. If the user leaves the field blank, and the field is not required, the flow doesn’t validate.

3. Customize the error message that appears if the user’s input fails validation.

Click \[ \] to switch between the plain text editor and the rich text editor. Using the rich text editor saves the content as HTML.

Example:

- Validate the format of an email address:
  
  \[
  \text{REGEX}({!Email\_Address},"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}\)"
  \]

- Validate the format of a zip code:
  
  \[
  \text{REGEX}({!Zipcode},"\d{5}(-\d{4})?\)"
  \]

Customize Choice Fields on a Flow Screen

After you add a choice field to a Screen element, use the Field Settings tab to customize the field.

To set up a radio button, drop-down list, multi-select checkbox, or multi-select picklist:

1. In the Screen overlay’s preview pane, click a choice type field.

2. Configure the field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Displays to the left of the field.</td>
</tr>
<tr>
<td></td>
<td>To format the label, click [ ].</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated based on Label.</td>
</tr>
<tr>
<td></td>
<td>A unique name is limited to underscores and alphanumeric characters.</td>
</tr>
<tr>
<td></td>
<td>It must begin with a letter, not include spaces, not end with an underscore,</td>
</tr>
<tr>
<td></td>
<td>and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Value Data Type</td>
<td>Controls which choices are available in Choice Settings. For example, if you</td>
</tr>
<tr>
<td></td>
<td>choose Number you can’t use a choice that has a data type of Text.</td>
</tr>
<tr>
<td></td>
<td>You can’t change the value data type of multi-select choice fields; only text</td>
</tr>
<tr>
<td></td>
<td>is supported.</td>
</tr>
<tr>
<td>Scale</td>
<td>Available for only Currency and Number choice fields. The scale is the</td>
</tr>
<tr>
<td></td>
<td>maximum number of digits allowed to the right of the decimal point. This</td>
</tr>
<tr>
<td></td>
<td>number can’t exceed 17. If you leave this field blank or set to zero, only</td>
</tr>
<tr>
<td></td>
<td>whole numbers are displayed when your flow runs.</td>
</tr>
<tr>
<td>Required</td>
<td>If selected, users must identify a choice before they progress to the next</td>
</tr>
<tr>
<td>Default Value</td>
<td>After you’ve added all the choices for this field, identify the choice that’s</td>
</tr>
<tr>
<td></td>
<td>preselected.</td>
</tr>
</tbody>
</table>

3. In Choice Settings, add choices or dynamic choices for this field.
Note:
- Add the choices in the order you want them to appear in the screen field. You can't rearrange choices.
- Use a dynamic choice to create a lookup that automatically populates the choice options during run time.

4. Provide flow users with help for the field.
   a. Click to expand the Help Text section.
   b. Enter the text that you want flow users to see when they click next to the field.

5. Click OK.

SEE ALSO:
- Multi-Select Choice Fields Limitations
- Add a Choice to a Flow
- Add a Dynamic Choice to a Flow
- Add a Screen Element to a Flow
- Customize Fields in a Flow Screen

Customize Display Text on a Flow Screen

After you add a display text field to a flow screen, use the Field Settings tab to customize the field.

1. In the Screen overlay's preview pane, click the Display Text field to configure its settings.

2. Configure the field.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Text</td>
<td>The text to display to the flow user. Click to switch between the plain text editor and the rich text editor. Using the rich text editor saves the content as HTML. For example, use a display text field to show flow users a welcome message or a terms and conditions statement.</td>
</tr>
</tbody>
</table>

3. Click OK.

SEE ALSO:
- Customize Fields in a Flow Screen
- Add a Screen Element to a Flow
Add a Placeholder Step to a Flow

Use the Step element as a placeholder when you're not sure which element you need. Using Step elements, you can sketch out a draft of any new flow or make conceptual changes to a flow.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. From the Palette, drag the Step element into the main area of the canvas.
3. Enter a name for the Step.
4. If you haven’t already added a description, click Add Description to do so.
5. Click OK.

Steps aren’t valid elements for active flows. You can run a draft flow with Steps in it as an administrator. However, replace the Steps with other elements before you can activate the flow and let users run it.

You can convert a Step into a Screen element at any time by hovering your mouse over it and clicking ✖️.

**Note:**
- If you convert a step that has multiple connectors, all of the connectors are deleted.
- After conversion, you can no longer use the Step’s original unique name.

SEE ALSO:
- Add a Screen Element to a Flow
- Flow Elements
- Add Connectors to Define Flow Paths

Send Email from a Flow

To send email from your flow, you can call an email alert workflow action or create the email in the flow.

IN THIS SECTION:
- Call an Email Alert from a Flow
  To reuse an email configuration that exists in your organization, call an email alert workflow action. The email alert is already configured with the email's contents, recipients, and sender, so the flow only provides the record ID.
- Configure a Send Email Element in a Flow
  To send an email from a flow without using templates or configurations in other parts of Salesforce, use the Send Email element.

SEE ALSO:
- Flow Elements
Call an Email Alert from a Flow

To reuse an email configuration that exists in your organization, call an email alert workflow action. The email alert is already configured with the email’s contents, recipients, and sender, so the flow only provides the record ID.

Before you begin:

- Make sure that the email alert you want to call from your flow exists. If not, create the email alert.
- Understand the daily limits for emails sent from email alerts.
- Store the ID for the record that you want to reference in this email, such as by using a Fast Lookup element. If the email alert has any merge fields, the referenced record is the starting point for those fields.

The unique name for each email alert is prefixed with its object. The object type of the referenced record must match the object type of the email alert. For example, if you have an email alert with unique name “Owner_Changed” for accounts, that email alert appears in the Palette as Account.Owner_Changed. Because the email alert is associated with the Account object, it can reference only an account record.

1. From Setup, click Create > Workflow & Approvals > Flows, and open a new or existing flow.
2. Find the email alert that you want to call.
   a. In the Palette, click and select EMAIL ALERTS.
      The Palette filters out everything except the email alert workflow actions in your organization.
   b. Optionally, type three or more letters to search for a specific email alert.
3. Drag the desired email alert onto the canvas.
4. Enter general settings, so that you can differentiate this email alert from other email alerts in the flow.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Appears after you click Add Description.</td>
</tr>
</tbody>
</table>

5. On the Inputs tab, for Record ID select a variable that contains the ID for the record you want the email to reference. If the email alert uses any merge fields, this record is the starting point for those merge fields.
   For example, if you stored the record ID in an sObject variable called storeAccount, the value is {!storeAccount.Id}. 
6. Click **OK**.

SEE ALSO:
- Send Email from a Flow
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

Configure a Send Email Element in a Flow

To send an email from a flow without using templates or configurations in other parts of Salesforce, use the Send Email element.

Before you begin, consider storing the text for the email body in a text template.

If you want to use an email template that exists in your organization, call an email alert instead.

1. From Setup, click **Create > Workflow & Approvals > Flows** and open a new or existing flow.
2. In the Palette, type **Send** next to **Q**.
3. Drag the Send Email element onto the canvas.
4. Enter general settings, so that you can differentiate this email from other emails in the flow.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>

Appears after you click **Add Description**.

5. Specify the recipients for the email. You can use the email address parameters to specify up to five email addresses.
   a. Click **Add Row**.
   b. Select either **Email Addresses (comma-separated)** or **Email Addresses (collection)**.
   c. Enter a value for the email address parameter.
      
      You can use both parameters, so long as the combined number of addresses is five or fewer. **Email Addresses (comma-separated)** accepts string values, which include variables of type Text, manually entered strings, and input fields from a Screen element. **Email Addresses (collection)** accepts only collection variables of type Text.

6. For each input parameter, enter a literal value, enter a merge field, select an existing flow resource, or create a flow resource. To enter values for optional parameters, click **Add Row** and select the parameter.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>Text for the body of the email. Consider using a text template resource.</td>
</tr>
<tr>
<td>Subject</td>
<td>Text for the subject of the email.</td>
</tr>
</tbody>
</table>
| Email Addresses (comma-separated) | Optional. Recipients of the email. This parameter accepts string values, which include variables of type Text, manually entered strings, and input fields from a Screen element.  
For the email to send successfully, enter a value for `Email Addresses (comma-separated)` or `Email Addresses (collection)`. You can use both parameters, so long as the combined number of email addresses is five or fewer. |
| Email Addresses (collection)  | Optional. Recipients of the email. This parameter accepts only collection variables of type Text.                                                                                                           
For the email to send successfully, enter a value for `Email Addresses (comma-separated)` or `Email Addresses (collection)`. You can use both parameters, so long as the combined number of email addresses is five or fewer. |
| Sender Address                | The organization-wide email address that's used to send the email. Required only if `Sender Type` is set to `OrgWideEmailAddress`.                                                                           |
| Sender Type                   | Optional. Email address used as the email's From and Reply-To addresses. Valid values are:                                                                                                                    
• `CurrentUser`—Email address of the user running the flow. This value is the default setting.                                                                                       
• `DefaultWorkflowUser`—Email address of the default workflow user.                                                                                                                  
• `OrgWideEmailAddress`—The organization-wide email address that is specified in `Sender Address`.                                                                                  |

7. Click **OK**

SEE ALSO:

- Send Email from a Flow
- Add a Text Template to a Flow
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows
Submit a Record for Approval from a Flow

Use a Submit for Approval element to have the flow submit a specified record for approval at run time. Use the input assignments to transfer data from the flow to the approval submission.

Before you begin, store the ID for the record that you want to submit for approval.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. In the Palette, type Submit next to .
3. Drag the Submit for Approval element onto the canvas.
4. Enter general settings, so that you can differentiate this approval submission from other approval submissions in the flow.

5. For Record ID, enter the ID of the record that you want to submit for approval.
6. Enter values for optional input parameters.

### Field | Description
--- | ---
Name | Helps you identify the element on the canvas.
Unique Name | Automatically populated if empty when you fill out the Name field and press TAB.

The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

**Description** Appears after you click **Add Description**.

<table>
<thead>
<tr>
<th>Optional Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Approver IDs</td>
<td>The ID of the user to be assigned the approval request when the approval process doesn’t automatically assign the approver. Must be a collection variable of type String that includes exactly one item.</td>
</tr>
</tbody>
</table>
| Approval Process Name or ID | The unique name or ID of the specific approval process to which you want the record to be submitted. The process must have the same object type as the record you specified in Record ID. This parameter is required if Skip Entry Criteria is set to $GlobalConstant.True. If this parameter and Submitter ID aren’t set, the flow succeeds only when:
  - The approver on submit is determined automatically, and
  - The user who launched the flow is an allowed initial submitter
Make sure that:
  - The approver on submit is determined automatically, and
  - The initial submitters (for the approval processes related to this object) include all users who could launch this flow |
Create a Flow

Submit a Record for Approval from a Flow

### Optional Input Parameter | Description
---|---
Skip Entry Criteria | If set to `$GlobalConstant.True`, the record isn’t evaluated against the entry criteria set on the process that is defined in Approval Process Name or ID.
Submission Comments | Text that you want to accompany the submission. 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.
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.

7. To assign values from the approval request to variables in your flow, select the Outputs tab. These variable assignments occur when the approval request is created.
   a. Select Outputs.
   b. For the Source drop-down list, select an optional output parameter.

| Optional Output Parameter | Description |
---|---
Instance ID | The ID of the approval process that was submitted for approval. |
Instance Status | The status of the current approval process. Valid values are “Approved,” “Rejected,” “Removed,” or “Pending”. |
New Work Item IDs | The IDs of the new items submitted to the approval process. There can be 0 or 1 approval processes. |
Next Approver IDs | The IDs of the users who are assigned as the next approvers. |
Record ID | The ID of the record that the flow submitted for approval. |

c. For Value, select a flow variable or sObject variable.

8. Click OK.
Consider using a fault connector path to gracefully handle any errors.

SEE ALSO:
- Flow Elements
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

Subflows Overview

With subflows, you can simplify your overall flow architecture. Build reusable flows to perform common tasks. For example, build utility flows to capture address and credit card information, and authorize a credit card purchase amount. Then call the utility flows as needed from multiple product-ordering flows.

A subflow element references another flow and calls that flow at run time. When a flow contains a subflow element, we call it the master flow to distinguish it from the referenced flow.

At run time, the master flow calls the active version of each referenced flow by default. If a referenced flow has no active version, then the master flow calls the latest version of the referenced flow. To run only the latest version of each referenced flow, use one of the following methods:

- Open the master flow in the Cloud Flow Designer, and click Run with Latest in the button bar.
- Run the master flow from the flow URL appended with ?latestSub=true.

**Note:** Only flow administrators can run inactive flows. For other users, the flow fails at run time if a subflow element tries to call a flow with no active version.

The Palette lists the other flows in your organization.

SEE ALSO:
- Add a Subflow Element to a Flow

Add a Subflow Element to a Flow

Use a subflow element to have the flow call another flow at run time. Use subflow input and output assignments to transfer data between the master flow and the referenced flow.

Before you begin, create or install the other flow that you want to call.

A subflow element references another flow and calls that flow at run time. When a flow contains a subflow element, we call it the master flow to distinguish it from the referenced flow.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. Find the flow you want to reference.
   a. In the Palette, click and select FLOWS.
      The Palette filters out everything except the flows in your organization.
   b. Optionally, type three or more letters to search for a specific flow.
3. Drag the flow that you want to call onto the canvas.
4. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
</tbody>
</table>
| Unique Name | Automatically populated if empty when you fill out the Name field and press TAB.  
               The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. |

Description | Appears after you click Add Description. |

5. On the Inputs tab, assign values to variables in the referenced flow.
These variable assignments occur when the master flow calls the referenced flow at run time.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
</table>
| Target        | Referenced flow's variable whose value you want to set.  
                By default, this drop-down list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the drop-down list contains the variables of the latest version of the referenced flow. You can, however, view or select from the inputs and outputs of other versions of the referenced flow. |
| Source        | Master flow's resource or value to assign to the target.  
                Options:  
                • Select an existing flow resource, such as a variable, constant, or user input.  
                • Select CREATE NEW to create a flow resource.  
                • Manually enter a literal value or merge field. |

6. On the Outputs tab, assign values from the referenced flow's variables to the master flow's variables.
These variable assignments occur when the referenced flow finishes running.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
</table>
| Source        | Referenced flow's variable whose value you want to assign to the target.  
                By default, this drop-down list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the drop-down list contains the variables of the latest version of the referenced flow. |
| Target        | Master flow's variable whose value you want to set. |
7. Click **OK**.

SEE ALSO:
- Flow Elements
- View Inputs and Outputs of Other Referenced Flow Versions
- Subflows Overview
- Customize What Happens When a Flow Fails
- Add Connectors to Define Flow Paths
- Cross-Object Field References in Flows

**View Inputs and Outputs of Other Referenced Flow Versions**

While configuring a subflow element, you can view the variables of a specified version of the referenced flow.

Before you begin, add a subflow element to your flow.

From a subflow element, you can assign values to only the referenced flow’s variables that allow input access. Similarly, you can assign values from only the referenced flow’s variables that allow output access. The **Input/Output Type** of the variable determines this access. To change the variable’s **Input/Output Type**, open the referenced flow to **edit the variable**.

By default, this drop-down list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the drop-down list contains the variables of the latest version of the referenced flow.

To populate the drop-down lists with the variables of another version of the referenced flow, complete the following steps. Do the same to view the descriptions of the referenced flow’s variables.

1. On the subflow overlay, expand the **Input/Output Variable Assignments** section.
2. Click **View input/output of other versions**.
3. Use one or more of the following options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a <strong>Version</strong> number in the left pane.</td>
<td>The Inputs and Outputs tabs display the variables in the selected version of the referenced flow.</td>
</tr>
<tr>
<td>Select the <strong>Inputs</strong> tab or the <strong>Outputs</strong> tab.</td>
<td>The tab displays:</td>
</tr>
<tr>
<td></td>
<td>• The variables available for input or output assignment in the selected version of the referenced flow.</td>
</tr>
<tr>
<td></td>
<td>• The data type of each variable.</td>
</tr>
<tr>
<td></td>
<td>• The description, if any, of each variable.</td>
</tr>
<tr>
<td><strong>Click OK.</strong></td>
<td>The subflow overlay’s drop-down lists for selecting the referenced flow’s variables are populated with the variables of the selected <strong>version</strong> of the referenced flow.</td>
</tr>
</tbody>
</table>

**EDITIONS**

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

**USER PERMISSIONS**

To open, edit, or create a flow in the Cloud Flow Designer:
- “Manage Force.com Flow”
When you configure subflow input and output assignments, you can specify variables from any version of the referenced flow. This way, you can develop both the master flow and referenced flow in parallel, while keeping another version of the referenced flow active for its users. When you save the master flow, however, the Cloud Flow Designer validates against the currently active version of the referenced flow. If that flow doesn’t have an active version, the latest version is validated. If you see validation messages about variables that couldn’t be found or that were configured differently in the referenced flow, you can still save the flow. Nevertheless, resolve all validation errors before you activate the master flow.

SEE ALSO:
Subflows Overview

Configure a Flow to Wait for Events

Use a Wait element to automate processes that require a waiting period. For example, configure a flow to wait until three days before a contract expires, when the flow sends an email reminder to the contract owner.

Note:
- Flows that contain Wait elements must be autolaunched. If a flow includes Wait elements and screens, choice, or dynamic choices, it can’t be activated or run.
- Before you add a Wait element to your flow, understand the special behavior and limitations. See Time-based Flow Limitations on page 10 for details.

To add or configure a Wait element:
1. From Setup, click Create > Workflow & Approvals > Flows, and open a new or existing flow.
2. From the Palette, drag the Wait element onto the canvas.
3. Enter the general settings for the element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Helps you identify the element on the canvas.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB.</td>
</tr>
<tr>
<td></td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have</td>
</tr>
<tr>
<td></td>
<td>the same unique name, provided they are used in different flows. A unique name is limited to underscores</td>
</tr>
<tr>
<td></td>
<td>and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore,</td>
</tr>
<tr>
<td></td>
<td>and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Appears after you click Add Description.</td>
</tr>
</tbody>
</table>

4. Define the events that the flow waits for before it proceeds. For each event:
   a. Provide the general settings for the event.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name appears on the connector that’s associated with this event.</td>
</tr>
</tbody>
</table>
### Configure a Flow to Wait for Events

**Create a Flow**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
</tbody>
</table>
| Event Type      | Determines whether the flow is waiting for:  
• An absolute time (such as 3 days from now), or  
• A time relative to a date/time field on a Salesforce record (such as 3 days after an opportunity closes) |

**b.** Fill out the parameters under Event Conditions.

Based on the event type that you selected, different input parameters are available. Parameters that appear automatically are required.

c. If you want the flow to wait for the event only when certain conditions are met, select **Wait for this event only if additional conditions are met**. Then define those conditions.

The flow waits for the event only if the waiting conditions evaluate to `true`. For details, see Define Flow Conditions on page 74.

d. Assign the event’s outputs to flow variables.

5. Configure the default path.

This path is executed only if all of the events have unmet waiting conditions. If one or more of your events don’t have waiting conditions set, the default path is never executed.

a. Click **[Default Path]**.

b. Optionally, provide a custom name for this path by replacing the predefined value.

The name displays on the Wait element’s default connector.

6. Click **OK** to save your changes.

After you define your events, connect the Wait element to other elements on the canvas to indicate what the flow does when:

• Each event is the first to occur. One connector (1) is available for each event that’s defined in the Wait element.

• There are no more events to wait for, because the waiting conditions for every event are unmet. One connector (2) is available for the Wait element’s default path.

• An error occurs related to the Wait element. One connector (3) is available for the Wait element’s fault path, and it’s always labeled **FAULT**.
If the flow waits for multiple events, consider returning the flow path to the Wait element again so that the flow waits for the other events. If you return the flow path to the Wait element, consider using waiting conditions to control when the flow waits for each event. For an example, see Sample Flow That Waits for Many Events on page 72.

IN THIS SECTION:

Event Types Overview

Event Type drives the fields that you use to define an event in a Wait element. The available event types are both alarms, which consist of a date/time value—the base time—and an optional offset from that time.

Waiting Conditions Overview

Each event that you define in a Wait element has optional waiting conditions. These conditions must be met for the flow interview to wait for that event at run time.

Sample Flows That Wait for Events

You can configure a flow to wait for events in various ways.

SEE ALSO:

Time-based Flow Limitations
Customize What Happens When a Flow Fails
Add Connectors to Define Flow Paths
Cross-Object Field References in Flows
Flow Elements

Event Types Overview

Event Type drives the fields that you use to define an event in a Wait element. The available event types are both alarms, which consist of a date/time value—the base time—and an optional offset from that time.

The base time, which is always required, is the date/time value from which the alarm is based. If there’s no offset for the alarm, the alarm is set to the exact value of the base time. The base time can be composed of one or multiple fields, based on the event type that you choose.

The offset, which is optional, is the amount of time before or after the base time at which the alarm occurs. An offset is always composed of two fields: Offset Number and Offset Unit. For example, if you want your alarm to occur three days after the base time, the number is 3 and the unit is Days.

IN THIS SECTION:

Absolute Time Alarms

An absolute time alarm waits for a defined time that’s based off an absolute date/time value. For example, you can use this event type in a Wait element to do something a day after the flow interview starts to wait.

Relative Time Alarms

A relative time alarm waits for a defined time that’s based off a date/time field on a record. For example, you can use this event type to do something three days before a contract ends.

SEE ALSO:

Configure a Flow to Wait for Events
Absolute Time Alarms

An absolute time alarm waits for a defined time that’s based off an absolute date/time value. For example, you can use this event type in a Wait element to do something a day after the flow interview starts to wait.

When you configure a Wait element in a flow:

- Define what the flow is waiting for
- Assign information from the event after it occurs to flow variables

Event Conditions

The following parameters are available to define events with an Event Type of Alarm: Absolute Time.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Time</td>
<td>A date/time value. If you enter values for Offset Number and Offset Unit, this field value is the base for the offset. You can manually enter a date/time value or reference a merge field or flow resource.</td>
<td>{$Flow.CurrentDate}</td>
</tr>
<tr>
<td>Offset Number</td>
<td>Optional. The number of days or hours to offset Base Time. Required if you set a value for Offset Unit. The value must be a manually entered integer. You can't use a merge field or flow resource for this value. To set the alarm to occur before Base Time, use a negative number. To set the alarm to occur after Base Time, use a positive number.</td>
<td>-3</td>
</tr>
<tr>
<td>Offset Unit</td>
<td>Optional. The unit to offset Base Time. Required if you set a value for Offset Number. Manually enter Days or Hours. You can't use a merge field or flow resource for this value.</td>
<td>Days</td>
</tr>
</tbody>
</table>

For an example of a flow that waits for an absolute time alarm, see Sample Flow That Waits for a Single Event.

Event Outputs

Reference information from the event in your flow by assigning its outputs to flow variables.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Time</td>
<td>The actual time at which the event occurred and the flow interview resumed.</td>
<td>11/26/2014 10:12 AM</td>
</tr>
</tbody>
</table>
The status of the event when the flow interview resumed. After an event occurs, Salesforce delivers 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.

SEE ALSO:
- [Event Types Overview](#)

**Relative Time Alarms**

A relative time alarm waits for a defined time that’s based off a date/time field on a record. For example, you can use this event type to do something three days before a contract ends.

When you configure a Wait element in a flow:

- Define what the flow is waiting for
- Assign information from the event after it occurs to flow variables

**Event Conditions**

The following parameters are available to define events with an Event Type of Alarm: Relative Time.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Object Type</strong></td>
<td>The API name of the object whose field you want to base the alarm on. See <a href="#">Supported Objects</a> on page 65. You must manually enter a string. You can’t use a merge field or flow resource for this value.</td>
<td><strong>Contract</strong></td>
</tr>
</tbody>
</table>
| **Base Date/Time Field** | The API name for a date or date/time field on the specified object. If you enter values for Offset Number and Offset Unit, this field value is the base for the offset.  
Manually enter a string. | **EndDate** |
| **Record ID**        | ID of the record that the alarm is based on. The record’s object type must match Object Type. You can enter a string, merge field, or flow resource. | {!ContractId} |
| **Offset Number**    | Optional. The number of days or hours to offset Base Date/Time Field. Required if you set a value for Offset Unit. The value must be a manually entered integer. You can’t use a merge field or flow resource for this value. | -3      |
To set the alarm to occur before Base Date/Time Field, use a negative number. To set the alarm to occur after Base Date/Time Field, use a positive number.

For examples of flows that wait for relative time alarms, see Sample Flow That Waits for Only the First Event or Sample Flow That Waits for Many Events.

Event Outputs
Reference information from the event in your flow by assigning its outputs to flow variables.

Supported Objects
You can create a relative time alarm for any custom object or any of the following standard objects.

- Account
- Asset
- Campaign
- CampaignMember
- Case
- CaseComment
- Certification
- CertificationDef
- CertificationSectionDef
- CertificationStep
- CertificationStepDef

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Time</td>
<td>The actual time at which the event occurred and the flow interview resumed.</td>
<td>11/26/2014 10:12 AM</td>
</tr>
<tr>
<td>Event Delivery Status</td>
<td>The status of the event when the flow interview resumed. After an event occurs, Salesforce delivers the event to the flow that's waiting for it, so that the flow knows to resume. Valid values are:</td>
<td>Delivered</td>
</tr>
<tr>
<td></td>
<td>• Delivered: The event was successfully delivered.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Invalid: An error occurred during delivery, but the flow successfully resumed.</td>
<td></td>
</tr>
</tbody>
</table>
Create a Flow

• Contact
• Contract
• ContractLineItem
• DandBCompany
• DuplicateRecordItem
• DuplicateRecordSet
• EmailMessage
• Entitlement
• EntitlementContact
• EnvironmentHubMemberRel
• Event
• ExternalEventMapping
• FeedItem
• Goal
• GoalLink
• Idea
• IdentityProvEventLog
• Lead
• LiveAgentSession
• LiveChatTranscript
• LiveChatTranscriptEvent
• LiveChatTranscriptSkill
• Macro
• MacroAction
• MacroInstruction
• Metric
• MobileDeviceCommand
• Opportunity
• OpportunityLineItem
• OpportunitySplit
• OpportunityTeamMember
• Order
• OrderItem
• Organization
• PersonAccount
• Product2
• ProfileSkill
• ProfileSkillEndorsement
• ProfileSkillUser
• Question

Configure a Flow to Wait for Events
Create a Flow

Configure a Flow to Wait for Events

- QuickText
- Quote
- QuoteLineItem
- Reply
- SOSSession
- SOSSessionActivity
- ServiceContract
- SignupRequest
- Site
- SocialPersona
- SocialPost
- Solution
- SsoUserMapping
- StreamingChannel
- Task
- UsageEntitlementPeriod
- User
- UserLicense
- UserProvisioningRequest
- WorkBadge
- WorkBadgeDefinition
- WorkCoaching
- WorkFeedback
- WorkFeedbackQuestion
- WorkFeedbackQuestionSet
- WorkFeedbackRequest
- WorkFeedbackTemplate
- WorkGoal
- WorkPerformanceCycle
- WorkReward
- WorkRewardFund
- WorkThanks
- WorkUpgradeAction
- WorkUpgradeCustomer
- WorkUpgradeUser

SEE ALSO:

Event Types Overview
Waiting Conditions Overview

Each event that you define in a Wait element has optional waiting conditions. These conditions must be met for the flow interview to wait for that event at run time.

When an interview encounters a Wait element, it checks the waiting conditions for each event to determine which events to wait for. If the waiting conditions aren’t met for an event, the interview doesn’t wait for that event. If all events have unmet waiting conditions, the interview executes the default path.

**Tip:** Add a default path if all of the events have waiting conditions set and you want the flow to proceed when the waiting conditions for every event are met.

**Example:** Here are two scenarios in which you would use waiting conditions.

- 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. The date on which you send the email depends, however, 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.

  In this example, you would create two events. The event for hot accounts occurs 30 days before the contract’s end date. Its waiting conditions would check if the Rating for the contract’s account is equal to “Hot.”

  The second event occurs 15 days before the contract’s end date. Its waiting conditions would check if the Rating for the contract’s account is not equal to “Hot.”

  When a flow interview executes the Wait element during run time, the interview checks the waiting conditions for each event. It only waits for the events whose waiting conditions are met. 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 Waits for Many Events on page 72.

**SEE ALSO:**
- Operators in Flow Record Filters
- Configure a Flow to Wait for Events
- Event Types Overview

Sample Flows That Wait for Events

You can configure a flow to wait for events in various ways.
IN THIS SECTION:

Sample Flow That Waits for a Single Event
This flow waits for a single event. The base time for the event in this example, which is an absolute alarm, is the `{!$Flow.CurrentDateTime}` system variable.

Sample Flow That Waits for Only the First Event
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 uses relative time alarms.

Sample Flow That Waits for Many Events
This flow waits for many events to occur, rather than just the first event. The base times for these events are field values, so this example uses relative time alarms.

SEE ALSO:
Configure a Flow to Wait for Events

Sample Flow That Waits for a Single Event
This flow waits for a single event. The base time for the event in this example, which is an absolute alarm, is the `{!$Flow.CurrentDateTime}` system 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.

Example
This flow already contains the following populated variables.

- `{!customerEmail}` contains the email address for the customer
- `{!creatorEmail}` contains the email address for the flow’s creator

The flow activates a contract (1) and then waits (2).

Within the Wait element, a single event is defined (1 day after activated). The flow sends the feedback request one day after the contract is activated, so use an absolute time alarm. The base time is the `{!$Flow.CurrentDateTime}` system variable (3), and the offset is one day (4).
Because there’s only one event and you only want the feedback request to be sent once, don’t set any waiting conditions for this event. 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:
  Configure a Flow to Wait for Events
  Absolute Time Alarms

Sample Flow That Waits for Only the First Event

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 uses relative time alarms.

You’re designing a flow that reminds account owners to follow up with their customers a week before either the account’s renewal date or the contract’s end date. The flow sends a reminder email for only the date that occurs first.

Example

This flow already contains these populated variables.

- {accountId} contains the ID for the account
- {contractId} contains the ID for the contract
- {accountOwner} contains the ID for the account’s owner
- {ownerEmail} contains the account owner’s email address

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.
The Wait element defines two relative alarm events.

![Diagram showing the Flow with Wait element and related components]

Tip: Every alarm event is comprised of a base time and an offset. With relative time alarms, 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 relative time alarms works the same as it does for absolute time alarms. The flow needs to 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).

![Configuration panel for the first event]

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).

![Configuration panel for the second event]
You only want to send one follow-up reminder and the flow always waits for both events, so neither of these events need waiting 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:
- Configure a Flow to Wait for Events
- Relative Time Alarms
- Configure a Flow to Wait for Events
- Relative Time Alarms
- Waiting Conditions Overview

**Sample Flow That Waits for Many Events**

This flow waits for many events to occur, rather than just the first event. The base times for these events are field values, so this example uses relative time alarms.

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.

**Example**

This flow already contains these populated variables.

- `{!contract}` is an sObject variable that contains the contract’s Id and OwnerId
- `{!oneWeekVisited}` is a Boolean variable whose default value is `{!$GlobalConstant.False}`
- `{!twoWeeksVisited}` is a Boolean variable whose default value is `{!$GlobalConstant.False}`

Before the flow executes the Wait element, it looks up and stores the contract’s Id and OwnerId.

Because the flow sends the reminder emails both two weeks and a week before the contract’s end date, the Wait element defines two relative alarm events.
Tip: Every alarm event is comprised of a base time and an offset. With relative time alarms, 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 relative time alarms works the same as it does for absolute time alarms. The flow needs to know the unit (either Days or Hours) and the number of those units. To wait for a number of days or hours before the base time, 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 {!contract.Id} (2). The offset is -14 days (3) to represent two weeks.

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 waiting conditions. When an interview executes a Wait element, it first checks the waiting conditions for each event to determine whether to wait for those events. If an event has waiting conditions set and those conditions aren’t met, the interview doesn’t wait for that event.

For the first 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.

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 event has occurred yet, since the variable is changed to true only in that event’s path. If that 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 (4); the offset is -7 days (5) to represent a week.
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.

Like with the first event, 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 of the events have waiting conditions that aren’t met, the flow executes the default event 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.

### Define Flow Conditions

Use flow conditions to control when a flow takes a specific decision outcome or waits for a specific wait event.

Before you begin, create the Decision or Wait element to add conditions to. To add conditions to a wait event, select **Wait for this event only if additional conditions are met**.

1. **Set up the conditions.**

   Click **Add Condition** to add another condition, or click **Remove** to remove a condition. 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. For details, see [Operators in Flow Conditions](#) on page 101.
Value | The Variable and Value in the same row must have compatible data types.

Options:
- Select an existing flow resource, such as a variable, constant, or user input.
- Select **CREATE NEW** to create a flow resource.
- Manually enter a literal value or merge field.
2. Identify the logic between the conditions.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>All conditions must be true (AND)</td>
<td>If one of the conditions is false, the flow evaluates the next outcome’s conditions.</td>
</tr>
<tr>
<td>One condition must be true (OR)</td>
<td>If one of the conditions is true, the flow immediately takes this outcome’s path.</td>
</tr>
</tbody>
</table>
| Advanced logic (Combination of ANDs and ORs)     | Custom logic. When you select this option, provide the customized Logic by entering a text string. Use:  
  • Numbers to refer to each condition  
  • AND or OR to identify whether all or just one of the conditions must true  
  • Parentheses to group parts of the string together  
  | Tip: If you enter AND, it’s the same as if you selected All conditions must be true (AND).  
If you enter OR, it’s the same as if you selected One condition must be true (OR). If you enter any other logic, make sure that you include a number for each condition.  
For example, for 1 AND (2 OR 3), the flow evaluates whether the first condition is true and either the second or third condition is true. |
<table>
<thead>
<tr>
<th><strong>Flow Resource</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice</td>
<td>Represents an individual value that can be used in choice screen fields. You can create a choice from the Resources tab or while defining screen choice fields.</td>
</tr>
<tr>
<td>Collection Variable</td>
<td>Represents a collection of values with a single data type. Use a collection variable as a container in the flow to store and reference multiple values at once. You can create a collection variable from the Resources tab.</td>
</tr>
<tr>
<td>Constant</td>
<td>Stores a fixed value. You can create a constant from the Resources tab.</td>
</tr>
<tr>
<td>Dynamic Choice</td>
<td>Looks up data from an object’s records and dynamically generates a set of choices for screen fields at run time. When referenced as a resource, a dynamic choice's value is determined by the most recent user selection of a choice within the generated set. You can create a dynamic choice from the Resources tab or while defining screen choice fields.</td>
</tr>
<tr>
<td>Element</td>
<td>Any element that you add to the flow is available as a resource with the was visited operator in outcome criteria. An element is considered visited if the element has already been executed in the flow interview. Any element that you add to the flow that supports a fault connector is available as a Boolean resource. If the element has already been 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.</td>
</tr>
<tr>
<td>Formula</td>
<td>Calculates a value from other resources in the flow. You can create a formula from the Resources tab.</td>
</tr>
<tr>
<td>Global Constant</td>
<td>Fixed, system-provided values, such as EmptyString, True, and False, that can be assigned as the values of flow resources.</td>
</tr>
<tr>
<td>Global Variables</td>
<td>System-provided variables that reference information about the organization or running user, such as the user’s ID or the API session ID. In Visual Workflow, global variables are available in only flow formulas.</td>
</tr>
<tr>
<td>Outcome</td>
<td>If you add a Decision element to the 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.</td>
</tr>
<tr>
<td>Picklist Values</td>
<td>System-provided values that are available as resources only in Assignment and Decision elements when selecting values for—or to compare against—picklist fields in sObject variables.</td>
</tr>
<tr>
<td>Screen Field</td>
<td>Any screen field that you add to the flow is available as a resource. The resource value depends on the type of screen field. The value for a screen input field is what the user enters. The value for a screen choice field is the stored value of the choice that the user selects. The value for a screen output field is the text that’s displayed to the user.</td>
</tr>
<tr>
<td>SObject Collection Variable</td>
<td>Represents a collection of records for a specified object. Use an sObject collection variable as a container in the flow to store and reference field values for multiple records. You can create an sObject collection variable from the Resources tab.</td>
</tr>
<tr>
<td>SObject Variable</td>
<td>Represents a record for a specified object. Use an sObject variable as a container in the flow to store, update, and reference field values for a record. You can create an sObject variable from the Resources tab.</td>
</tr>
<tr>
<td>System Variable</td>
<td>System-provided values about the running flow interview that can be referenced as resources, such as {!$Flow.CurrentDate}, {!$Flow.CurrentDateTime}, and {!$Flow.FaultMessage}.</td>
</tr>
<tr>
<td>Text Template</td>
<td>Stores formatted text with merge fields that reference flow resources. You can create a text template from the Resources tab.</td>
</tr>
</tbody>
</table>
Create a Flow

Add a Choice to a Flow

A Choice resource is a stand-alone choice option. Use a choice to create a choice field in a screen, such as a dropdown list, and then reference or reuse it throughout your flow.

For example, if your flow asks users to 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 pick from a dropdown list.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. From the Resources tab, double-click Choice.
3. Fill out the fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>The value that the users see when this choice is used in a screen choice field. To format the label, click **.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Automatically populated based on Label. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>If you haven’t already added a description, click Add Description to do so.</td>
</tr>
<tr>
<td>Value Data Type</td>
<td>Controls which choice fields this choice can be used in. For example, you can’t use a Text choice in a Currency radio button field.</td>
</tr>
<tr>
<td>Scale</td>
<td>Available for only Currency and Number choices. The scale is the maximum number of digits allowed to the right of the decimal point. This number can’t exceed 17. If you leave this field blank or set to zero, only whole numbers are displayed when your flow runs.</td>
</tr>
<tr>
<td>Stored Value</td>
<td>If the user selects this choice, the choice field has this value. If a user leaves a choice blank or unselected, its stored value is set to <strong>null</strong>.</td>
</tr>
</tbody>
</table>
4. If you selected Show Input on Selection:
   a. Configure the input field.

   Field | Description
   --- | ---
   Label | The value that the users see when this choice is used in a screen choice field.
   | To format the label, click [ ].
   Required | Select if the user must enter a value before moving on to the next screen.

   b. Add validation to verify what the user enters.

5. Click OK.

   After you create a choice, it shows up on the Explorer tab, where you can edit it as needed.

SEE ALSO:
- Flow Resources
- Add a Dynamic Choice to a Flow
- Cross-Object Field References in Flows

Add a Collection Variable to a Flow

Create a collection variable to store multiple values that have the same data type, such as a group of email addresses. After the collection variable is populated, you can use it throughout your flow.

Before you can populate a collection variable with values, create the variable.

1. From Setup, click Create > Workflow & Approvals > Flows and open a new or existing flow.
2. From the Resources tab, double-click Collection Variable.
3. Fill out the fields.

   Field | Description
   --- | ---
   Unique Name | The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
<table>
<thead>
<tr>
<th>Description</th>
<th>Helps you determine when to assign or use this collection variable as you set up your flows. The description appears in the Description pane when the variable is selected in the Explorer tab. If the Input/Output Type isn't Private, you can see this description from another flow's subflow element.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Determines the type of values that can be assigned to the collection variable.</td>
</tr>
<tr>
<td>Input/Output Type</td>
<td>Determines whether the collection variable can be accessed outside the flow.</td>
</tr>
</tbody>
</table>
|             | • Private—Can be assigned and used only within the flow  
|             | • Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs  
|             | • Output—Can be accessed from Visualforce controllers 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, Record or Fast Create, Record or Fast Lookup, and Apex Plug-in.  
|             | The default value of the field is Private. |

**Warning:** 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, Visualforce controllers, subflows, and processes.

4. Click OK

**IN THIS SECTION:**

**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 Record Lookup or Fast Lookup element to populate a collection variable, but there are some workarounds.

**Sample Flow That Populates a Collection Variable**
Populate a collection variable by populating an sObject collection variable. Then individually assign the sObject collection variable’s values to the collection variable.

**SEE ALSO:**
Flow Resources  
Add an SObject Collection to a Flow  
Add a Loop Element to a Flow  
Cross-Object Field References in Flows
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 Record Lookup or Fast Lookup element to populate a collection variable, but there are some workarounds.

To use values from outside the flow, set the collection variable’s Input/Output Type to “Input” and then use URL parameters, Visualforce controllers, or subflow inputs. When the values are coming from outside the flow, the values can be set only at the start of the flow interview.

To store values in a collection variable from within the flow:

<table>
<thead>
<tr>
<th>To add values that are stored in...</th>
<th>Do this...</th>
<th>For more information</th>
</tr>
</thead>
</table>
| A screen field                    | Add the field’s entered or stored value to a collection variable by using an Assignment element | • Choice fields  
• Input fields  
• Output fields  
• Assignments |
| A variable                         | Add the variable’s stored value to a collection variable by using an Assignment element | • Variables  
• Assignments |
| An sObject variable                | Add one of the sObject variable’s stored field values to a collection variable by using an Assignment element | • sObject variables  
• Assignments |
| An sObject collection variable     | Loop through the sObject collection variable. Within the loop, add one of the loop variable’s stored field values to a collection variable by using an Assignment element | • sObject collection variables  
• Loops  
• Assignments |

SEE ALSO:

Add a Collection Variable to a Flow
Sample Flow That Populates a Collection Variable

Sample Flow That Populates a Collection Variable

Populate a collection variable by populating an sObject collection variable. Then individually assign the sObject collection variable’s values to the collection variable.

Scenario

In this scenario, you’re designing a flow to send an email to every employee who lives in San Francisco.
The Send Email element allows you to 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 Fast Lookup or Record Lookup to populate collection variables. First populate a User-based sObject collection variable with field values, including Email, from the employees who live in San Francisco. Then add those emails to the collection variable. Once the collection variable is populated, you simply use the collection variable as the value for the Send Email element’s Email Addresses (collection) parameter.

**Example:** This flow already contains these resources.
- A User-based sObject collection variable called employeesInSF
- A User-based sObject variable called loopVariable
- A Text-based collection variable called emails_employeesInSF

The example flow:
1. Finds all user records whose City is "San Francisco" and populates employeesInSF with those records’ Email.
   Because employeesInSF is an sObject collection variable, use a Fast Lookup element to populate the variable.
2. Loops through the sObject 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 a Collection Variable to a Flow
- Add Values to a Collection Variable
Add a Constant to a Flow

Use the Constant resource to define a fixed value that you can use throughout your flow.

1. From Setup, click **Create > Workflow & Approvals > Flows**
2. From the Resources tab, double-click **Constant**.
3. Fill out these fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Helps you determine when to assign or use this constant as you set up your flows. The description appears in the Description pane when the constant is selected in the Explorer tab.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Determines the types of values that can be assigned to the constant.</td>
</tr>
<tr>
<td>Value</td>
<td>The value for the constant.</td>
</tr>
</tbody>
</table>

4. Click **OK**.

SEE ALSO:
- **Flow Resources**
Add a Dynamic Choice to a Flow

A dynamic choice looks up data from an object’s records and dynamically generates a set of choices at runtime. Throughout your flow, you can reuse a Dynamic Choice resource for choice fields, and you can reference the stored value of a Dynamic Choice resource. The stored value is determined by the most recent user selection of a choice within the generated set.

For example, in a flow, your Acme, Inc. employees identify which division they work in. Instead of manually entering the divisions as separate choices, create a dynamic choice to look up all divisions where “Acme” appears in the name. Then your employees can select from a dynamically generated choice list of all division names containing “Acme”.

1. From Setup, click Create > Workflow & Approvals > Flows
2. Create a dynamic choice using one of the following methods:
   - From the Resources tab, double-click Dynamic Choice.
   - From the Screen overlay, add or select a choice field in the preview pane. In the Choice Settings for that field, select CREATE NEW > Dynamic Choice.
   - From any resource-selecting drop-down list, select CREATE NEW > Dynamic Choice.
3. Enter the general settings for the choice.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Appears after you click Add Description.</td>
</tr>
<tr>
<td>Value Data Type</td>
<td>Data type of the choice’s stored value.</td>
</tr>
</tbody>
</table>

4. Select the object whose records contain the data for the generated choices.
5. Specify the filter criteria for selecting the record from the database.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Field for filtering records.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available filter operators depend on the data type selected for Field. For details, see Operators in Flow Record Filters on page 109.</td>
</tr>
<tr>
<td>Value</td>
<td>The Field and Value in the same row must have the same data type. Options:</td>
</tr>
<tr>
<td></td>
<td>• Select an existing flow resource, such as a variable, constant, or user input.</td>
</tr>
<tr>
<td></td>
<td>• Select CREATE NEW to create a flow resource.</td>
</tr>
<tr>
<td></td>
<td>• Manually enter a literal value or merge field.</td>
</tr>
</tbody>
</table>
If you remove all filter rows, a choice is generated for every record of the selected object.

6. Select the records’ fields whose values you want to use in the generated choices.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice Label</td>
<td>Record’s field whose value <em>appears</em> as the choice label at run time.</td>
</tr>
<tr>
<td></td>
<td>Make sure to choose a field that contains data. Otherwise, users see blank generated choices at run time.</td>
</tr>
<tr>
<td>Choice Stored Value</td>
<td>Record’s field whose value is <em>stored</em> when the user selects this choice at run time.</td>
</tr>
<tr>
<td></td>
<td>The available options depend on the Value Data Type of the Dynamic Choice resource.</td>
</tr>
<tr>
<td></td>
<td>If you don’t fill out this field, the stored value is set to null.</td>
</tr>
</tbody>
</table>

For example, you’re designing a support flow for a computer hardware manufacturer. You want to let users identify a product to find its latest updates. You create a dynamic choice 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.

7. To sort the generated choices:
   a. Select Sort results by.
   b. Select the record’s field to sort by. Only sortable fields appear in the drop-down list.
   c. Select the sort order, either ascending or descending.

8. To limit the number of generated choices:
   a. Select Limit number of choices to.
   b. Enter the maximum number of choices (up to 200) to include in the generated set.

9. Set up field assignments that occur at run time when the user selects a choice from the generated set. Use flow variables to store field values from the record associated with the user-selected choice.

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
<td>Record’s field whose value you want to assign to a variable.</td>
</tr>
<tr>
<td>Variable</td>
<td>Select an existing flow variable, or select CREATE NEW to create a variable.</td>
</tr>
</tbody>
</table>

Note: When a multi-select choice field uses a dynamic choice, only values from the last record that the user selects are stored in the flow variables. If multiple multi-select choice fields on one screen use the same dynamic choice, the first multi-select choice field on the screen determines the flow variable assignments.

Let’s return to our example where the flow user selects a product name from the generated set of choices. Elsewhere in the flow, you want to display the associated product ID and description. Assign the ID and Description field values from the user-selected record to flow variables.

10. Click OK
After you create a dynamic choice, it is available on the Explorer tab.

SEE ALSO:
- Operators in Flow Record Filters
- Add a Screen Element to a Flow
- Add a Choice to a Flow
- Flow Resources

Add a Formula to a Flow

Use a Formula resource to calculate a value using Salesforce functions and resources in your flow.

1. From Setup, click **Create > Workflow & Approvals > Flows.**
2. From the Resources tab, double-click **Formula.**
3. Fill out these fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Helps you determine when to assign or use this formula as you set up your flows.</td>
</tr>
<tr>
<td>Value Data Type</td>
<td>The data type for the value calculated by the formula.</td>
</tr>
<tr>
<td>Scale</td>
<td>Optional. Appears when Value Data Type is Number or Currency.</td>
</tr>
<tr>
<td></td>
<td>The scale is the maximum number of digits allowed to the right of the decimal point. This number can't exceed 17. If you leave this field blank or set to zero, only whole numbers are displayed when your flow runs.</td>
</tr>
<tr>
<td>Formula</td>
<td>Enter your formula.</td>
</tr>
<tr>
<td></td>
<td>Use ▼ to select resources from your flow. Use the CREATE NEW section of the drop-down list to create resources, if necessary.</td>
</tr>
<tr>
<td></td>
<td>For a complete list of operators and functions for building formulas in Salesforce, see “Formula Operators and Functions Overview” in the Salesforce Help.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> Some formula operators are not supported in the Cloud Flow Designer. If your formula contains an unsupported function or operator, it returns a null value. See Flow Formula Limitations for the complete list.</td>
</tr>
</tbody>
</table>
4. Click OK.

SEE ALSO:
- Flow Resources
- Cross-Object Field References in Flows

## Global Constants in Flows

Depending on the data type of the resource you’re working with, resource drop-down lists include a section called GLOBAL CONSTANT. Global constants are system values that, depending on the context of the drop-down list, let you assign `EmptyString`, `True`, or `False` as the value for that field.

<table>
<thead>
<tr>
<th>Global Constant</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{$GlobalConstant.True}</code></td>
<td>Boolean</td>
</tr>
<tr>
<td><code>{$GlobalConstant.False}</code></td>
<td>Boolean</td>
</tr>
<tr>
<td><code>{$GlobalConstant.EmptyString}</code></td>
<td>Text</td>
</tr>
</tbody>
</table>

**Note:** At run time, `{!$GlobalConstant.EmptyString}` and `null` are treated as separate, distinct values. For example:

- If you leave a text field or resource value blank, that value is `null` at run time. If you instead want the value to be treated as an empty string, set it to `{!$GlobalConstant.EmptyString}`.
- For flow conditions, use the `is null` operator to check whether a value is `null`. If the condition compares two text variables, make sure that their default values are correctly either set to `{!$GlobalConstant.EmptyString}` or left blank (`null`).

**Example:** When you create a Boolean variable, the GLOBAL CONSTANT section includes `$GlobalConstant.True` and `$GlobalConstant.False`. However, when you create a Currency variable, the GLOBAL CONSTANT section isn’t available.

SEE ALSO:
- Flow Resources
Global Variables in Visual Workflow

Reference general information about your organization or the user who's running the flow interview. For example, use `$!User.Id` to easily access the ID of the user who's running the flow interview. In Visual Workflow, global variables are supported only in flow formula resources.

The following global variables are supported in flow formulas. If a value in the database has no value, the corresponding merge field returns a blank value. For example, if nobody has set a value for your organization’s Country field, `{!$Organization.Country}` returns no value.

<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Api</td>
<td>References API URLs or the session ID. The following merge fields are available.</td>
</tr>
<tr>
<td></td>
<td>• <code>Enterprise_Server_URL_xxx</code>—The Enterprise WSDL SOAP endpoint where <code>xxx</code> represents the version of the API.</td>
</tr>
<tr>
<td></td>
<td>• <code>Partner_Server_URL_xxx</code>—The Partner WSDL SOAP endpoint where <code>xxx</code> represents the version of the API.</td>
</tr>
<tr>
<td></td>
<td>• <code>Session_ID</code></td>
</tr>
<tr>
<td>$Label</td>
<td>References custom labels. This global variable appears only if custom labels have been created in your organization. The returned value depends on the language setting of the contextual user. The value returned is one of the following, in order of precedence:</td>
</tr>
<tr>
<td></td>
<td>1. The local translation’s text</td>
</tr>
<tr>
<td></td>
<td>2. The packaged translation’s text</td>
</tr>
<tr>
<td></td>
<td>3. The master label’s text</td>
</tr>
<tr>
<td>$Organization</td>
<td>References information about your company.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> <code>{!$Organization.UiSkin}</code> returns one of these values.</td>
</tr>
<tr>
<td></td>
<td>• <code>Theme1</code>—Obsolete Salesforce theme</td>
</tr>
<tr>
<td></td>
<td>• <code>Theme2</code>—Salesforce theme used before Spring ’10</td>
</tr>
<tr>
<td></td>
<td>• <code>PortalDefault</code>—Salesforce Customer Portal theme</td>
</tr>
<tr>
<td></td>
<td>• <code>Webstore</code>—Salesforce AppExchange theme</td>
</tr>
<tr>
<td></td>
<td>• <code>Theme3</code>—Current Salesforce theme, introduced during Spring ’10</td>
</tr>
</tbody>
</table>
| $Permission     | References information about the running user’s current access to any of your organization’s custom permissions. This global variable appears only if custom permissions have been created in your organization.
<table>
<thead>
<tr>
<th>Global Variable</th>
<th>Description</th>
</tr>
</thead>
</table>
| $Profile        | References information from the current user's profile, such as license type or name.  
Tip:  
- 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. |
| $Setup          | References custom settings of type “hierarchy”. This global variable appears only if hierarchy custom settings have been created in your organization. You can access custom settings of type “list” only in Apex.  
Hierarchical custom settings allow values at any of three different levels:  
- Organization—the default value for everyone  
- Profile—overrides the Organization value  
- User—overrides both Organization and Profile values  
Salesforce automatically determines the correct value for this custom setting field based on the running user’s current context. |
| $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 flow interview. For example, reference the user’s ID or title.  
Tip:  
- The current 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 current user is the Default Lead Owner or Default Case Owner.  
$User.UITheme and $User.UIThemeDisplayed identify the look and feel the running user sees on a given 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 may have the permissions to see the new user interface theme look and feel, but if they are using a browser that doesn’t support that look and feel, for example, Internet Explorer 6, $User.UIThemeDisplayed returns a different value. These merge fields return one of the following values:  
- Theme1—Obsolete Salesforce theme  
- Theme2—Salesforce theme used before Spring ‘10  
- PortalDefault—Salesforce Customer Portal theme  
- Webstore—Salesforce AppExchange theme  
- Theme3—Current Salesforce theme, introduced during Spring ‘10 |
| $UserRole       | References information about the current user’s role, such as the role name or ID.  
Tip:  
- The current user is the person who caused the flow to start. |
Global Variable | Description
---|---
• | When a flow is started because a Web-to-Case or Web-to-Lead process changed a record, the current user is the Default Lead Owner or Default Case Owner.

SObject Collection Variable Overview

An sObject collection variable is a container for an ordered group of related items. Similar to an sObject variable, each item in an sObject collection variable stores field values for one Salesforce record.

Once you’ve created an sObject collection variable, use one of these elements to interact with Salesforce.

• **Fast Create** element—Create new Salesforce records using all the populated fields of an sObject collection.

• **Fast Update** element—Update existing Salesforce records using all the populated fields of an sObject collection.

• **Fast Lookup** element—Populate an sObject collection variable by looking up existing Salesforce records and querying their fields.

• **Fast Delete** element—Delete Salesforce records by using the IDs that are stored in an sObject collection variable.

You can use a **Loop** element to examine every item in the collection. When an item is being examined in the loop, the item’s field values are copied into an sObject variable that you specify as the loop variable. If you want the loop to modify a collection item, such as to update an item’s field values:

1. Configure the elements within the loop to update the loop variable.

2. Add the variable’s field values to a separate collection.

You can add new items to the end of the collection (Assignment element) or replace all items in the collection (Fast Lookup element). However, you can’t update existing collection items. To get around this limitation, have the loop iteratively add the contents of the loop variable to another collection. When the loop finishes, you can update the Salesforce records with values from the new collection.

SEE ALSO:

- Sample Flow That Loops Through a Collection
- Add an SObject Collection to a Flow
- Add a Loop Element to a Flow
- Flow Resources
Add an SObject Collection to a Flow

Create an sObject collection variable to store field values for multiple Salesforce records. After the sObject collection is populated, you can use it to create, update, or delete records in the Salesforce database.

Before you can populate an sObject collection variable with field values, create the variable.

1. From Setup, click **Create > Workflow & Approvals > Flows** and open a new or existing flow.
2. From the Resources tab, double-click **SObject Collection Variable**.
3. Fill out the fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>The description appears in the Description pane when the collection is selected in the Explorer tab. If its Input/Output Type isn’t set to Private, you can see its description from another flow’s subflow element.</td>
</tr>
</tbody>
</table>
| Input/Output Type | Determines whether the sObject collection variable can be accessed outside the flow.  
• Private—Can be assigned and used only within the flow  
• Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs  
• Output—Can be accessed from Visualforce controllers and other flows  
The default value is Private. |
| ObjectType    | Type of Salesforce records that the sObject collection represents in the flow. |

4. Click **OK**
You can use an sObject variable’s field values to create an item at the end of a collection. To do so, use the `add` operator in an Assignment element.

SEE ALSO:
- Sample Flow That Loops Through a Collection
- SObject Collection Variable Overview
- Add a Loop Element to a Flow
- Flow Resources

Add an SObject Variable to a Flow

Use the SObject Variable resource to store and manipulate multiple field values for a Salesforce record in your flow.

Before you can populate an sObject variable with field values, create the variable.

1. From Setup, click **Create > Workflow & Approvals > Flows** and open a new or existing flow.
2. From the Resources tab, double-click **SObject Variable**.
3. Fill out the fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>The description appears in the Description pane when the variable is selected in the Explorer tab. If its Input/Output Type isn’t set to Private, you can see its description from another flow’s subflow element.</td>
</tr>
<tr>
<td>Input/Output Type</td>
<td>Determines whether the sObject variable can be accessed outside the flow.</td>
</tr>
<tr>
<td></td>
<td>• Private—Can be assigned and used only within the flow</td>
</tr>
<tr>
<td></td>
<td>• Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs</td>
</tr>
<tr>
<td></td>
<td>• Output—Can be accessed from Visualforce controllers and other flows</td>
</tr>
</tbody>
</table>

This field doesn’t affect how variables are assigned or used within the same flow, for example, through these types of elements: Assignment, Record or Fast Create, Record or Fast Lookup, and Apex Plug-in.

The default value of the field is Private.

**Warning:** 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, Visualforce controllers, subflows, and processes.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Type</td>
<td>Type of Salesforce record that the sObject variable represents in the flow.</td>
</tr>
</tbody>
</table>

4. Click **OK**

When an sObject variable is created, its default value is `null`. Before reference an sObject variable's values in another element, make sure that the sObject variable has a value by using the `is null` operator in a Decision element.

To determine whether a field value that's stored in an sObject variable was ever set, even if it's currently set to `null`, use the `was set` operator in a Decision element. If the entire sObject variable is set to `null` (rather than one of its field values), you can't use `was set`. You can't access any fields in an unpopulated sObject.

Once you've created an sObject variable, use one of these elements to interact with Salesforce.

- **Fast Create** element—Create a new Salesforce record by using all the populated fields of an sObject variable.
- **Fast Update** element—Update an existing Salesforce record by using all the populated fields of an sObject variable.
- **Fast Lookup** element—Populate an sObject variable by looking up an existing Salesforce record and querying its fields.
- **Fast Delete** element—Delete a Salesforce record by using the ID that's stored in an sObject variable.

**SEE ALSO:**

- Flow Resources

### System Variables in Flows

When you design a flow, reference values that refer to properties of a flow interview. For example, reference the time that the flow interview executes a certain element.

In any value or resource drop-down list in the Cloud Flow Designer, a section is available for system variables. Depending on the context of the drop-down list, you can expand SYSTEM AND GLOBAL VARIABLES to reference a system variable as the value for that field.

#### System Variable

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Supported Resource Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>!$Flow.CurrentDate</code></td>
<td>Text, Date, and DateTime</td>
<td>Date when the flow interview executes the element that references the system variable.</td>
</tr>
</tbody>
</table>
Example: A flow is used only 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 field on the screen 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}

SEE ALSO:
- Customize What Happens When a Flow Fails
- Flow Resources

Add a Text Template to a Flow

Use the Text Template resource to create text to use in your flow.

For 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.

1. From Setup, click Create > Workflow & Approvals > Flows
2. From the Resources tab, double-click Text Template.
3. Fill out these fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Helps you determine when to assign or use this formula as you set up your flows. The description appears in the Description pane when the formula is selected in the Explorer tab.</td>
</tr>
<tr>
<td>Text Template</td>
<td>The text for the template. Use to select resources from your flow. Use the CREATE NEW section of the drop-down list to create resources, if necessary.</td>
</tr>
</tbody>
</table>
4. Click OK.

SEE ALSO:
- Flow Resources
- Cross-Object Field References in Flows

Add a Variable to a Flow

Before you can assign values to flow variables with the Assignment element, create them. Variables are updatable values that you can reference throughout your flow.

1. From Setup, click Create > Workflow & Approvals > Flows
2. From the Resources tab, double-click Variable.
3. Fill out the fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.</td>
</tr>
<tr>
<td>Description</td>
<td>Helps you determine when to assign or use this variable as you set up your flows. The description appears in the Description pane when the variable is selected in the Explorer tab. If its Input/Output Type isn’t set to Private, you can see its description from another flow’s subflow element.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Determines the types of values that can be assigned to the variable.</td>
</tr>
<tr>
<td>Scale</td>
<td>Appears only when the Data Type is set to Number or Currency. The scale is the maximum number of digits allowed to the right of the decimal point. This number can’t exceed 17. If you leave this field blank or set to zero, only whole numbers are displayed when your flow runs.</td>
</tr>
<tr>
<td>Input/Output Type</td>
<td>Determines whether the variable can be accessed outside the flow.</td>
</tr>
<tr>
<td></td>
<td>• Private—Can be assigned and used only within the flow</td>
</tr>
<tr>
<td></td>
<td>• Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs</td>
</tr>
<tr>
<td></td>
<td>• Output—Can be accessed from Visualforce controllers and other flows</td>
</tr>
</tbody>
</table>
This field doesn’t affect how variables are assigned or used within the same flow, for example, through these types of elements: Assignment, Record or Fast Create, Record or Fast Lookup, and Apex Plug-in.

The default value of the field depends on the release or API version in which the variable is created:

- **Private** for a variable created in Summer ’12 and later or in API version 25.0 and later.
- **Input and Output** for a variable created in Spring ’12 and earlier or in API version 24.0.

⚠️ **Warning**: 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, Visualforce controllers, subflows, and processes.

### Default Value

Enter a default value for the variable, or use the drop-down list to assign the value of a flow resource as the default. Use the CREATE NEW section of the drop-down list to create resources, if necessary. If you leave this field blank, the variable value is set to `null`.

Default values aren’t available for Picklist and Picklist (Multi-Select) variables.

4. **Click OK**

You can delete a variable at any time. Any variable assignments that use the deleted variable are set to `null`.

SEE ALSO:

- Flow Resources
- Add an Assignment Element to a Flow
- Add an SObject Variable to a Flow
- Cross-Object Field References in Flows

### Flow Operators

Visual Workflow uses operators in three different contexts.

**IN THIS SECTION:**

- Operators in Flow Assignment Elements
  - Use Assignment element operators to change the value of a selected resource.
- Operators in Flow Conditions
  - Use condition operators to verify the value of a selected resource. Conditions are used in Decision elements and Wait elements.
- Operators in Flow Record Filters
  - A flow record filter narrows the scope of records that the flow operates on. For example, use a record filter to update only the contacts that are associated with the Acme Wireless account. When you add a Record Update element, use the record filters to narrow the scope to just the contacts whose parent account is Acme Wireless.
Operators in Flow Assignment Elements

Use Assignment element operators to change the value of a selected resource.

Use this reference, organized by the data type that you select for Resource, to understand the supported operators.

- Boolean
- Collection
- Currency
- Date
- Date/Time
- Multi-Select Picklist
- Number
- Picklist
- sObject
- Text

Boolean

Replace a boolean resource with a new value.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals</td>
<td>What you enter or select for Value replaces the value of Variable.</td>
<td>Boolean</td>
<td>Before Assignment: {!varBoolean} is false. Assignment: {!varBoolean} equals {!$GlobalConstant.True} After Assignment: {!varBoolean} is true.</td>
</tr>
</tbody>
</table>

Collection

Replace the value of a collection variable or sObject collection variable (equals) or add an item to the end of the variable (add).

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| equals   | What you enter or select for 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. |
### Operators in Flow Assignment Elements

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>add</strong></td>
<td>What you enter or select for Value is added as a new item at the end of the collection.</td>
<td>Variable of the same data type or sObject variable of the same object type Text, Picklist, and Multi-Select Picklist data types are compatible with each other.</td>
<td>Before the Assignment: <code> {!collText}</code> is Yellow, Green, Blue <code> {!varPicklist}</code> is Red Assignment: <code> {!collText} add {!varPicklist}</code>. After the Assignment: <code> {!collText}</code> is Yellow, Green, Blue, Red.</td>
</tr>
</tbody>
</table>

### Currency and Number

Replace (equals), add to (add), or subtract from (subtract) the value of a currency or number resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>equals</strong></td>
<td>The number that you enter or select for Value replaces the value of Variable.</td>
<td>Currency, Number</td>
<td>Before the Assignment: <code> {!varCurrency}</code> is 10. Assignment: <code> {!varCurrency} equals 7</code>. After the Assignment: <code> {!varCurrency}</code> is 7.</td>
</tr>
<tr>
<td><strong>add</strong></td>
<td>The number that you enter or select for Value is added to the value of Variable.</td>
<td>Currency, Number</td>
<td>Before the Assignment: <code> {!varCurrency}</code> is 10. Assignment: <code> {!varCurrency} add 7</code>. After the Assignment: <code> {!varCurrency}</code> is 17.</td>
</tr>
<tr>
<td><strong>subtract</strong></td>
<td>The number that you enter or select for Value is subtracted from the value of Variable.</td>
<td>Currency, Number</td>
<td>Before the Assignment: <code> {!varCurrency}</code> is 10. Assignment: <code> {!varCurrency} subtract 7</code>. After the Assignment: <code> {!varCurrency}</code> is 3.</td>
</tr>
</tbody>
</table>

### Date

Replace (equals), add to (add), or subtract from (subtract) the value of a date/time resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>equals</strong></td>
<td>The date that you enter or select for Value replaces the value of Variable.</td>
<td>Date, Date/Time</td>
<td>Before the Assignment: <code> {!varDate}</code> is 1/16/2016. Assignment: <code> {!varDate} equals 1/15/2016</code>. After the Assignment: <code> {!varDate}</code> is 1/15/2016.</td>
</tr>
</tbody>
</table>
# Operators in Flow Assignment Elements

## Example Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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).

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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. |
<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| **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).

콘accordion Note: Before values are assigned or added to a multi-select picklist resource, they’re converted into string values.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| **equal** | 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. |
### Operators in Flow Assignment Elements

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>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</td>
<td></td>
<td></td>
<td>Before the Assignment: <code>{!varMSP}</code> is Blue; Green. Assignment: <code>{!varMSP} add item Yellow</code>. After the Assignment: <code>{!varMSP}</code> is Blue; Green; Yellow. This value includes three separate selections.</td>
</tr>
<tr>
<td>add item</td>
<td>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.</td>
<td>• Boolean • Currency • Date • Date/Time • Multi-Select Picklist • Number • Picklist • Text</td>
<td></td>
</tr>
<tr>
<td>sObject</td>
<td>Replace an sObject variable with a new value (equals).</td>
<td>sObject with the same object type</td>
<td>Before the Assignment:</td>
</tr>
<tr>
<td>equals</td>
<td>The sObject that you select for Value replaces the value of Variable.</td>
<td>• <code>{!account1}</code> contains field values for the Acme Wireless account • <code>{!account2}</code> contains field values for the Global Media account</td>
<td>Assignment: <code>{!account1} equals {!account2}</code>. After the Assignment: both <code>{!account1}</code> and <code>{!account2}</code> contain the field values for the Global Media account.</td>
</tr>
<tr>
<td>Text</td>
<td>Replace a text resource with a new value (equals) or concatenate a value onto the end of the original value (add).</td>
<td></td>
<td>Note: Before values are assigned or added to a text resource, they’re converted into string values.</td>
</tr>
</tbody>
</table>
### Operators in Flow Conditions

Use condition operators to verify the value of a selected resource. Conditions are used in Decision elements and Wait elements.

Use this reference, divided up by the data type that you select for Resource, to understand the supported operators.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
</table>
| 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  
- Text | Before the Assignment: `{!varText}` is Blue.  
Assignment: `{!varText}.equals Yellow.`  
After the Assignment: `{!varText}` is Yellow. |
| 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  
- Text | Before the Assignment: `{!varText}` is Blue.  
Assignment: `{!varText}.add Yellow.`  
After the Assignment: `{!varText}` is BlueYellow. |

**EDITIONS**

Available in:  
- Enterprise  
- Performance  
- Unlimited  
- Developer
Boolean

Check whether a Boolean resource’s value matches another value or resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The value of the selected Resource doesn't match what you enter or select for Value.</td>
<td>Boolean</td>
</tr>
<tr>
<td>equals</td>
<td>The value of the selected Resource matches what you enter or select for Value. An outcome resolves to true if the flow interview took that outcome. A wait event resolves to true if all of the waiting conditions for that event are met.</td>
<td>Boolean</td>
</tr>
<tr>
<td>was set</td>
<td>The value for Resource is a field in an sObject variable, and that field has been populated with a value in the flow at least once.</td>
<td>Boolean</td>
</tr>
<tr>
<td>was visited</td>
<td>The selected Resource is an element in the flow, and it has been visited during the flow interview.</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>was selected</td>
<td>A user selected that choice or dynamic choice in a screen choice input field. If your flow references the same choice option in multiple screens, was selected always evaluates to the most recent screen that the flow visited. If your flow references the same choice option with a user input in more than one place on the same screen, this operator always evaluates the first usage in the screen.</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

Collection

Check whether a Collection resource’s value contains or matches another value or resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains</td>
<td>An item in the collection that’s selected for Resource contains the exact same value as Value.</td>
<td>Varies</td>
</tr>
</tbody>
</table>

If the resource is an sObject collection variable, only sObject resources with the same object type are supported.
### Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The collection that’s selected for Resource doesn’t match the collection that’s selected for Value. Two sObject collection variables are unequal if they include different fields or if the fields have different values.</td>
<td>Otherwise, only resources with the same data type are supported.</td>
</tr>
<tr>
<td>equals</td>
<td>The collection that’s selected for Resource matches the collection that’s selected for Value. Two sObject collection variables are equal if they include the same fields and those fields have the same values.</td>
<td>Collection of the same object type are supported.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Otherwise, only collection variables with the same data type are supported.</td>
</tr>
<tr>
<td>is null</td>
<td>The collection that’s selected for resource isn’t populated with any values</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### Currency and Number

Check whether a Currency or Number resource’s value matches, is larger than, or is smaller than another value or resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
</tbody>
</table>
### Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>greater than</td>
<td>The value of the Resource is larger than what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>greater than or equal</td>
<td>The value of the Resource is larger than what’s entered or selected for Value or is the same</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>less than</td>
<td>The value of the Resource is smaller than what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>less than or equal</td>
<td>The value of the Resource is smaller than what’s entered or selected for Value or is the same</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>is null</td>
<td>Resource isn’t populated with a value</td>
<td>Boolean</td>
</tr>
<tr>
<td>was set</td>
<td>The value for Resource is a field in an sObject variable, and that field has been populated with a value in the flow at least once</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### Date and Date/Time

Check whether a Date or Date/Time resource’s value matches, is before, or is after another value or resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>Date, Date/Time</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>Date, Date/Time</td>
</tr>
<tr>
<td>greater than</td>
<td>The value of the Resource is a later date or time than what’s entered or selected for Value</td>
<td>Date, Date/Time</td>
</tr>
<tr>
<td>greater than or equal</td>
<td>The value of the Resource is a later date or time than what’s entered or selected for Value or is the same date or time</td>
<td>Date, Date/Time</td>
</tr>
<tr>
<td>less than</td>
<td>The value of the Resource is an earlier date or time than what’s entered or selected for Value</td>
<td>Date, Date/Time</td>
</tr>
<tr>
<td>less than or equal</td>
<td>The value of the Resource is an earlier date or time than what’s entered or selected for Value or is the same date or time</td>
<td>Date, Date/Time</td>
</tr>
</tbody>
</table>
Create a Flow

Operators in Flow Conditions

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>is null</td>
<td>Resource isn’t populated with a value</td>
<td>Boolean</td>
</tr>
<tr>
<td>was set</td>
<td>The value for Resource is a field in an sObject variable, and that field has been populated with a value in the flow at least once</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains</td>
<td>The value for Resource contains what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td>For example, if the value of {!varPicklist} is yellow-green, the condition {!varPicklist} contains green evaluates to true.</td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
</tbody>
</table>
### Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
</tr>
</thead>
<tbody>
<tr>
<td>was set</td>
<td>The value for Resource is a field in an sObject variable, and that field has been populated with a value in the flow at least once</td>
</tr>
<tr>
<td>contains</td>
<td>The value for Resource contains what’s entered or selected for Value</td>
</tr>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource exactly matches what’s entered or selected for Value</td>
</tr>
</tbody>
</table>

#### 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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains</td>
<td>The value for Resource contains what’s entered or selected for Value</td>
</tr>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource exactly matches what’s entered or selected for Value</td>
</tr>
</tbody>
</table>

**Tip:** 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 organization 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.

**Note:** Order matters. If you aren’t sure which order the values that you’re checking for will appear in, 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.

**Note:** Order matters. If you aren’t sure which order the values that you’re checking for will appear in, 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 will resolve to false.
### Operators in Flow Conditions

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>was set</td>
<td>The value for Resource is a field in an sObject variable, and that field has been populated with a value in the flow at least once</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multi-select</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Text</td>
</tr>
</tbody>
</table>

### sObject

Check whether an sObject resource’s value matches another value or resource.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>sObject with the same object type</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>sObject with the same object type</td>
</tr>
<tr>
<td>is null</td>
<td>Resource isn’t populated with a value</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### 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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains</td>
<td>The value for Resource contains what’s entered or selected for Value</td>
<td>Boolean, Currency, Date, Date/Time, Multi-select, Picklist, Number, Picklist, Text</td>
</tr>
<tr>
<td>Operator</td>
<td>True if...</td>
<td>Supported Data Types</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>• Number</td>
</tr>
<tr>
<td>ends with</td>
<td>The end of the value for Resource matches what’s entered or selected for Value</td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>is null</td>
<td>Resource isn’t populated with a value</td>
<td>Boolean</td>
</tr>
<tr>
<td>starts with</td>
<td>The beginning of the value for Resource matches what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
</tbody>
</table>
Operators in Flow Record Filters

A flow record filter narrows the scope of records that the flow operates on. For example, use a record filter to update only the contacts that are associated with the Acme Wireless account. When you add a Record Update element, use the record filters to narrow the scope to just the contacts whose parent account is Acme Wireless.

Use this reference, organized by the data type of the field that you select, to understand the supported operators.

- Address Fields
- Autonumber Fields
- Checkbox Fields
- Currency Fields
- Date Fields
- Date/Time Fields
- Email Fields
- Encrypted Text Fields
- External Lookup Relationship Fields
- Fax Fields
- Lookup Relationship Fields
- Multi-Select Picklist Fields
- Number Fields
- Parent Fields
- Percent Fields
- Phone Fields
- Picklist Fields
- Text Fields
- Text Area (Long) Fields
- Text Area (Rich) Fields
- URL Fields

Checkbox Fields

When you select a checkbox field under Field, these operators are available.
### Operator

<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value ...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>Doesn’t match what you enter or select for Value</td>
<td>Boolean</td>
</tr>
<tr>
<td>equals</td>
<td>Matches what you enter or select for Value</td>
<td>Boolean</td>
</tr>
<tr>
<td>is null</td>
<td>Hasn’t been populated with a value yet, if you select True for Value</td>
<td>Boolean</td>
</tr>
</tbody>
</table>

### Currency, Number, and Percent Fields

When you select a currency, number, or percent field under Field, these operators are available.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value ...</th>
<th>Supported Data Types</th>
</tr>
</thead>
</table>
| 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              |
| 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                                              |                      |

### Date and Date/Time

When you select a date or date/time field under Field, these operators are available.
<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value ...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>The value for Resource doesn’t match what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td>greater than</td>
<td>The value of the Resource is a later date or time than what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td>greater than or equal</td>
<td>The value of the Resource is a later date or time than what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td>or is the same date or time</td>
<td>• Date/Time</td>
</tr>
<tr>
<td>is null</td>
<td>Hasn’t been populated with a value yet, if you select True for Value</td>
<td>Boolean</td>
</tr>
<tr>
<td>less than</td>
<td>The value of the Resource is an earlier date or time than what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td>less than or equal</td>
<td>The value of the Resource is an earlier date or time than what’s entered or selected for Value</td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td>or is the same date or time</td>
<td>• Date/Time</td>
</tr>
</tbody>
</table>

**Picklist and Text Fields**

When you select a picklist or text field under Field, these operators are available.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value ...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>contains</td>
<td>Contains what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>does not equal</td>
<td>Doesn’t match what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
</tbody>
</table>

111
<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>equals</td>
<td>Matches what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>ends with</td>
<td>Ends with what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>is null</td>
<td>Hasn’t been populated with a value yet, if you select True for Value</td>
<td>Boolean</td>
</tr>
<tr>
<td>starts with</td>
<td>Begins with what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
</tbody>
</table>

**Multi-Select Picklist Fields**

When you select a multi-select picklist field under Field, these operators are available.
In the Cloud Flow Designer, contains appears in the operator list when you select a multi-select picklist field. Don’t use this operator. Flows that use this operator in a record filter for a multi-select picklist field always fail.

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.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Filters to records where the selected field’s value ...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>does not equal</td>
<td>Doesn’t match what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>equals</td>
<td>Matches what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>ends with</td>
<td>Ends with what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
<tr>
<td>is null</td>
<td>Hasn’t been populated with a value yet, if you select True for Value</td>
<td>Boolean</td>
</tr>
<tr>
<td>Operator</td>
<td>Filters to records where the selected field’s value …</td>
<td>Supported Data Types</td>
</tr>
<tr>
<td>---------------</td>
<td>-------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>starts with</td>
<td>Begins with what’s entered or selected for Value</td>
<td>• Boolean</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Currency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Date/Time</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Multi-select Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Number</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Picklist</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Text</td>
</tr>
</tbody>
</table>

**Flow Connectors**

Connectors determine the available paths that a flow can take at run time.

In the Cloud Flow Designer canvas, a connector looks like an arrow that points from one element to another.

<table>
<thead>
<tr>
<th>Label</th>
<th>Example</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlabeled</td>
<td>![Unlabeled Arrow]</td>
<td>Identifies which element to execute next.</td>
</tr>
<tr>
<td>Decision outcome name</td>
<td>![Decision X &gt; Y Arrow]</td>
<td>Identifies which element to execute when the criteria of a Decision outcome are met.</td>
</tr>
<tr>
<td>Wait event name</td>
<td>![Wait 1 day after X Arrow]</td>
<td>Identifies which element to execute when an event that’s defined in a Wait element occurs.</td>
</tr>
<tr>
<td>FAULT</td>
<td>![FAULT Arrow]</td>
<td>Identifies which element to execute when the previous element results in an error.</td>
</tr>
<tr>
<td>Next element</td>
<td>![Next element Arrow]</td>
<td>Identifies the first element to execute for each iteration of a Loop element.</td>
</tr>
<tr>
<td>End of loop</td>
<td>![End of loop Arrow]</td>
<td>Identifies which element to execute after a Loop element finishes iterating through a collection.</td>
</tr>
</tbody>
</table>

**SEE ALSO:**
- Flow Elements
Add Connectors to Define Flow Paths

To identify which flow elements to execute and in what order, add connectors between the elements on your 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 outcome to assign to the path.

SEE ALSO:
- Delete Connectors
- Flow Connectors
- Customize What Happens When a Flow Fails

Delete Connectors

You can't modify a connector's target or source elements, so to change a path, delete the connector and then add a new one.

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.

1. In your flow, select the connector to delete.
   When you select a connector, its color changes from gray to green. You can also select a connector by clicking and dragging an area on the canvas that includes both ends of the connector.
2. Press DELETE.

SEE ALSO:
- Add Connectors to Define Flow Paths
- Flow Connectors

Customize What Happens When a Flow Fails

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

IN THIS SECTION:
- 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 by default.
Configure Every Fault Path to Send You an Email (Best Practice)
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 of 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”. Do this only if the distribution method you’re using supports flows that contain screens. In other words, don’t do this 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
Add Connectors to Define Flow Paths

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 by default.

Here’s what happens by default.

• This error message displays to the running user—the user who was running the flow.

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 that can appear in a fault email.

An error occurred at element Fast_Delete_1.
DELETE --- There is nothing in Salesforce matching your delete criteria.

SEE ALSO:
Customize What Happens When a Flow Fails
Configure Every Fault Path to Send You an Email (Best Practice)

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 of your flow’s resources. The resource values can give you insight into why the flow failed.

1. Create a text template that includes the values of all the flow resources.

   Doing so lets you see the exact values of flow variables when the interview failed. Also, if the flow contains screens, you see exactly what the user entered and selected.

   Here’s an example text template for the Customer Satisfaction Survey flow in the Cloud Flow Designer Workbook.

   ```
   Error: {!$Flow.FaultMessage}
   
   RESOURCE VALUES
   Customer Response: {!Customer_Response}
   Value of Decision's Yes outcome: {!Yes}
   Company: {!Company_Name}
   Satisfaction Choice Field: {!Satisfaction}
   Service Choice Field: {!Service}
   Other Comments:
   {!OtherComments}
   ```

2. Configure a Send Email element. Use the text template as the body and your email address as the recipient.

   In this example, Body is set to the text template we created: {!allVariableValues}.

3. From each element that can fail, draw a fault connector to the Send Email element.

   In this example, Record Create is the only element that supports fault connectors.
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”. Do this only if the distribution method you’re using supports flows that contain screens. In other words, don’t do this if your flow is distributed through a process.

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

2. Add a Screen element. In a Display Text field, reference the text template.

3. For every element that can fail, draw a fault connector to the Screen element.
In this example, Record Create is the only element that supports fault connectors. After the flow displays the better error message to the user, it sends an email to the admin with debugging information.

SEE ALSO:
- Customize Display Text on a Flow Screen
- Add a Text Template to a Flow
- 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.

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 ID to a Text variable (e.g., `!caseId`), 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 a case 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

Cross-Object Field References in Flows

When building a flow, you can reference fields for records that are related to the values that are stored in an sObject variable. To do so, manually enter the references. Cross-object field values are valid wherever you can reference a flow resource or manually enter a value.

EDITIONS

Available in:
- Enterprise
- Performance
- Unlimited
- Developer
Implementation Tips

If a flow interview encounters a null value at any point in the cross-object expression, the element containing the reference fails. The reference runs successfully if the last field value in the expression is null. For example, store a contact in {!sObjContact} and try to reference {!sObjContact}.Account.Name. The flow fails if AccountId on the stored contact is null (because there isn’t an account to look at), but it succeeds if Name on the related account is null.

If an element contains a cross-object reference that fails and the element doesn’t have a fault path defined, the entire interview fails. To avoid this situation, you can:

- Make the fields that you want to reference in the expression required in Salesforce. For example, for the expression {!sObjContact}.Account.Name, you could require AccountId on contact page layouts. Then use another flow to find any records with null values for that field and update them.
- Determine whether each field that’s referenced in the expression has a value by using the wasSet operator in a Decision element.

Cross-object field references in flows don’t count against your organization’s limits for:

- Cross-object relationships per object
- DML operations per transaction

Syntax

To reference a field on a related record, use this syntax.

```
{sObjectVariable.objectName1.objectName2.fieldName}
```

where:

- **sObjectVariable** is the unique name for the sObject variable that you want to start from.
- **objectName1** is the API name for an object that’s related to sObjectVariable’s object type. The API names for all custom objects end in __r.
- (Optional) **objectName2** is the API name for an object that’s related to **objectName1**.
  
  Your expression must include at least one object name, but you can add more objects as needed.

- **fieldName** is the name for the field that you want to reference on the last object in the expression. The API names for all custom fields end in __c.

For example, {!sOv_Contract.Account.Id} references Id of the account that’s related to the contact record represented by an sObject variable in the flow.

When you create an sObject variable to reference fields on related records from, store the ID for the first related record in the variable. For example, to reference an opportunity’s contract, store ContractId in the sObject variable or add a value for ContractId by using an Assignment element.

Example: For example, to update a contract’s owner to be the contract’s account’s owner:
1. Use a Fast Lookup element to store the contract’s fields, including AccountId, in an sObject variable called varContract.

2. Use a Decision element to verify that the value of AccountId was set in varContract.

3. Use a Fast Lookup to store the fields for the contract’s account, including OwnerId, in another sObject variable called varAccount.

4. Use a Decision element to confirm that the value of OwnerId was set in varAccount.

5. Use an Assignment element to specify {!varContract.Account.OwnerId} as the value for {!varContract.OwnerId}.

6. Use a Fast Update element to write the values in varContract, including the updated OwnerId value, to the contract in Salesforce.

**Set a Flow’s Start Element**

Before you can save a flow, indicate which element to execute first.

1. Hover over the starting element in your flow.

2. Click ⚫.

SEE ALSO:

Save a Flow
Save a Flow

After you create a flow in the Cloud Flow Designer, you have some options for saving the flow.

Initial save
When you save a new flow for the first time, a dialog box appears. Enter values for each of the flow's properties. Once you save the flow, the unique name can't be changed.

Quick save
After you've saved a flow once, the Save button works as a quick-save, overwriting your previous work. However, the Save button doesn't work when editing active flows. To save your changes as a new version or new flow, use Save As.

Save As
After you've saved your flow once, this button is enabled with two options:

- **Save as new flow** opens a dialog box where you can input a new name, unique name, and description, then save your changes as an entirely new flow.
- **Save as new version** saves the flow as a new version of the current flow. Use this option if you want to change a flow and keep the old configuration as a backup to refer to it later.

Each flow can have up to 50 versions. You can't update the unique name when you save a new version.

When saving a flow or flow version:

- If you have the flow detail page open in one browser tab, then edit a version in another tab, before you run the edited version:
  1. Save the version.
  2. Close the Cloud Flow Designer.
  3. Refresh the flow detail page in the first tab.

- If you've changed the flow properties and for some reason the flow fails to save, the flow properties don't revert to the previous values.

SEE ALSO:

- Cloud Flow Designer Overview
- Activate a Version of a Flow

EDITIONS

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

USER PERMISSIONS

To open, edit, or create a flow in the Cloud Flow Designer:
- "Manage Force.com Flow"
CHAPTER 4 Manage Your Flows

To manage a flow, from Setup, click Create > Workflow & Approvals > Flows and click a flow name.

On the flow detail page you can:

- View a list of flow versions.
- See the flow namespace prefix, if it was installed from a managed package. The Cloud Flow Designer can’t open flows that are installed from managed packages.
- See the status of each version. Inactive flows are valid and can be activated. Draft flows are inactive, invalid, and can’t be activated.
- Edit the name or description of the flow.
- Open a flow.
- Test a flow.
- Activate or deactivate a version of the flow.
- Delete a flow interview.
- Delete a flow or flow version.

SEE ALSO:
  - Visual Workflow Overview
  - General Flow Limits
  - Flow Administration Considerations

Flow Properties

A flow's properties consist of its name, description, interview label, and type. The properties of a flow and its flow versions are separate.

Tip: The properties for a given flow's versions automatically match the active version's properties by default. In other words, if you have three versions and you activate version 2, Salesforce updates the properties for versions 1 and 3 to match version 2. However, if you edit the properties for an inactive version, that version's properties are no longer automatically updated to match the active version.

From the Cloud Flow Designer, click ![Link Icon] to update the properties for a flow or a flow version.
### Property | Description
--- | ---
Name | The name for the flow or flow version. The name appears in the flow management page and flow detail page. It also appears in the run time user interface. You can edit the name for inactive flows and flow versions.

Unique Name | The unique name for the flow. The unique name is used to refer to this flow from other parts of Salesforce, such as in a URL or Visualforce page. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. The unique name appears on the flow detail page. You can’t edit the unique name after the flow has been saved.

Description | The description for the flow or flow version. The description appears in the flow management page and flow detail page. You can edit the description for inactive flows and flow versions.

Type | The type for the flow or flow version. The type appears in the flow management page and flow detail page. It determines which elements and resources are supported in the flow or flow version, as well as the ways that the flow can be implemented. For details, see Flow Types on page 124. If the type is Login Flow, you can’t update the type after the flow has been saved.

Interview Label | The label for the flow’s interviews. An interview is a running instance of a flow. This label appears in:
- The Paused and Waiting Interviews list on the flow management page
- The Paused Interviews component on the Home tab
- The Paused Interviews item in Salesforce1
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\} system variable.

Use a text template to reference multiple resources in the label. For example, **Flow Name** - \{!Account.Name\} - \{!$Flow.CurrentDateTime\}.

**SEE ALSO:**
- Visual Workflow Overview
- Cloud Flow Designer Overview
- Manage Your Flows

**Flow Types**

A flow or flow version’s type determines which elements and resources are supported, as well as the ways that the flow can be distributed.

**Standard Flow Types**

The following flow types are supported in the Cloud Flow Designer.

**EDITIONS**

Available in:
- Enterprise
- Performance
- Unlimited
- Developer

124
### Other Flow Types

Not all flow types are supported in the Cloud Flow Designer. Some flow types are used in other parts of Salesforce. You can’t create or edit these flows in the Cloud Flow Designer, so you don’t see them in the list of flows. However, the Paused and Waiting Interviews list on the flow management page can display interviews with one of these types.

For example, when you run a process (from the Process Builder), a flow interview is created. You can monitor that interview in the Paused and Waiting Interviews list by looking for the type “Workflow”.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workflow</td>
<td>A running instance of a process created in the Process Builder.</td>
</tr>
</tbody>
</table>

SEE ALSO:

Flow Properties
Open and Modify a Flow

To modify a flow, open it in the Cloud Flow Designer.

You can’t save changes to an active flow version. You can, however, open an active version of a flow, modify it, and then save as a new version or a new flow.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. Click the name of the flow.
3. Open the flow.
   - To open a specific version, click the Open link next to that version number.
   - To open the active version of the flow, click the Open button. If there isn’t an active version, the latest version opens.

SEE ALSO:
- Flow Design Considerations
- Manage Your Flows
- Activate a Version of a Flow

Test a Flow

Test your flows before you activate them to make sure they’re working as expected.

⚠️ Warning: Be careful when testing flows that contain delete elements. Even if the flow is inactive, it triggers the delete operation.

We recommend that you test all possible paths through the flow, so that you can find and fix any errors before activating the flow. For example, incomplete data in the flow can cause a data element (create, update, lookup, or delete) to fail at run time. Add a fault connector to a path that corrects the data and allows the flow to successfully finish.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. Click the name of the flow you want to run.
3. Run the flow.
   - To run a specific version, click the Run link for that version.
   - To run the active version of the flow, click the Run button. If there isn’t an active version, the latest version runs.
   - To run a flow version from the Cloud Flow Designer, open that version and then click Run from the button bar.

⚠️ Tip: If you recently modified the flow that you’re testing, save it. Only the most recently saved changes are included when you run a flow.
Once you’re confident that your flow is working as expected, activate the version that you tested and then distribute the flow.

SEE ALSO:
- Activate a Version of a Flow
- Customize What Happens When a Flow Fails
- Flow Run Time Considerations
- Manage Your Flows

Activate a Version of a Flow

You can have several different versions of a single flow in Salesforce, but only one version of each flow can be active at a time. To activate or deactivate a version of a flow, go to that flow’s detail page in Setup.

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.

1. From Setup, click Create > Workflow & Approvals > Flows.
2. Click the name of the flow.
3. Click Activate or Deactivate next to the relevant version of the flow.

SEE ALSO:
- Flow Administration Considerations
- Manage Your Flows

Delete a Paused or Waiting Flow Interview

If you no longer need to wait for a long-running flow interview to finish or for a user to resume a paused interview, delete the interview. For example, when you’re updating or deleting the associated flow version.

1. From Setup, click Create > Workflow & Approvals > Flows.
   If there are waiting interviews for any of your flows, the Waiting Interviews related list appears underneath the list of flows.

2. For each interview that you want to delete, click Del.
Delete a Flow

To delete an active flow version, first deactivate it. If a flow has any paused or waiting interviews, it can’t be deleted until those interviews are finished or deleted. Flows that have never been activated can be deleted immediately.

1. From Setup, click "Create > Workflow & Approvals > Flows."
2. Click the name of the flow.
3. To delete the flow completely, including all versions, click the Delete button.
4. To delete an individual version, click the Del link for that version.

SEE ALSO:
- Flow Administration Considerations
- Manage Your Flows

Let Users Pause Flows

Customize your organization’s workflow and approval settings to enable your users to pause a flow interview that they can’t finish yet. A flow interview is a running instance of a flow. For example, a customer service representative can pause a flow interview when the customer doesn’t have all of the necessary information.

1. From Setup, click "Create > Workflow & Approvals > Settings."
2. Select Let Users Pause Flows.
3. Click Save.

Screens don’t automatically display the Pause button once Let Users Pause Flows is enabled. If you want your users to be able to pause at a given screen, select “Show Pause button” when you configure that screen.

SEE ALSO:
- Configure a Flow Screen’s General Settings
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.

IN THIS SECTION:

- **Distribute a Flow to Internal Users**
  Enable your internal users to run your flow through the flow URL or a Visualforce page.

- **Distribute a Flow to External Users**
  Enable external users to run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a Force.com site, Customer Portal, or Partner Portal.

- **Launch a Flow Automatically**
  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.

- **Deploy a Flow to Other Organizations**
  Flows created in the Cloud Flow Designer can be included in change sets and packages. The recipient organization of either the change set or package must have Visual Workflow enabled.

### Distribute a Flow to Internal Users

Enable your internal users to run your flow through the flow URL or a Visualforce page.

IN THIS SECTION:

- **Distribute a Flow URL**
  Users in your organization 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 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.
Distribute a Flow URL

Users in your organization 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, click Create > Workflow & Approvals > Flows.
2. Click the name of the flow.
3. Verify that there’s an active version.
   Only users with the "Manage Force.com 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:

   http://na1.salesforce.com/flow/MyFlowName

   If the flow was installed from a managed package, include the namespace prefix in the flow URL. For example:

   http://na1.salesforce.com/flow/namespace/MyFlowName

5. To set the initial values of your flow’s variables, append ?variable1=value1&variable2=value2 to the URL. See Set Flow Variables from a Flow URL for details.
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.

IN THIS SECTION:

Set Flow Variables from a Flow URL
When you distribute a flow using a URL, you can set the initial values of flow variables and collection variables by using parameters in the URL.

Set Flow Finish Behavior with a Flow URL
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 users to be redirected to another page within Salesforce when they click Finish, use the retURL parameter in the flow URL.
Set Flow Variables from a Flow URL

When you distribute a flow using a URL, you can set the initial values of flow variables and collection variables by using parameters in the URL.

Implementation Tips

- You can’t set the values for sObject variables or sObject collection variables using URL parameters. The variable that you want to set must have its Input/Output Type set to allow input access.
- Variable names are case-sensitive. For example, you can’t set the variable varNumber by entering VarNumber as a URL parameter.
- 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 variables of type Currency can accept only numeric values, so the flow fails at runtime. Instead, pass the record’s ID to a flow Text variable with a URL parameter. Then in the flow, use the ID to look up that record’s value for the currency field.

Format

To set the initial value of a flow variable:

http://instance.salesforce.com/flow/flowName?variableName=value

To set the initial value of a flow variable when launching a specific version of a flow:

http://instance.salesforce.com/flow/flowName/flowVersionNumber?variableName=value

Note: Only users with the “Manage Force.com Flow” permission can run inactive flows.

To set the initial values of multiple flow variables:

http://instance.salesforce.com/flow/flowName?variable1Name=value1&variable2Name=value2

To set the initial values for items in a collection variable:

http://instance.salesforce.com/flow/flowName?collection=value1&collection=value2

Valid Values

Use this table to determine which values are valid for a given flow variable.

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Acceptable Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Merge field of type Date or YYYY-MM-DD</td>
</tr>
<tr>
<td>DateTime</td>
<td>Merge field of type Date/Time or YYYY-MM-DDThh:mm:ssZ</td>
</tr>
<tr>
<td>Text</td>
<td>Merge field of any type or a string</td>
</tr>
<tr>
<td>Number</td>
<td>Merge field of type Number or a numeric value</td>
</tr>
<tr>
<td>Currency</td>
<td>Merge field of type Number or a numeric value</td>
</tr>
</tbody>
</table>
### Variable Type

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Acceptable Values</th>
</tr>
</thead>
</table>
| Boolean       | • Merge field of type Checkbox  
               | • True values: true or 1  
               | • False values: false or 0 |

### Examples

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.

```
http://na1.salesforce.com/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.

```
http://na1.salesforce.com/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.

```
http://na1.salesforce.com/flow/Contact_Info?collNames={!Contact.FirstName}&collNames={!Contact.LastName}
```

### Set Flow Finish Behavior with a Flow URL

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 users to be redirected to another page within Salesforce when they click Finish, use the retURL parameter in the flow URL.

#### Format

To redirect users to a specific page in Salesforce after they click Finish, use the `retURL` parameter in the flow URL.

```
http://instance.salesforce.com/flow/flowName?retURL=page_name
```

where `page_name` is the part of the URL that comes after `http://instance.salesforce.com/`.

#### Limitations

- You can't redirect flow users to a URL that's external to your Salesforce organization.
- 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.
Examples

The following flow URL redirects the user to the home tab for cases (http://na1.salesforce.com/500/o).

http://na1.salesforce.com/flow/Case_Management?retURL=500/o

The following 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 http://na1.salesforce.com/home/home.jsp.

http://na1.salesforce.com/flow/User_Info?varUserFirst={!$User.FirstName} &varUserLast={!$User.LastName} &retURL=home/home.jsp

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.

1. Find the flow’s unique name.
   a. From Setup, click Create > Workflow & Approvals > Flows.
   b. Click the name of the flow.
   c. Copy the unique name of the flow.

2. From Setup, click Develop > 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 unique name of the flow.
   For example:

   <apex:page>
   <flow:interview name="flowuniquename"/>
   </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 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 Force.com app by using a custom button, link, or Visualforce tab.
IN THIS SECTION:

Set Flow Variable Values from a Visualforce Page
After you embed your flow in a Visualforce page, you can set the initial values of variables, sObject variables, collection variables, and sObject collection variables through the `<apex:param>` component.

Get Flow Variable Values to a Visualforce Page
Flow variable values can be displayed in a Visualforce page. Once you’ve embedded your flow in a Visualforce page, you can use Visualforce markup to get values for variables or sObject variables. To display values for a collection variable or an sObject collection variable, you can use Visualforce markup to get the individual values contained in the collection.

Configure 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.

SEE ALSO:
Set Flow Variable Values from a Visualforce Page
Get Flow Variable Values to a Visualforce Page
Configure the Flow’s Finish Behavior

Set Flow Variable Values from a Visualforce Page
After you embed your flow in a Visualforce page, you can set the initial values of variables, sObject variables, collection variables, and sObject collection variables through the `<apex:param>` component.

Note: You can set variables only at the beginning of an interview. The `<apex:param>` tags are evaluated only once, when the flow is launched.

You can set only variables that allow input access. For each flow variable, input access is controlled by:

- The Input/Output Type variable field in the Cloud Flow Designer
- The isInput field on FlowVariable in the Metadata API

For a variable that doesn’t allow input access, attempts to set the variable are ignored, and compilation may fail for the Visualforce page, its `<apex:page>` component, or the Apex class.

The following table lists the ways you can set a flow’s variable, sObject variable, and sObject collection variable values using Visualforce.

<table>
<thead>
<tr>
<th>Method</th>
<th>Variables</th>
<th>sObject Variables</th>
<th>Collection Variables</th>
<th>sObject Collection Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without a controller</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>With a standard controller</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>With a standard List controller</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>With a custom Apex controller</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>With an Interview Map</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

134
Setting Variable Values without a Controller

This example sets myVariable to the value 01010101 when the interview starts.

```apex
<apex:page>
    <flow:interview name="flowname">
        <apex:param name="myVariable" value="01010101"/>
    </flow:interview>
</apex:page>
```

Setting Variable Values with a Standard Controller

You can use standard Visualforce controllers to set variables or sObject variables by passing in data from a record. This example sets the initial value of myVariable to the Visualforce expression {!account} when the interview starts.

```apex
<apex:page standardController="Account" tabStyle="Account">
    <flow:interview name="flowname">
        <apex:param name="myVariable" value="{"account}"/>
    </flow:interview>
</apex:page>
```

Setting an sObject Collection Variable Value with a Standard List Controller

Because sObject collection variables represent an array of values, you must use a standard list controller or a custom Apex controller. This example sets myCollection to the value of {!accounts} when the interview starts.

```apex
<apex:page standardController="Account" tabStyle="Account" recordSetVar="accounts">
    <flow:interview name="flowname">
        <apex:param name="myCollection" value="{"accounts}"/>
    </flow:interview>
</apex:page>
```

Setting Variable Values with a Custom Apex Controller

If you need finer control over your Visualforce page than a standard controller allows, write a custom Apex controller that sets the variable value, and then reference that controller in your Visualforce page. This example uses Apex to set myVariable to a specific account’s Id when the interview starts.

```apex
public class MyCustomController {
    public Account apexVar {get; set;}

    public MyCustomController() {
        apexVar = [
            SELECT Id, Name FROM Account
            WHERE Name = 'Acme' LIMIT 1];
    }
}
```

```apex
<apex:page controller="MyCustomController">
    <flow:interview name="flowname">
        <apex:param name="myVariable" value="{"apexVar}"/>
    </flow:interview>
</apex:page>
```
This example uses Apex to set an sObject collection variable `myAccount` to the `Id` and `Name` field values for every record with a Name of \texttt{Acme}.

```apex
public class MyCustomController {
    public Account[] myAccount { 
        get {
            return [ 
                SELECT Id, Name FROM account
                WHERE Name = 'Acme'
                ORDER BY Id
            ] ;
        } 
        set { 
            myAccount = value;
        } 
    } 
    public MyCustomController () {
    }
}
```

```
<apex:page id="p" controller="MyCustomController">
    <flow:interview id="i" name="flowname">
        <apex:param name="accountColl" value="{"myAccount}"/>
    </flow:interview>
</apex:page>
```

**Setting Variable Values with an Interview Map**

This example uses an Interview map to set the value for `accVar` to a specific account's Id when the interview starts.

```apex
public class MyCustomController {
    public Flow.Interview.TestFlow myflow { get; set; }

    public MyCustomController() {
        Map<String, Object> myMap = new Map<String, Object>();
        myMap.put('accVar', [SELECT Id FROM Account
            WHERE Name = 'Acme' LIMIT 1]);
        myflow = new Flow.Interview.ModemTroubleShooting(myMap);
    }
}
```

```
<apex:page controller="MyCustomController">
    <flow:interview name="flowname" interview="{"myflow}"/>
</apex:page>
```

Here's a similar example that sets the value for `accVar` to a new account when the interview starts.

```apex
public class MyCustomController {
    public Flow.Interview.TestFlow myflow { get; set; }

    public MyCustomController() {
        Map<String, List<Object>> myMap = new Map<String, List<Object>>();
        myMap.put('accVar', new Account(name = 'Acme'));
        myflow = new Flow.Interview.ModemTroubleShooting(myMap);
    }
}
```
This example uses a map to add two values to a string collection variable (stringCollVar) and two values to a number collection variable (numberCollVar).

```java
public class MyCustomController {
    public Flow.Interview.flowname MyInterview { get; set; }

    public MyCustomController() {
        String[] value1 = new String[]{'First', 'Second'};
        Double[] value2 = new Double[]{999.123456789, 666.123456789};
        Map<String, Object> myMap = new Map<String, Object>();
        myMap.put('stringCollVar', value1);
        myMap.put('numberCollVar', value2);
        MyInterview = new Flow.Interview.flowname(myMap);
    }
}
```

Get Flow Variable Values to a Visualforce Page

Flow variable values can be displayed in a Visualforce page. Once you’ve embedded your flow in a Visualforce page, you can use Visualforce markup to get values for variables or sObject variables. To display values for a collection variable or an sObject collection variable, you can use Visualforce markup to get the individual values contained in the collection.

Note: You can get only variables that allow output access. For each flow variable, output access is controlled by:

- The Input/Output Type variable field in the Cloud Flow Designer
- The isOutput field on FlowVariable in the Metadata API

For a variable that doesn’t allow output access, attempts to get the variable are ignored, and compilation may fail for the Visualforce page, its `<apex:page>` component, or the Apex class.

The following example uses an Apex class to get an sObject variable value from a flow and then displays it in a Visualforce page.

```java
public class FlowController {
    public Flow.Interview.flowname myflow { get; set; }
    public Case apexCaseVar;
    public Case getApexCaseVar() {
        return myflow.caseVar;
    }
}
```

<apex:page controller="FlowController" tabStyle="Case">
    <flow:interview name="flowname" interview="{!myflow}" />
    <apex:outputText value="Default Case Priority: {!apexCaseVar.Priority}" />
</apex:page>
The following example uses an Apex class to get the values that are stored in a string collection variable (emailsCollVar) in the flow and then uses a Visualforce page to run the flow interview. The Visualforce page iterates over the flow’s collection variable and displays the values for each item in the collection.

```apex
public class FlowController {
    public Flow.Interview.flowname myflow { get; set; }

    public List<String> getVarValue() {
        if (myflow == null) {
            return null;
        } else {
            return (List<String>)myflow.emailsCollVar;
        }
    }
}

<apex:page controller="FlowController">
    <flow:interview name="flowname" interview="{"myflow}" />
    <apex:repeat value="{"varValue}" var="item">
        <apex:outputText value="{"item}"/><br/>
    </apex:repeat>
</apex:page>
```

The following example uses an Apex class to set the flow to `{!myflow}` and then uses a Visualforce page to run the flow interview. The Visualforce page uses a data table to iterate over the flow’s sObject collection variable and display the values for each item in the collection.

```apex
public class MyCustomController {
    public Flow.Interview.flowname myflow { get; set; }
}

<apex:page controller="MyCustomController" tabStyle="Account">
    <flow:interview name="flowname" interview="{"myflow}" reRender="nameSection" />
    <!-- The data table iterates over the variable set in the "value" attribute and sets that variable to the value for the "var" attribute, so that instead of referencing "{"myflow.collectionVariable}" in each column, you can simply refer to "account".-->
    <apex: dataTable value="{"myflow.collectionVariable}" var="account" rowClasses="odd,even" border="1" cellpadding="4">
        <!-- Add a column for each value that you want to display.-->
        <apex:column>
            <apex: facet name="header">Name</apex:facet>
            <apex:outputlink value="/{"account['Id']}"">
                {!account['Name']}
            </apex:outputlink>
        </apex:column>
        <apex:column>
            <apex: facet name="header">Rating</apex:facet>
            <apex:outputText value="{"account['Rating']}" />
        </apex:column>
        <apex:column>
            <apex: facet name="header">Billing City</apex:facet>
            <apex:outputText value="{"account['BillingCity]}" />
        </apex:column>
    </apex: dataTable>
</apex:page>
```
Depending on the contents of the sObject collection variable in your flow, here’s what that data table looks like.

<table>
<thead>
<tr>
<th>Name</th>
<th>Rating</th>
<th>Billing City</th>
<th>Employees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Global Media</td>
<td>Hot</td>
<td>Toronto</td>
<td>14668</td>
</tr>
<tr>
<td>ABC Labs</td>
<td>Warm</td>
<td>San Jose</td>
<td>120</td>
</tr>
<tr>
<td>Canson</td>
<td>Hot</td>
<td>Ohta-ku, Tokyo</td>
<td>125</td>
</tr>
<tr>
<td>Acme Inc</td>
<td>Hot</td>
<td>Atlanta</td>
<td>680</td>
</tr>
<tr>
<td>Ecotech - Switzerland</td>
<td>Cold</td>
<td>Geneva</td>
<td>3500</td>
</tr>
<tr>
<td>Informatica Global</td>
<td>Warm</td>
<td>Buenos Aires</td>
<td>300</td>
</tr>
<tr>
<td>Lutron Technologies</td>
<td>Hot</td>
<td>Murray Hill</td>
<td>200</td>
</tr>
<tr>
<td>Sapient-UK</td>
<td>Cold</td>
<td>London</td>
<td>80</td>
</tr>
<tr>
<td>Target</td>
<td>Warm</td>
<td>Anaheim</td>
<td>1200</td>
</tr>
</tbody>
</table>

Configure 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 organization.

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
<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
<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.

- `getPageA` instantiates a new page reference by passing a string to define the location.
- `getPageB` returns a string that is treated like a PageReference.
- `getPageC` returns a string that gets translated into a PageReference.

```java
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 references that 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 and see the first screen of the flow, without the record. This is 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.
Distribute a Flow to External Users

Enable external users to run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a Force.com site, Customer Portal, or Partner Portal.

For example, you can set up a self-service tool for your public Force.com 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 unique name.
   a. From Setup, click Create > Workflow & Approvals > Flows.
   b. Click the name of the flow.
   c. Copy the unique name of the flow.

2. From Setup, click Develop > 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 unique name of the flow.

   For example:

   ```
   <apex:page>
   <flow:interview name="flowuniquename"/>
   </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 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. Distribute your Visualforce page by taking one of these actions.

   • Add the Visualforce page to your Force.com site.
• Define a custom Visualforce tab by using the Visualforce page, and then add that tab to your portal or community.

SEE ALSO:
- Configure the Flow’s Finish Behavior
- Get Flow Variable Values to a Visualforce Page
- Set Flow Variable Values from a Visualforce Page

Launch a Flow Automatically

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. An autolaunched 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 is run. If there’s no active version, the latest version is run. When a flow admin invokes an autolaunched flow, the latest version is always run.

IN THIS SECTION:
- Start a Flow with a Process
- Start a Flow with a Workflow Action—Pilot
- Invoke a Flow from the Force.com REST API
- Invoke a Flow with Apex

Start a Flow with 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 “Creating Processes” in the Salesforce Help.
3. Add a “Launch a Flow” 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.

Start a Flow with 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 Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

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.

Flow Trigger Considerations (Pilot)

Flow trigger workflow actions have special behaviors and limitations.

Note: The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

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 new flow trigger. From Setup, click Create > Workflow & Approvals > Flow Triggers.
- Only active, autolaunched flows can be launched by flow triggers. However, if a flow trigger is in test mode, administrators 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 taken into account 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 won’t be 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 and sObject variables in the flow, with the following limitations.
  – Flow triggers can’t use multi-select picklist fields to set flow variables or sObject 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 sObject collection variables in flows.

• Always keep one version of the flow active if it’s referenced by an active workflow rule’s flow trigger.

• Once 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 would violate the run restrictions for an autolaunched flow. If you modify a flow to no longer be autolaunched, 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: Record Create, Record Update, Record Delete, Fast Create, Fast Update, or Fast Delete.

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 a Record Update 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.
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 Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

1. From Setup, click **Create > Workflow & Approvals > 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Name of the flow trigger.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>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 are 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.</td>
</tr>
<tr>
<td>Protected Component</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>Flow</td>
<td>Unique name of the autolaunched flow that this workflow action launches.</td>
</tr>
<tr>
<td>Set Flow Variables</td>
<td>Whether to pass values into the flow’s variables and sObject variables.</td>
</tr>
</tbody>
</table>

5. If you select **Set Flow Variables**, specify their names and values.

   Click **Set Another Value** to set up to 25 variables.

6. To put the flow trigger in test mode, select **Administrators run the latest flow version.**

   When selected and an administrator 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 and sObject 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.
Associate the Flow Trigger with a Workflow Rule

Add the flow trigger as an immediate action on your workflow rule.

**Note:** The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

Before you begin, create:

- An autolaunched flow
- A workflow rule
- A flow trigger that launches the autolaunched flow

1. From Setup, click **Create > Workflow & Approvals > Workflow Rules**.
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 drop-down list, select Flow Trigger.
   
   The Available Actions box lists all existing flow triggers.

6. Select the flow trigger to associate with this workflow rule. Use the right arrow to move the flow trigger to Selected Actions.
7. Click **Save**.

Invoke a Flow from the Force.com REST API

Use the Custom Invocable Actions endpoint to invoke an autolaunched flow from the Force.com REST API.

**Example:** This example invokes the active version of the flow "Escalate_to_Case".

```plaintext
POST /v33.0/actions/custom/flow/Escalate_to_Case
```

The request sets values for two of the flow’s input variables: CommentCount and FeedItemId. Once invoked, the flow checks whether:

- A given feed item has more than five comments and
- A best comment hasn’t been selected yet

```json
{
    "inputs" : [ {
        "CommentCount" : 6,
        "FeedItemId" : "0D5D0000000cfMY"
    } ]
}
```
Invoke a Flow with Apex

Use the `start` method in the `Flow.Interview` class to launch an autolaunched flow or user provisioning flow from Apex.

**Example:** The following includes a sample controller that starts a flow and the corresponding Visualforce page. The Visualforce page contains an input box and a start button. When the user enters a number in the input box and clicks `Start`, the controller’s `start` method is called. The button saves the user-entered value to the flow’s `input` variable and launches the flow using the `start` method. The flow doubles the value of `input` and assigns it to the `output` variable, and the output label displays the value for `output` by using the `getVariableValue` method.

```java
public class FlowController {
    // Instance of the Flow
    public Flow.Interview.doubler myFlow {get; set;}
    public Double value {get; set;}

    public Double getOutput() {
        if (myFlow == null) return null;
        return (Double)(myFlow.getVariableValue('v1'));
    }

    public void start() {
        Map<String, Object> myMap = new Map<String, Object>();
        myMap.put('v1', input);
        myFlow = new Flow.Interview.doubler(myMap);
        myFlow.start();
    }
}
```

The following is the Visualforce page that uses the sample flow controller.

```visualforce
<apex:page controller="FlowController">
    <apex:outputLabel id="text">v1 = {!output}</apex:outputLabel>
    <apex:form>
        value : <apex:inputText value="{"output}"/>
        <apex:commandButton action="{"start}" value="Start" reRender="text"/>
    </apex:form>
</apex:page>
```

**IN THIS SECTION:**

- `start()`: Invokes an autolaunched flow or user provisioning flow.
- `getVariableValue(variableName)`: Returns the value of the specified flow variable. The flow variable can be in the flow embedded in the Visualforce page, or in a separate flow that is called by a subflow element.

**start()**

Invokes an autolaunched flow or user provisioning flow.
Signature

public Void start()

Return Value

Type: Void

Usage

This method can be used only with flows that have one of these types.

• Autolaunched Flow
• User Provisioning Flow

For details, see “Flow Types” in the Visual Workflow Guide.

When a flow user invokes an autolaunched flow, the active flow version is run. If there’s no active version, the latest version is run. When a flow admin invokes an autolaunched flow, the latest version is always run.

getVariableValue(variableName)

Returns the value of the specified flow variable. The flow variable can be in the flow embedded in the Visualforce page, or in a separate flow that is called by a subflow element.

Signature

public Object getVariableValue(String variableName)

Parameters

variableName
  Type: String
  Specifies the unique name of the flow variable.

Return Value

Type: Object

Usage

The returned variable value comes from whichever flow the interview is currently running. If the specified variable can’t be found in that flow, the method returns null.

This method checks for the existence of the variable at run time only, not at compile time.
Deploy a Flow to Other Organizations

Flows created in the Cloud Flow Designer can be included in change sets and packages. The recipient organization of either the change set or package must have Visual Workflow enabled.

IN THIS SECTION:

Change Set Considerations for Flows
Before you use change sets to deploy a flow, understand the limits and unexpected behaviors that are related to component dependencies, deployment, and flow triggers.

Packaging Considerations for Flows
Flows can be included in both managed and unmanaged packages. Before you deploy one, understand the limitations and behaviors of packages that contain flows.

Change Set Considerations for Flows
Before you use change sets to deploy a flow, understand the limits and unexpected behaviors that are related to component dependencies, deployment, and flow triggers.

Component Dependencies
- If you plan to deploy a flow with change sets, consider limitations in migration support. Make sure your flows reference only fields and components that are available in change sets.
- 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 component is referenced by the following flow elements, the Component Dependencies page doesn’t display that component. To deploy the flow successfully, manually add those referenced components to the change set.
  - Apex
  - Email Alerts
  - Post to Chatter
  - Quick Actions
  - Send Email
  - Submit for Approval

  For example, if you use an email alert, manually add the email template that is used by that email alert.

Deployment
- 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. Activate the flow manually 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 change sets creates a version of the flow in the destination organization.

Flow Triggers
Flow triggers aren’t available in change sets.
The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

Packaging Considerations for Flows

Flows can be included in both managed and unmanaged packages. Before you deploy one, understand the limitations and behaviors of packages that contain flows.

Component Dependencies

- If you plan to deploy a flow with packages, consider limitations in migration support. Make sure your flows reference only packageable components and fields.
- Referential integrity works the same for flows as it does for other packaged elements.
- If any of the following elements are used in a flow, packageable components that they reference aren’t included in the package automatically. To deploy the package successfully, manually add those referenced components to the package.
  - Apex
  - Email Alerts
  - Post to Chatter
  - Quick Actions
  - Send Email
  - Submit for Approval

  For example, if you use an email alert, manually add the email template that is used by that email alert.

Flow Status

You can package only active flows. The active version of the flow is determined when you upload a package version. If none of the flow’s versions are active, the upload fails.

Updating Packages

- To update a managed package with a different flow version, activate that version and upload the package again. You don’t need to add the newly activated version to the package. However, if you activate a flow version by mistake and upload the package, you’ll distribute that flow version to everyone. Be sure to verify which version you really want to upload.
- You can’t include flows in package patches.

Other Limitations

- Flow triggers aren’t packageable.

  The Process Builder has superseded flow trigger workflow actions, formerly available in a pilot program. Organizations that are using flow trigger workflow actions can continue to create and edit them, but flow trigger workflow actions aren’t available for new organizations. For information on enabling the Process Builder in your organization, contact Salesforce.

- In a development organization, you can’t delete a flow or flow version after you upload it to a released or beta managed package.

SEE ALSO:

  Installed Flow Considerations
The following terminology is used for Visual Workflow in Salesforce:

**Cloud Flow Designer**
Cloud-based application that lets administrators create a flow for use in Salesforce.

**Connector**
Connectors determine the available paths that a flow can take at run time.

**Element**
Each element represents an action that the flow can execute. Examples of such actions include reading or writing Salesforce data, displaying information and collecting data from flow users, executing business logic, or manipulating data.

**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 the Cloud Flow Designer.

**Flow Interview**
A flow interview is a running instance of a flow.

**Master Flow**
A master flow is a flow that contains a subflow element. The term “master” is used to distinguish it from the flow that is referenced and called by the subflow element.

**Resource**
Each resource represents a value that you can reference throughout the flow.

**Subflow**
A subflow element references another flow, which it calls at run time. The flow that contains the subflow element is referred to as the master flow.

SEE ALSO:
- Cloud Flow Designer Overview
- Visual Workflow Overview
INDEX

<flow:interview>
  usage 134, 137

A
  About subflows 57
  Absolute Alarm Event example 69
  Alarm events
    absolute time 63
    relative time 64
  Apex
    call from a flow 22
  Apex Plug-in element 24
  Approvals
    submit from a flow 55
  Assignment element
    adding 27
    configuring 27
    operators 96

C
  Call Apex element 26
  Chatter
    post from a flow 42
    checkbox input fields 47
  Choice 77
  Cloud Flow Designer
    overview 15
    system requirements 15
  Collection
    example 35, 68
  Collection variable
    example 80
    populate with values 80
  Conditions
    flow 74
  Constant 82
  Create a Flow 20
  Cross-object references in flows 119

D
  Decision element
    adding 28
    configuring 28
  Dynamic Choice 83

E
  Elements
    Apex plug-in 24
    assignment 27
    Call Apex 26
    decision 28
    email alert 52
    fast create 29
    fast delete 30
    fast lookup 31
    fast update 33
    fault connector 115
    fault connector, best practice 117–118
    fault connector, default behavior 116
    fault connector, examples 119
    global actions 41
    loop 34
    object-specific actions 41
    Post to Chatter 42
    record create 36
    record delete 37
    record lookup 38
    record update 40
    screen 44–47, 49–50, 77, 83
    Send Email 53
    step 51
    subflow 57
    Submit for Approval 55
  Email
    send from flow 51, 53
  Email Alert element 52
  Event types, flow
    absolute time alarm 63

F
  Fast Create element 29
  Fast Delete element 30
  Fast Lookup element 31
  Fast Update element 33
  Fault Connector element
    best practice 117–118
    default behavior 116
    examples 119
    flow
      activating 127
Flow

absolute time alarms 63
accessibility 14
administration considerations 12
building blocks 1
call Apex 22
call email alert workflow action 52
conditions 74
connectors, types of 114–115
considerations 3
considerations, Cloud Flow Designer 5
considerations, large flows 6
creating 20
delete waiting interviews 127
deleting 128
delivering to users 129
delivering to users, external 141
delivering to users, internal 129, 133
design considerations 4–6
design considerations, formula 9
designing properties 123
elements overview 20
embedding in Visualforce pages 133, 141
formula limitations 9
global variables 87
in change sets 149
Invoke global action 41
Invoke object-specific action 41
launching from processes 142
launching from workflow rules 143
limits 3
managing 123
managing connectors 17
managing elements 17
managing resources 17
modifying 126
opening 126
operators 95
operators, assignment 96
operators, conditions 101
operators, record filter 109
overview 1
pause, enable 128
process action 142
reference cross-object field values 119
relative time alarms 64
resources overview 75
run time considerations 13
saving 122

Flow (continued)

searching 18
send email 51, 53
setting start element 121
settings 128
sharing 129, 133, 141
submit record for approval 55
testing 126
time-based 10
time-based considerations 10
type 124
wait 60
wait event types 62
wait, waiting conditions 68
workflow action 143
workflow action considerations 143
Flow collection values
getting 137
setting 134
Flow condition
operators 101
Flow connectors
adding 17
editing 17
removing 17
Flow constant values
getting 137
setting 134
Flow Designer
user interface 16
Flow elements
adding 17
editing 17
removing 17
Flow event types
absolute time alarm 63
Flow examples
absolute alarm event 69
loop through a collection 35, 68
populate a String collection variable 80
relative alarm event 70, 72
wait for many events, all 72
wait for many events, any 70
wait for one event 69
Flow finish location
setting via URL 132
Flow formula resource
create 85
limitations 9
Flow interviews
  delete 127
Flow record filters
  operators 109
Flow resources
  adding 17
  editing 17
  formula, create 85
  formula, limitations 9
  removing 17
Flow screen
  validate user input 48
Flow sObject variable values
  getting 137
  setting 134
Flow trigger
  associated with workflow rule 146
  considerations 143
  defining 143
  test mode 143
Flow URL
  setting finish location 132
  setting flow variable values 131
Flow variable values
  getting 137
  setting 134
  setting via URL 131
Flows in packages 149

G
Global action
  Invoke in flow 41
Global action element 41
Global constants 86
Global variables
  in flows 87

L
Limits
  flow 3
Loop element
  adding 34
  configuring 34
  example 35, 68

M
multi-select choice fields 8

O
Object-specific action
  Invoke in flow 41
Object-specific action element 41
Operators
  flow assignment element 96
  flow condition 101
  flow record filters 109

P
Palette, searching 19
Populate variable with values
  collection variable 80
Post to Chatter element 42

R
Record Create element 36
Record Delete element 37
Record Lookup element 38
Record Update element 40
Relative Alarm Event
  example 70, 72
Resources
  choice 77
  Collection variable 78
  constant 82
  dynamic choice 83
  global constants 86
  sObject collection 90
  sObject collection variable 89
  sObject variable 91
  system variables 92
  text template 93
  variable 94
  variable in referenced flow 59

S
Saving a flow 122
Screen element
  about checkbox input fields 47
  about multi-select choice fields 8
  Add a Field tab 46
  add fields 46
  adding 44
  configuring a choice 77
  configuring a dynamic choice 83
  configuring choice fields on the Field Settings tab 49
  configuring output fields on the Field Settings tab 50
  configuring the Field Settings tab 46–47, 49–50
Screen element (continued)
 configuring the General Info tab 45
 configuring user input fields on the Field Settings tab 47
 editing 44
 remove fields 46
 reorder fields 46
 settings 45–47, 49–50, 77, 83
 Searching
 a flow 18
 the Palette 19
 Send Email element 53
 SObject variable 91
 Step element 51
 Subflow element 57
 Submit for Approval element 55
 System variables 92
 T
 Test mode for flow triggers 143
 Text Template 93
 V
 Variable 94
 Variable in referenced flow 59
 Visual Workflow
 administration considerations 12
 considerations 3
 considerations, Cloud Flow Designer 5
 considerations, large flows 6
 Visual Workflow (continued)
 design considerations 4–6
 design considerations, formulas 9
 design considerations, time-based flows 10
 differences from Workflow 2
 formula limitations 9
 run time considerations 13
 Visualforce
 embedding flows 133, 141
 getting flow variable values 137
 setting flow variable values 134
 W
 Wait element
 example 69–70, 72
 Wait event types
 flow 62
 Wait events
 relative time 64
 Wait in a flow
 configure 60
 event types 62
 waiting conditions 68
 Waiting conditions
 flow 68
 Workflow
 differences from Visual Workflow 2
 flow trigger considerations 143
 flow triggers 143