# CONTENTS

**Visual Workflow Guide** .................................................................................. 1
**Which Tool Should I Use?** ............................................................................. 2
**Limits and Considerations** ................................................................. 4
  - Limits .................................................................................. 5
  - Best Practices .................................................................... 12
  - Design Considerations .......................................................... 13
  - Management Considerations ................................................ 27
  - Runtime Considerations .......................................................... 28
  - Accessibility ......................................................................... 29
**Create a Flow** .......................................................................................... 30
  - Building Blocks ..................................................................... 30
  - Cloud Flow Designer ................................................................ 31
  - Set the Start Element .................................................................. 35
  - Save Options ........................................................................ 36
  - Common Tasks ......................................................................... 36
  - Flow Reference ......................................................................... 58
**Samples** .................................................................................................. 139
**Manage Your Flows** ............................................................................... 150
  - Flow and Flow Version Fields ................................................ 151
  - Open and Modify ....................................................................... 152
  - Test ........................................................................................ 153
  - Activate or Deactivate .............................................................. 153
  - Delete an Interview ...................................................................... 154
  - Delete a Version ........................................................................ 154
  - Let Users Pause ......................................................................... 155
**Distribute Your Flows** ............................................................................. 155
  - Runtime Experiences .................................................................... 156
  - Internal Users ........................................................................ 157
  - External Users .......................................................................... 180
  - Systems ................................................................................ 184
  - Other Orgs .............................................................................. 190
**Troubleshoot Your Flows** ...................................................................... 192
  - Error Emails ........................................................................ 192
  - Add Temporary Elements .......................................................... 194
  - Troubleshoot URLs .................................................................... 195
**Terminology** ............................................................................................ 196

**Index** ....................................................................................................... 198
Automate business processes by building applications, known as flows, that collect, update, edit, and create Salesforce information. Then make those flows available to the right users or systems. Flows can either require user interaction—perhaps a wizard or guided UI for data entry—or run in the background on their own—perhaps something that automatically transfers records when a user’s role changes.

IN THIS SECTION:

Which Automation Tool Do I Use?
Salesforce provides multiple tools to automate your organization’s repetitive business processes: Approvals, Process Builder, Workflow, and Visual Workflow.

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

Create a Flow
Once you understand the process that you want to automate, design a flow in the Cloud Flow Designer for that process.

Manage Your Flows
Use the flow detail page to do anything with your flow outside of designing it—such as activating a flow, testing it, or viewing its properties.

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

Why Did My Flow Interview Fail?
To troubleshoot a failed flow interview, use the flow fault email. You can also set up temporary Screen or Send Email elements to identify the problem.

Visual Workflow Terminology
Which Automation Tool Do I Use?

Salesforce provides multiple tools to automate your organization’s repetitive business processes: Approvals, Process Builder, Workflow, and Visual Workflow.

The best automation tool for your needs depends on the type of business process that you’re automating.

- **How a record gets approved**
  
  Example: Managers approve their direct reports’ requests for vacation.

- **What to do when a record has certain values**
  
  Example: Notify the account owner when a related case is escalated.

- **Collecting information from users or customers and then doing something with that information**
  
  Example: Customer support uses a wizard to step through a call script, and cases are created based on the information that they enter.

**How a Record Gets Approved**

For example, when an employee requests time off, that time has to be approved by the employee’s manager. You need to ensure that when a time-off request is submitted for approval, the right person (the employee’s manager) receives the request.

To automate your organization’s processes for approving records, create approval processes.

**What to Do When a Record Has Certain Values**

Three of our tools can address this use case: Workflow, Process Builder, and Visual Workflow. Respectively, these tools create workflow rules, processes, and flows.

We recommend starting with Process Builder, especially for business processes that can be simplified to if/then statements. For example: if a case is escalated, then notify the account owner.

Process Builder includes almost all the functionality that’s available in workflow rules, and more. In fact, a single process can do what it would normally take multiple workflow rules to do. The only thing you can do with workflow that you can’t do with processes is send outbound messages without code. However, you can work around this limitation by calling Apex code from a process.

If the process is too complicated for the Process Builder or requires more advanced functionality, create a flow by using the Cloud Flow Designer. For example, create a flow to:

- **Use complex branching logic (if certain conditions are true, evaluate for further conditions)**
  
  Example: First, check whether a case is escalated. If the case is escalated, check the account’s region and route the case accordingly.

- **Sort through, iterate over, and operate on several records**
  
  Example: After an opportunity is closed and won, calculate the opportunity’s discount. Then apply that discount to all the related opportunity products.
Getting Information from Users or Customers and Then Doing Something with It

If you need to build a wizard to collect information, Visual Workflow is the tool for you. Create a flow that displays information to and requests information from a user. Then take the information that they enter and perform actions in Salesforce with it.

For example, create a flow that walks customer support representatives through a call script. The flow uses information that the representative entered, such as the caller’s name and account number, to create a case that’s assigned to the right person.

You can add more complexity to the flow to match your business process. For example:

- Route the representative to different screens, depending on earlier choices. This prevents the representative from doing things like trying to upsell a product to a customer who already bought that product.
- Check whether the reported problem is blocking the customer’s business and the account is high-value. If so, the flow notifies the region director.

Automation Tool Features

Here’s the breakdown of all the features and actions that are supported in each of our automation tools. Use it to figure out which tool is best for your business needs.

<table>
<thead>
<tr>
<th></th>
<th>Process Builder</th>
<th>Visual Workflow</th>
<th>Workflow</th>
<th>Approvals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Complexity</strong></td>
<td>Multiple if/then statements</td>
<td>Complex</td>
<td>A single if/then statement</td>
<td>A single if/then statement</td>
</tr>
<tr>
<td><strong>Visual designer</strong></td>
<td>✅</td>
<td>✅</td>
<td>✅</td>
<td></td>
</tr>
<tr>
<td><strong>Browser support</strong></td>
<td>All (Chrome recommended)</td>
<td>All (Safari not recommended)</td>
<td>All</td>
<td>All</td>
</tr>
</tbody>
</table>
| **Starts when**           | Record is changed
  - Invoked by another process
  - User clicks button or link
  - User accesses custom tab
  - Process starts
  - Apex is called | Record is changed
  - User clicks button or link
  - Process or flow starts that includes a Submit for Approval action
  - Apex is called |
| **Supports time-based actions** | ✅ | ✅ | ✅ |
| **Supports user interaction** | ✅ | | |

**Supported Actions**

<table>
<thead>
<tr>
<th></th>
<th>Process Builder</th>
<th>Visual Workflow</th>
<th>Workflow</th>
<th>Approvals</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Call Apex code</strong></td>
<td>✅</td>
<td>✅</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Create records</strong></td>
<td>✅</td>
<td>✅</td>
<td>Tasks only</td>
<td>Tasks only</td>
</tr>
<tr>
<td><strong>Invoke processes</strong></td>
<td>✅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Delete records</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The Process Builder has superseded flow trigger workflow actions, previously available in a pilot program. Orgs that are using flow trigger workflow actions can continue to create and edit them, but they aren’t available for new orgs.

### Limits and Considerations for Visual Workflow

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

**IN THIS SECTION:**

- **Limits for Visual Workflow**
  When using Visual Workflow, keep flow limits and Apex governor limits in mind.

- **Flow Best Practices**
  Before you begin building and distributing flows, understand the best practices.

- **Considerations for Designing Flows**
  When you design flows, keep certain guidelines in mind.

- **Considerations for Managing Flows**
  When managing flows, consider the administration and activation limits.

- **Considerations for Running Flows**
  When you run or test a flow, keep the limits and guidelines in mind.
Flow Accessibility
Visual Workflow is 508-compliant with a few exceptions.

SEE ALSO:
- Cloud Flow Designer
- Considerations and Limitations for Flows in Lightning Pages
- Considerations for Translating Flows

Limits for Visual Workflow

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

<table>
<thead>
<tr>
<th>Limit Description</th>
<th>Limit</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>2,000</td>
</tr>
<tr>
<td>Maximum number of active flows and processes per org</td>
<td>500</td>
</tr>
<tr>
<td>Maximum number of flows and processes per org</td>
<td>1,000</td>
</tr>
<tr>
<td>Maximum number of flow interviews or groups of scheduled actions (from processes)</td>
<td>30,000</td>
</tr>
<tr>
<td>Maximum number of flow interviews that are resumed or groups of scheduled actions</td>
<td>1,000</td>
</tr>
<tr>
<td>Maximum number of relative time alarms defined in flow versions or schedules</td>
<td>20,000</td>
</tr>
<tr>
<td>based on a field value in processes</td>
<td></td>
</tr>
</tbody>
</table>

IN THIS SECTION:

Apex Governor Limits that Affect Flows
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.

Flows in Transactions
Each flow interview runs in the context of a transaction. A transaction represents a set of operations that are executed as a single unit. For example, a transaction can execute Apex triggers and escalation rules in addition to a flow interview. If one interview in a transaction fails, all the interviews in the transaction are rolled back, as well as anything else the transaction did. The transaction doesn’t retry any of the operations—including the flow interview.
Flow Bulkification in Transactions
Programmers can design their code so that similar actions are performed together in one batch. For example, one operation to create 50 records rather than 50 separate operations that each create one record. This process is called bulkification, and it helps your transaction avoid governor limits. If you’re working with flows, you don’t even have to think about bulkification. Flow interviews bulkify actions for you automatically.

SEE ALSO:
Visual Workflow
Limits and Considerations for Visual Workflow

Apex Governor Limits that Affect Flows
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>

1 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 when 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:
Apex Developer Guide: Execution Governors and Limits
Limits for Visual Workflow
Flows in Transactions

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

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

IN THIS SECTION:

When Does a Flow’s Transaction Start?
Depending on how the flow was distributed, a transaction that runs an interview for that flow starts in different ways.

When Does a Flow’s Transaction End?
When a transaction ends depends on whether the flow contains certain elements and whether it originally started because a record was changed.

SEE ALSO:
Flow Bulkification in Transactions

When Does a Flow’s Transaction Start?
Depending on how the flow was distributed, a transaction that runs an interview for that flow starts in different ways.

<table>
<thead>
<tr>
<th>Distribution Method</th>
<th>Transaction starts when...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process Builder¹</td>
<td>A record is created or updated.</td>
</tr>
<tr>
<td>Flow URL</td>
<td>The URL is accessed.</td>
</tr>
<tr>
<td>Custom button or link</td>
<td>The button or link is clicked.</td>
</tr>
<tr>
<td>Visualforce page</td>
<td>The page is accessed.</td>
</tr>
<tr>
<td>Interview.start() method</td>
<td>If the method starts via a before or after trigger, the transaction starts when a record is created or updated. Otherwise, the transaction starts when the method (or a parent method) is invoked. The start() method shares its limits with other operations in the transaction and other methods in the class.</td>
</tr>
<tr>
<td>REST API (Custom Actions or Flows resource)</td>
<td>When the REST call is made. Depending on how the REST call is implemented, the limits can be shared with other operations.</td>
</tr>
</tbody>
</table>
When Does a Flow’s Transaction End?

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

The transaction ends when:

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

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

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

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

<table>
<thead>
<tr>
<th>Transaction Operation</th>
<th>DML Statement Used</th>
<th>SOQL Query Used</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  Cases are saved to the database, but aren’t committed yet.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2  Case assignment rules are executed. Each case’s owner is updated.</td>
<td></td>
<td>✅</td>
</tr>
<tr>
<td>3  Case escalation rules are executed. If any case has been open for 10 days, an email is sent to the owner.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4  Process is started.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5  Process looks up the case’s account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6  If the account is hot, process uses Chatter to notify the account owner that there’s a new case associated with the account.</td>
<td></td>
<td>✅</td>
</tr>
<tr>
<td>7  Process launches a flow interview.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8  Flow interview looks up the parent account and how many cases it has.</td>
<td></td>
<td>✅</td>
</tr>
<tr>
<td>9  Flow interview checks whether the account has more than five open cases.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10 If it does, flow interview looks up the account’s division manager then posts on the account’s Chatter feed to notify the division manager and account owner.</td>
<td></td>
<td>✅</td>
</tr>
</tbody>
</table>
Flow Bulkification in Transactions

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

IN THIS SECTION:
  How Does Flow Bulkification Work?
  Interview operations are bulkified only when they execute the same element. That means that the interviews must all be associated with the same flow.
  Which Flow Elements Can Be Bulkified?
  Flows can bulkify any element that performs a DML statement or SOQL query or does something else external to the flow, like sending an email.
  Example of Flow Bulkification
  This example demonstrates how operations are bulkified for a flow when 100 cases are updated through Data Loader.

SEE ALSO:
  Flows in Transactions

How Does Flow Bulkification Work?

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

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

If, despite the bulkification, any interview hits a governor limit, all the interviews in the transaction fail. Any operations that the interviews performed are rolled back, and the transaction doesn’t try to perform the operations again.
Example: When you upload 100 cases, the flow MyFlow_2 triggers one interview for each case.

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

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

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

Which Flow Elements Can Be Bulkified?

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

**Elements that create, update or delete records**

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

- Create elements (Record Create, Fast Create)
- Update elements (Record Update, Fast Update)
- Delete elements (Record Delete, Fast Delete)
- Quick Action elements
- Post to Chatter elements
- Submit for Approval elements
- Apex elements—depending on your organization (invocable Apex only)

**Elements that look up records**

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

- Lookup elements (Record Lookup, Fast Lookup)
- Record Update elements
- Record Delete elements
- Apex elements—depending on your organization (invocable Apex only)

**Elements that send emails**

- Send Email elements
- Email Alert elements
- Apex elements—depending on your organization (invocable Apex only)

**Note:**

- Unlike invocable Apex, Apex Plug-in elements aren’t bulkified.
- Although invocable Apex is bulkified, the flow has no way of knowing what the invoked methods’ operations are. If you want those operations to also be bulkified, make sure the code follows bulkification best practices.

SEE ALSO:

*Apex Developer Guide: Running Apex within Governor Execution Limits*
Example of Flow Bulkification

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

The Associated Flow

You’ll understand the concepts better if you understand the design of the associated flow.

The flow:

1. Looks up the case’s parent account and how many open cases that account has.
2. Checks whether the account has more than five cases open.
3. If the account has more than five open cases:
   a. Looks up the division manager for the account.
   b. Posts on the account’s Chatter feed to notify the division manager and the account owner.
4. If the account has five or fewer open cases, posts on the account’s Chatter feed to notify only the account owner.

The Bulkified Interviews

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

The first interview goes through the Record Lookup element (1). Because Record Lookups can be bulkified, the interview waits there until all the other interviews have done the same. Then, Salesforce executes all the Record Lookup operations together (because they're all for the same element in the same flow). Instead of 100 SOQL queries, the transaction issues one SOQL query.
The first interview is evaluated by the Decision element (2). The account has six cases, so the interview is routed down the “More than 5” path. The interview proceeds to the second Record Lookup element (3a). Because it’s a bulkifiable element, the interview waits there.

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

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

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

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

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

Flow Best Practices

Before you begin building and distributing flows, understand the best practices.

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

**Build your flows in a test environment—like a sandbox or Developer Edition org.**
The last thing you want to do is accidentally change records in your company’s production org. Build your flows in a separate environment. That way, you can enter fake data and test various permutations of your flow without worrying about changing or deleting data that your users actually need.

**Never hard-code Salesforce IDs.**
IDs are org-specific, so don’t hard-code new or existing IDs. Instead, let Salesforce create the IDs, and pass them into variables when the flow starts. You can do so, for example, by using merge fields in URL parameters or by using a lookup element.

**Wait until the end of the flow to make changes to the database.**
Have you heard about flow limits? Because flows operate under Apex governor limits, the sky is not the limit. To avoid hitting those limits, we recommend bunching all your database changes together at the end of the flow, whether those changes create, update, or delete records.

**Control when running users can navigate backward.**
If the flow commits changes to the database between two screens, don’t let users navigate from the later screen to the previous screen. Otherwise, the flow can make duplicate changes to the database.

**Provide an error handler.**
Sad to say, but sometimes a flow doesn’t perform an operation that you configured it to do. Perhaps the flow is missing crucial information, or the running user doesn’t have the required permissions. By default, the flow shows an error message to the user and emails the admin who created the flow. However, you can control that behavior. See Customize What Happens When a Flow Fails for more information and recommendations.
Save early and often.
Sometimes the Cloud Flow Designer falls victim to unexpected problems, like losing Internet access. Salesforce doesn’t save your changes automatically, so it’s up to you to save your work. Save as often as possible, so that you don’t accidentally lose a few hours’ worth of work.

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

Considerations for Designing Flows

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.

Manipulating Percentage Values
Test your flows carefully if they use sObject variables to manipulate percentage values. When you insert a value into an sObject variable’s percentage field and then reference that field in a formula, the value is automatically divided by 100.

For example, an opportunity’s Probability field is set to 100. If you assign that value to sObject variable {!Opportunity.Probability}, the value is still 100. But if you create a formula whose expression is {!Opportunity.Probability}, the value is 1.

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}.

Boolean Types Treat null Differently than false
Flow treats null as a different value than false. For example, if you try to find a record whose checkbox field is set to null, no records are returned. Instead, look for records where the checkbox field is set to false. If you’re using a variable (such as myCheckbox = {!varBoolean}), make sure that the variable isn’t set to null before you reference it in your record filter or condition.

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.

Working with Person Accounts
If your org uses person accounts, reference Contact.Salutation instead of Account.Salutation.

Shield Platform Encryption
Encrypted fields aren’t supported for filtering or sorting records. This limitation applies to the following elements and resources.

- Record Update element
- Record Delete element
- Record Lookup element
- Fast Lookup element
- Dynamic Record Choice resource
External Objects
External objects aren't supported in flows.

IN THIS SECTION:
Considerations for the Cloud Flow Designer
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.

Guidelines for Working with Large Flows
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.

Best Practices for Designing Flows for Translation
Review these best practices if you plan to translate flows.

Considerations for Two-Column Flows
If your org has Lightning runtime enabled, you can control whether a flow displays in one column or two columns. Before you use this feature, understand how the flow layout currently behaves.

Limitations for Multi-Select Choice Fields
Multi-select checkboxes and multi-select picklist fields let flow users select multiple choices in a screen field. Before you start using multi-select choice fields, understand how they work in flows, both when you design the flows and when your users run them.

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

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

Flow Operations and Read-Only Fields
Understand when flows have read-only access to field values. You can control the behavior when a flow tries to update a read-only field and remove read-only field values from flow operations.

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

Considerations for the Cloud Flow Designer
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:
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”.

Guidelines for Working with Large Flows

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.
**Best Practices for Designing Flows for Translation**

Review these best practices if you plan to translate flows.

- Keep your labels as short as possible, so that the translated label doesn’t exceed 1,000 characters (or 255 characters for definition name and version name). If you have long labels for a display text field, consider breaking it up into multiple fields.
- When you update a master label, check if it has translations and update as needed.
- Avoid text templates if you need to translate an email body or other formatted block text.

**Considerations for Two-Column Flows**

If your org has Lightning runtime enabled, you can control whether a flow displays in one column or two columns. Before you use this feature, understand how the flow layout currently behaves.

**Granularity**

The layout setting is applied at the flow level. So you can’t control the layout at the screen or field level. If you set a flow to use two columns, every screen in that flow displays in two columns.

**Order of Fields**

You can’t manually control which fields go in which columns. If the flow is set to display two columns, the fields alternate in each column. The odd fields (first, third, fifth, and so on) are placed in the left column. The even fields (second, fourth, sixth, and so on) are placed in the right column.

If your users navigate screens with the TAB key, they’ll tab through all the fields in the left column and then all the fields in the right column. You can’t configure the fields to tab left-to-right.

**Responsiveness**

The flow layout isn’t responsive to the user’s screen dimensions. It uses the same layout whether the user’s screen is one inch wide or twenty inches wide.
Tip: Don’t apply two-column layout to a flow if users will run it from a phone or small tablet.

SEE ALSO:
- Limits and Considerations for Visual Workflow
- Considerations and Limitations for Flows in Lightning Pages
- Render Two-Column Screens from a Flow URL

**Limitations for Multi-Select Choice Fields**

Multi-select checkboxes and multi-select picklist fields let flow users select multiple choices in a screen field. Before you start using multi-select choice fields, understand how they work in flows, both when you design the flows and when your users run them.

**Configuring a Multi-Select Resource Field**

- A multi-select choice field can have only one default value.
- A dynamic record 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 record 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 record choice, the variable assignments obey the first of those fields.

**Using Values from a Multi-Select Resource Field**

- At run time, a multi-select field’s value is a concatenation of the user-selected choice values, separated by semicolons. If any of the selected choices’ values included semi-colons, those semi-colons are removed.
- If you referenced multi-select choice fields in flow conditions, follow these best practices.
  - 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:
- Flow Screen Element: Choice Fields

**Limitations for Flow Formulas**

When you create a formula resource or add validation to a screen input field, understand the formula limitations 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
For a complete list of operators and functions for building formulas in Salesforce, see Formula Operators and Functions.

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

Here’s an example. `varContract` refers to 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 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 formula expression has an error at run time, it resolves to null.
- If a flow contains an invalid formula resource, you can’t activate the flow.
- To reference a platform event in a formula, pass the event data into an sObject variable in the Wait element. Then reference the appropriate field in that sObject variable.

**SEE ALSO:**
- Flow Formula Resource
- Flow Resources

### Limitations for Waiting Flows

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 org can process up to 1,000 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 org exceeds this limit, Salesforce processes the remaining events in the next hour. For example, an org has 1,200 events scheduled to be processed between 4:00 PM and 5:00 PM. Salesforce processes 1,000 events between 4:00 PM and 5:00 PM and the additional 200 events between 5:00 PM and 6:00 PM.
- An org can have up to 30,000 interviews waiting at a given time.
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 Limits for Visual Workflow on page 5.

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.

limitations for platform events

Tip: Make sure to also review the considerations for platform events.

formulas

To reference a platform event in a formula, pass the event data into an sObject variable in the Wait element. Then reference the appropriate field in that sObject variable.

value truncation

In event conditions, values can’t be more than 765 characters.
Subscriptions Related List
On the platform event’s detail page, the Subscriptions related list shows which entities are waiting for notifications of that platform event to occur. When at least one flow interview is waiting for that platform event to occur, a “Process” subscriber appears in the Subscriptions related list.

Uninstalling Events
Before you uninstall a package that includes a platform event, delete the interviews that are waiting for the event to occur.

Limitations for General Alarms
- 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 reschedules 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 reschedules the interview within an hour.

Limitations for Absolute Time Alarms
- Absolute time alarms are evaluated based on the time zone of the user who created the flow.

Limitations for Relative Time Alarms
- Relative time alarms are evaluated based on the org’s time zone.
- Across all your flow versions, your org 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 reschedules 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:
- Platform Events Developer Guide: Considerations
- Considerations for Designing Flows
- Limits and Considerations for Visual Workflow
- Operators in Flow Conditions
- Flow Wait Element

Flow Operations and Read-Only Fields

Understand when flows have read-only access to field values. You can control the behavior when a flow tries to update a read-only field and remove read-only field values from flow operations.

IN THIS SECTION:

- Which Fields Are Inaccessible When a Flow Creates or Updates Records?
- Control What Happens When a Flow Tries to Set Values for Read-Only Fields
- Remove Read-Only Fields from an sObject Variable

Which Fields Are Inaccessible When a Flow Creates or Updates Records?

A flow can perform an operation only if the running user has permission to do so. When a flow tries to create or update records, fields that the running user can’t edit are considered inaccessible, or read only. A field can be inaccessible because the user hasn’t been granted permission to edit the field or because it’s a system field that’s always read only.

To determine which fields are system fields, see the Object Reference for Salesforce and Force.com. To determine which other fields aren’t editable, review the running user’s permissions.
How Did Read-Only Fields Get in My sObject Variable?

<table>
<thead>
<tr>
<th>If the Variable Is Populated by ...</th>
<th>The Variable Includes ...</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Fast Lookup element</td>
<td>Id and any other read-only fields that you choose to include.</td>
</tr>
<tr>
<td>An Assignment or Record Lookup element</td>
<td>Any read-only fields that you choose to include.</td>
</tr>
<tr>
<td>A process or a workflow rule</td>
<td>All the object’s system fields and any fields that the running user doesn’t have permission to edit. The variable includes every field for the object by default.</td>
</tr>
</tbody>
</table>

What Do I Do When My sObject Variable Includes Read-Only Fields?

For each read-only field that’s stored in your sObject variable:

1. Determine whether the flow uses that field anywhere. If it doesn’t, update the flow so that it doesn’t store a value for that field. This suggestion applies only if the variable is populated by an element in the flow, like Fast Lookup.

   For example, a Fast Lookup element stores CreatedByDate, but no other elements reference that field. You update the Fast Lookup so that it’s no longer storing CreatedByDate.

2. If the read-only field is referenced in the flow, give the running users the permissions needed for the flow to execute its operations.

3. If you can’t give the running users the needed permissions for a field, update the flow so that it doesn’t try to update that field.

   **Example:** Using a Fast Update element, a flow updates several fields on an account. While your users can edit Description and Account Rating, they can’t edit Owner ID or LastModifiedDate. To prevent the flow from failing at runtime:
   - Give your users “Edit” permission for Owner ID.
   - Copy only the writable field values (Description, Account Rating, and Owner ID) from the original sObject variable into a new sObject variable. Reference the new sObject variable in the Fast Update element.

   Copying only the writable field values ensures that the flow doesn’t try to set a value for LastModifiedDate at runtime.

SEE ALSO:

- Remove Read-Only Fields from an sObject Variable
- Control What Happens When a Flow Tries to Set Values for Read-Only Fields
- Object Reference for Salesforce and Force.com: System Fields
Control What Happens When a Flow Tries to Set Values for Read-Only Fields

When creating or updating records, the flow sets values for specific fields. But what happens if the running user doesn’t have edit access to all those fields? For Fast Create and Fast Update elements, that’s up to you. To control the behavior, select or deselect the Filter Inaccessible Fields from Flow Requests preference.

A flow request is when a flow tries to perform an operation, such as create or update records.

<table>
<thead>
<tr>
<th>When Filter Inaccessible Fields from Flow Requests is</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selected</strong></td>
<td><strong>Not Selected</strong> (Recommended)</td>
<td></td>
</tr>
<tr>
<td>Result when the running user doesn’t have edit access to all fields</td>
<td>The operation partially succeeds. The flow filters read-only fields out of the operation. The fields that the user can edit are updated. The fields that the user can’t edit aren’t updated. The flow doesn’t execute the fault path.</td>
<td>The operation fails. No fields in the operation are updated. The flow executes the fault path if there is one.</td>
</tr>
<tr>
<td>Notification when one or more fields aren’t updated</td>
<td>No notification is sent to the user or admin to indicate that some fields weren’t updated.</td>
<td>The admin receives a flow error email with full details.</td>
</tr>
<tr>
<td>Compared to Record Create and Record Update elements</td>
<td>Inconsistent</td>
<td>Consistent</td>
</tr>
</tbody>
</table>

**Tip:** We recommend disabling this preference so that you always know when a flow doesn’t set all expected field values.

1. From Setup, enter Automation in the Quick Find box, then select Process Automation Settings.
2. Select or deselect Filter Inaccessible Fields from Flow Requests.

If your org was created in Winter ’17 or earlier, the preference is enabled by default. Otherwise, the preference is disabled by default.

**Example:** Using a Fast Update element, a flow updates several fields on an opportunity. At runtime, the flow tries to update the Acme account on behalf of your user. The user can edit Stage and Close Date but not Amount. As a result, the flow doesn’t have permission to update Amount.

- If Filter Inaccessible Fields from Flow Requests is selected, the flow successfully updates the account, but it only updates Stage and Close Date. The flow doesn’t notify anybody that Amount wasn’t updated.
- If Filter Inaccessible Fields from Flow Requests is not selected, the flow fails to update the account. The admin receives a flow error email. The email includes this error.

`INVALID_FIELD_FOR_INSERT_UPDATE: Unable to create/update fields: Amount` That’s API-speak for “The running user doesn’t have permission to edit the Amount field.”
**Warning:** If you change your org’s selection for this preference, use a sandbox to test how the change impacts your flows. Consider following the same process as you would for a critical update.

**SEE ALSO:**
Which Fields Are Inaccessible When a Flow Creates or Updates Records?

### Remove Read-Only Fields from an sObject Variable

If a flow tries to update fields that the running user can’t edit and Filter Inaccessible Fields from Flow Requests is not enabled for your org, the flow fails. If your sObject variable includes read-only fields and you can’t grant your running users “Edit” permissions for those fields, remove the fields from the sObject variable. Use a Record Create or Record Update element instead of a Fast Create or Fast Update element, or copy the writable field values into a new sObject variable.

**Note:** If the read-only fields are populated in the sObject variable in a Fast Lookup or Assignment element, consider updating those elements so that they don’t populate that field at all.

### IN THIS SECTION:

**Copy Field Values from One sObject Variable to Another**

sObject variables and sObject collection variables can have values set for fields that the running user can’t edit. However, you can use the other values to create or update records with Fast Create or Fast Update elements. To do so, map the writable values from the original sObject variable into a new sObject variable.

**SEE ALSO:**
Flow Record Create Element
Flow Record Update Element

### Copy Field Values from One sObject Variable to Another

sObject variables and sObject collection variables can have values set for fields that the running user can’t edit. However, you can use the other values to create or update records with Fast Create or Fast Update elements. To do so, map the writable values from the original sObject variable into a new sObject variable.

**Note:** With sObject collection variables, use loops to map the field values to a new collection.

1. Add an Assignment element to your flow. Make sure that the flow executes this element after the original sObject variable has been populated but before the Create or Update element.
2. For each writable field in the original sObject variable, add a row.
   - **Variable**—Select ` {!sObjectName field}`, where `sObjectName` is the name of the new variable and `field` is the field on that variable.
   - **Operator**—Select `equals`.
   - **Value**—Select ` {!sObjectName field}`, where `sObjectName` is the name of the original variable and `field` is the field on that variable.
Note: If you plan to reference the variable in a Fast Update element, include the record’s ID in the new sObject variable. Although Id is read only, the flow uses the value to determine which records to update.

Example: You have a case sObject variable called {!myCaseVar_all}. It stores values for some read-only fields, so you can’t use it in a Fast Update element. You copy the fields that you want to update to a new sObject variable: IsEscalated and Status. You also copy Id, because it’s required for an update operation. Here’s what those assignment rules look like.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{!myCaseVar_final.Id}</td>
<td>equals</td>
<td>{!myCaseVar_original.Id}</td>
</tr>
<tr>
<td>{!myCaseVar_final.IsEscalated}</td>
<td>equals</td>
<td>{!myCaseVar_original.IsEscalated}</td>
</tr>
<tr>
<td>{!myCaseVar_final.Status}</td>
<td>equals</td>
<td>{!myCaseVar_original.Status}</td>
</tr>
</tbody>
</table>

The same example works for an sObject collection variable. However, because you can’t directly change the values of a collection variable, you use a loop.

1. Using a Loop element, the flow passes each item’s values into a loop variable ( {!myCaseLoopVar_original}).
2. For each iteration, an Assignment element copies the Id, IsEscalated, and Status fields from the loop variable to another sObject variable ( {!myCaseLoopVar_final}).
3. The flow then adds the (myCaseLoopVar_final) variable’s values to a new collection. The second Assignment element includes this rule.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{!myCaseColl_updated}</td>
<td>add</td>
<td>{!myCaseLoopVar_final}</td>
</tr>
</tbody>
</table>
After the flow has iterated over every item in the original collection, it exits the loop.

SEE ALSO:
- Which Fields Are Inaccessible When a Flow Creates or Updates Records?
- Control What Happens When a Flow Tries to Set Values for Read-Only Fields
- Flow Assignment Element

Considerations for Managing Flows
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:
- Considerations for Installed Flows
  Keep these considerations in mind when you distribute, upgrade, or remove a flow that you installed from a package.

SEE ALSO:
- Manage Your Flows
- Limits and Considerations for Visual Workflow

Considerations for Installed Flows
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 a managed package, error emails for that flow’s interviews don’t include any details about the individual flow elements. The email is sent to the user who installed the flow.
• 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.

Translating Installed Flows
You can translate flow definition names only on the Translate page.

SEE ALSO:
Flows in Change Sets and Packages
Considerations for Deploying Flows with Packages
Install a Package

Considerations for Running Flows
When you run or test a flow, keep the 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.
• Interviews don’t perform actions—such as sending emails or creating, editing, or deleting records—until the associated transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element. Record and Fast elements aren’t the only ones to create or update records. Post to Chatter, Submit for Approval, and Quick Actions elements do, as well.
• 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.

Lightning Runtime Limitations
When Lightning runtime is enabled for your org, flows in Lightning Experience don’t load in:

• Web tabs
• List buttons that are set to display an existing window with or without a sidebar

When Lightning runtime is enabled for your org, flows in Salesforce Classic don’t load in:

• Web tabs
• Custom buttons or links that are set to display in an existing window with or without a sidebar

Users can’t enter more than 16 digits, including digits before and after a decimal point.

SEE ALSO:
  Test a Flow
  Flow Interviews
  Limits and Considerations for Visual Workflow
  Flow Runtime Experiences

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

• The title of the screen doesn’t change when you click Next or Previous, so you might not realize you’re on a new page.
• Radio button fields don’t have labels. Screen readers can’t distinguish between questions.
• Questions without defined prompts can read incorrectly.
• Errors are not noted when reading the fields.

SEE ALSO:
  Limits and Considerations for Visual Workflow
Create a 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 flow projects in the Automate Your Business Processes trail on Trailhead. They’re a great way to learn about the tool and discover how it works.

1. Open the Cloud Flow Designer. From Setup, enter Flows in the Quick Find box, then select Flows, and then click New Flow.
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
- Considerations for Designing Flows
- Flow Accessibility
- Flow Building Blocks

Flow Building Blocks

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

Cloud Flow Designer

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.

EDITIONS
Available in: both Salesforce Classic and Lightning Experience

Available in: Enterprise, Performance, Unlimited, and Developer Editions
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.
Button Bar (1)

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 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 Flow Elements, Resources, and Connectors
- Search Within the Palette
- Search Within a Flow

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 find an element or resource in the flow by using one or more of the following options.

- On the Explorer tab, enter search text. The Explorer tab displays only the elements and resources whose properties contain the entered text.

To remove the filter, click and select SEARCH ALL.

- Filter the Explorer tab contents to one type of element or resource by clicking .

- Dim all visible elements on the canvas other than the results by selecting Highlight Results on Canvas.

- 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 .

  - 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 . 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 \( \text{🔍} \), 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 \( \text{🔍} \) and select what you want to see.
- To remove the filter, click \( \text{🔍} \) and select SEARCH ALL.

SEE ALSO:
  Tour the Cloud Flow Designer User Interface

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 \( \text{🔍} \).

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.

  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
- Activate or Deactivate a Flow Version

Common Flow Tasks

A handful of tasks are common to multiple flow use cases. For example, you can define conditions in both Decision and Wait elements.

**IN THIS SECTION:**
- Manage Flow Elements, Resources, and Connectors
  Customize your flow by adding, editing, or removing elements, resources, and connectors.
- Working with Salesforce Records in a Flow
  The real power of a flow is that it can automate updates to your organization’s records. In a flow, you can automatically look up values from records, create records, update records, delete records—the whole shebang!
Validate Users’ Inputs with Flow Formulas
Just like with regular validation rules, you can validate what users enter in flow screens.

Define Flow Conditions
Control when a flow takes a specific decision outcome or waits for a specific wait event.

Send Email from a Flow
To send email from your flow, either call an email alert workflow action or create the email in the flow.

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.

View Inputs and Outputs of Other Referenced Flow Versions
While configuring a subflow element, view the variables of a specified version of the referenced flow. Doing so lets you configure draft master and referenced flows at the same time.

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

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.

Define the Path That a Flow Takes
Identify which elements the flow executes and in what order by connecting the elements on your canvas together.

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.

Manage Flow Elements, Resources, and Connectors
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>Double-click, or hover over it and click ✶</td>
<td>Hover over it and click ✶</td>
</tr>
<tr>
<td>Drag from the Palette tab and drop it on to the canvas.</td>
<td>From the Explorer tab, double-click or hover over it and click ✶.</td>
<td>From the Explorer tab, 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>
</tbody>
</table>
Working with Salesforce Records in a Flow

The real power of a flow is that it can automate updates to your organization’s records. In a flow, you can automatically look up values from records, create records, update records, delete records—the whole shebang!

For each of those operations, the Cloud Flow Designer offers at least two elements to choose from. Review the following topics to understand the differences between those elements and decide which one is best for your use case.

Tip: Be familiar with the API names for the objects and fields that you want to work with. The Cloud Flow Designer displays API names instead of labels.

IN THIS SECTION:

Pull Values from Salesforce Records into a Flow
Before you can use information from your Salesforce records in a flow, pull that information into variables in your flow. Use either a Record Lookup element or a Fast Lookup element. The right element depends on what the rest of your flow is doing.

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

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

Delete Salesforce Records from a Flow
To delete Salesforce records, use either the Record Delete or Fast Delete element. The right element depends on what the rest of your flow is doing.
Pull Values from Salesforce Records into a Flow

Before you can use information from your Salesforce records in a flow, pull that information into variables in your flow. Use either a Record Lookup element or a Fast Lookup element. The right element depends on what the rest of your flow is doing.

Alternatively, pass values in from an element that interacts with the Salesforce database—such as the ID of the post created by a Post to Chatter element.

**Example:** You need to email a given account’s owner. To do so, the flow needs to know the email address and name of that user.

To pull values into a flow from records in your organization, use either the Record Lookup or Fast Lookup element in the Cloud Flow Designer.

How do I choose between flow lookup elements?

The two flow lookup elements are pretty similar. This table summarizes the two main differences between them.

<table>
<thead>
<tr>
<th>Can store values in ...</th>
<th>To map field values to flow variables ...</th>
<th>Number of records it looks up</th>
</tr>
</thead>
</table>
| **Record Lookup**       | 1. Identify each field that you want to store.  
2. For each field, identify a flow variable to store that specific value in.  
Because you directly map each field value to a variable, you get more granularity with this element. However, with more granularity comes more clicking. | Exactly one. |
| • Variables              |                                         |                               |
| • sObject variables     |                                         |                               |
| **Fast Lookup**          | 1. Identify the flow variable in which you want to store all field values.  
2. Identify the fields whose values you want to store in that flow variable. | If an sObject variable: one.  
If an sObject collection variable: at least one. |
| • sObject variables     |                                         |                               |
| • sObject collection variables |                                      |                               |

Unless you want to map each field to a variable with fewer mouse clicks, it can be hard to choose between the two elements. To choose the right lookup element, figure out what type of variable you need to store the values in.

- To store the values in a single-value non-sObject variable, use the Record Lookup element.
- To store the values in an sObject collection variable, use the Fast Lookup element.
- To store the values in a single-value sObject variable, it’s your choice. (Fast Lookup might save you some clicks!)

**Tip:** It’s best practice to use Fast elements whenever possible, so that you save your org’s limits. For more information, see Flow Bulkification in Transactions.
Example: Here’s how you’d store a user’s email and name by using each of the lookup elements.

<table>
<thead>
<tr>
<th>Record Lookup</th>
<th>Field</th>
<th>Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>![UserId]</td>
<td>![UserEmail]</td>
</tr>
<tr>
<td>Email</td>
<td>![UserEmail]</td>
<td>![UserEmail]</td>
</tr>
<tr>
<td>FirstName</td>
<td>![UserFirstName]</td>
<td>![UserFirstName]</td>
</tr>
<tr>
<td>LastName</td>
<td>![UserLastName]</td>
<td>![UserLastName]</td>
</tr>
</tbody>
</table>

| Fast Lookup | Variable ![UserFields] | Assign null to the variable if no r |

SEE ALSO:
- Flow Fast Lookup Element
- Flow Record Lookup Element
- Working with Salesforce Records in a Flow

Create Salesforce Records from a Flow

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

Example: When the customer’s satisfaction score drops below a certain number, automatically create a case.

To create one or more Salesforce records, your flow:
1. Identifies the field values for the new records.
2. Saves those changes to the Salesforce database. (In other words, until the changes are saved to the database, the changes exist only within the flow.)

How do I choose between flow elements that create records?

The main difference between create elements lies in how many records the element can create and how it knows the field values to apply.
I need to create more than one record at a time.
To create more than one record at a time, use a Fast Create element with an sObject collection variable. It’s best practice to use Fast elements whenever possible, so that you stay within your org’s limits. For more information, see Flow Bulkification in Transactions. Record Create and Quick Action elements can create only one record at a time. Fast Create elements can create either one record (if using an sObject variable) or multiple records (if using an sObject collection variable).

I need to create exactly one record.
If you’ve already populated an sObject variable with the values you want your record to have, use a Fast Create element. If you want to use a combination of the values from an sObject variable and values from other resources (like single-value variables or screen input fields), use either a Record Create or Quick Action element. Those two elements differ in these ways.

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

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

<table>
<thead>
<tr>
<th>Field Availability</th>
<th>Required Fields</th>
<th>New Record ID</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record Create</strong></td>
<td>Every field on the object. You manually select the object and every field you want to have a value.</td>
<td>Not indicated</td>
</tr>
<tr>
<td><strong>Quick Action</strong> (&lt;i&gt;of type Create&lt;/i&gt;)</td>
<td>Only fields that are included in the Quick Action layout. If you supplied default values for certain fields when you created the quick action, those values are used when the record is created.</td>
<td>Indicated Requiredness is based on what’s marked required in the quick action layout.</td>
</tr>
</tbody>
</table>

Tip: Use the Quick Actions element when all these statements are true.
1. The action is of type Create.
2. The action’s layout includes all the fields that you want to update.
3. You don’t need to reference the new record’s ID later in the flow.
Otherwise, use the Record Create element.
**Example:** Here’s how you’d create a case when a customer’s satisfaction score is too low by using each of the create elements.

<table>
<thead>
<tr>
<th>Record Create</th>
<th>Fast Create</th>
<th>Quick Action (of type Create)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ContactId</strong></td>
<td><strong>Variable</strong></td>
<td><strong>Contact ID</strong></td>
</tr>
<tr>
<td><strong>Description</strong></td>
<td><strong>(IsvarCase)</strong></td>
<td><strong>Subject</strong></td>
</tr>
<tr>
<td><strong>Status</strong></td>
<td><strong>(IsvarContactId)</strong></td>
<td><strong>Status</strong></td>
</tr>
<tr>
<td><strong>Subject</strong></td>
<td></td>
<td><strong>Description</strong></td>
</tr>
<tr>
<td><strong>AccountId</strong></td>
<td></td>
<td><strong>(textCaseDescription)</strong></td>
</tr>
</tbody>
</table>

You can set any field on the record, but the Record Create element doesn’t know which fields are required for this object.

Assumes (IsvarCase) is already populated with the right fields.

These four fields are the only fields that you can set for this element, because they’re the only ones available from the action layout. Contact ID is required by the associated action layout, so it’s required in this element.

**SEE ALSO:**
- Flow Fast Create Element
- Working with Salesforce Records in a Flow
- Flow Quick Action Element
- Flow Record Create Element

**Update Salesforce Records from a Flow**

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

**Example:** On an opportunity record, when a user clicks the “Won” button, a flow updates the opportunity’s stage.

To update fields on one or more existing Salesforce records, your flow:

1. Identifies the records to update.
2. Identifies the new field values for those records.
3. Saves those changes to the Salesforce database. (In other words, until the changes are saved to the database, the changes exist only within the flow.)

**How do I choose between flow elements that update records?**

The main difference between the elements lies in these areas: how it knows which records to update, how it knows the new field values to apply, and how many records it can update.

Quick Action elements can update only one record at a time, while Record Update and Fast Update elements can update multiple records.

<table>
<thead>
<tr>
<th></th>
<th>To identify records to update</th>
<th>To identify new field values for the records</th>
<th>Number of records it updates</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record Update</strong></td>
<td>In the same element, use filter criteria.</td>
<td>In the same element, map each field that should be updated with a variable or other resource.</td>
<td>At least one.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All resources are supported, so long as the resource’s data type matches the selected field’s data type.</td>
<td></td>
</tr>
<tr>
<td><strong>Quick Action</strong></td>
<td>Populate a single-value variable with the ID in another element. Use this ID for the Related Record ID parameter.</td>
<td>In the same element, map each field that should be updated with a variable or other resource.</td>
<td>Exactly one.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All resources are supported, so long as the resource’s data type matches the selected field’s data type.</td>
<td></td>
</tr>
<tr>
<td><strong>Fast Update</strong></td>
<td>Populate an sObject variable or sObject collection variable in another element</td>
<td>In another element, such as an Assignment element, update the values in the sObject variable or sObject collection variable.</td>
<td>If an sObject variable: one.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>If an sObject collection variable: at least one.</td>
<td></td>
</tr>
</tbody>
</table>

If the following statement is true, use a Fast Update element:

- You’ve already populated an sObject variable or sObject collection variable with the values you want:

  🔄 **Tip:**
  - You can always update the field values in an sObject variable or sObject collection variable by using an Assignment element.
  - It’s best practice to use Fast elements whenever possible, so that you save your org’s limits. For more information, see Flow Bulkification in Transactions.

If all the following statements are true, use a Quick Action element:

- You need to update exactly one record
- You’ve already populated a variable with the record’s ID
- The Quick Action’s layout includes all the fields you need to update

If any of those statements aren’t true, use a Record Update element.
**Example:** Here’s how you’d update an opportunity’s stage by using each of the update elements.

<table>
<thead>
<tr>
<th>Record Update</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Update</strong></td>
<td>Opportunity ▼</td>
</tr>
<tr>
<td>Field</td>
<td>Operator ▼</td>
</tr>
<tr>
<td>Id</td>
<td>equals ▼</td>
</tr>
<tr>
<td><strong>Value</strong></td>
<td>(!varOpportunityId) ▼</td>
</tr>
<tr>
<td>Add Row</td>
<td></td>
</tr>
</tbody>
</table>

**Update record fields with variable, constant, input, or other values.**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>StageName</td>
<td>Closed Lost</td>
</tr>
<tr>
<td>CloseDate</td>
<td>(!$Flow.CurrentDate)</td>
</tr>
</tbody>
</table>

You can update any field on the record, but the Record Update element doesn’t know which fields are required for this object.

<table>
<thead>
<tr>
<th>Fast Update</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Variable</strong></td>
<td>(!varOpportunity) ▼</td>
</tr>
</tbody>
</table>

Assumes (!varOpportunity) is already populated with the right fields.

<table>
<thead>
<tr>
<th>Quick Action (of type Update)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Close Date ▼</td>
<td>(!$Flow.CurrentDate)</td>
</tr>
<tr>
<td>Related Record ID ▼</td>
<td>(!varOpportunityId)</td>
</tr>
<tr>
<td>Stage ▼</td>
<td>Closed - Lost</td>
</tr>
</tbody>
</table>

These three fields are required by the associated action layout, so they’re required in this element. Related Record ID identifies which opportunity to update.

**SEE ALSO:**
- Flow Fast Update Element
- Flow Record Update Element
- Flow Quick Action Element
- Working with Salesforce Records in a Flow
Delete Salesforce Records from a Flow

To delete Salesforce records, use either the Record Delete or Fast Delete element. The right element depends on what the rest of your flow is doing.

**Example:** When a customer accepts a quote, automatically delete the remaining quotes from the opportunity.

To delete one or more records, your flow:

1. Identifies the records that to delete.
2. Saves those changes to the Salesforce database. (In other words, until the changes are saved to the database, the changes exist only within the flow.)

How do I choose between flow elements that delete records?

The main difference between elements lies in how the element knows which records to delete.

<table>
<thead>
<tr>
<th>To identify records to delete</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record Delete</strong></td>
<td>In the same element, use filter criteria.</td>
</tr>
<tr>
<td><strong>Fast Delete</strong></td>
<td>In another element, populate an sObject variable or sObject collection variable with the ID of the record to be deleted.</td>
</tr>
</tbody>
</table>

If you’ve already populated an sObject variable or sObject collection variable with the records you want to delete, use a Fast Delete. (sObject collection variables are supported for record deletion only with a Fast Create element.) It’s best practice to use Fast elements whenever possible, so that you save your org’s limits. For more information, see Flow Bulkification in Transactions.

If you haven’t yet identified which records to delete or you’ve stored the IDs in non-sObject resources—such as a single-value variable—use a Record Delete element. sObject collection variables aren’t supported for this element.

**Example:** Here’s how you’d delete remaining quotes from an opportunity by using each of the delete elements.

<table>
<thead>
<tr>
<th>Record Delete</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete = Quote that meet the following criteria:</td>
<td></td>
</tr>
<tr>
<td>Field</td>
<td>Operator</td>
</tr>
<tr>
<td>OpportunityId</td>
<td>equals</td>
</tr>
<tr>
<td>Status</td>
<td>does not equal</td>
</tr>
</tbody>
</table>

The flow finds all quotes that are associated with a specific opportunity and haven’t been approved, and then deletes them.
Assumes {!svarQuotesUnnecessary} is already populated with the IDs of the quotes to delete. The flow deletes all records whose IDs are included in that variable.

SEE ALSO:
- Flow Fast Delete Element
- Flow Record Delete Element
- Working with Salesforce Records in a Flow

### Validate Users’ Inputs with Flow Formulas

Just like with regular validation rules, you can validate what users enter in flow screens.

- The formula expression must return a Boolean value (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, the flow doesn’t validate the field.

When you configure a screen input field:

1. In the Input Validation section, select Validate.
2. Define the values allowed for the field by entering a Boolean formula expression.

**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.
- 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:
  ```plaintext
  REGEX({!Email_Address},"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}")
  ```
- Validate the format of a zip code:
  ```plaintext
  REGEX({!Zipcode},"\d{5}(-\d{4})?"")
  ```

**EDITIONS**

Available in: both Salesforce Classic and Lightning Experience

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

**USER PERMISSIONS**

To open, edit, or create a flow in the Cloud Flow Designer:

- Manage Force.com Flow
Define Flow Conditions

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.**

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

<table>
<thead>
<tr>
<th>Column Header</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Flow resource whose value you want to evaluate.</td>
</tr>
<tr>
<td>Operator</td>
<td>The available operators depend on the data type selected for Resource. For details, see Operators in Flow Conditions on page 118.</td>
</tr>
<tr>
<td>Value</td>
<td>The Variable and Value in the same row must have compatible data types. 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>
<tr>
<td></td>
<td><strong>Note:</strong> When you add or subtract a number from a date value, the date adjusts in days, not hours.</td>
</tr>
</tbody>
</table>

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>
<tr>
<td>Advanced logic (Combination of ANDs and ORs)</td>
<td>Custom logic. When you select this option, provide the customized Logic by entering a text string. Use:</td>
</tr>
<tr>
<td></td>
<td>• Numbers to refer to each condition</td>
</tr>
<tr>
<td></td>
<td>• AND or OR to identify whether all or just one of the conditions must true</td>
</tr>
<tr>
<td></td>
<td>• Parentheses to group parts of the string together</td>
</tr>
<tr>
<td><strong>Tip:</strong> If you enter <strong>AND</strong>, it’s the same as if you selected <strong>All conditions must be true (AND)</strong>. If you enter <strong>OR</strong>, it’s the same as if you selected <strong>One condition must be true (OR)</strong>. If you enter any other logic, make sure that you include a number for each condition.</td>
<td></td>
</tr>
</tbody>
</table>
Option | Description
--- | ---
 | 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.

SEE ALSO:
- What Are Waiting Conditions?
- Flow Wait Element
- Flow Decision Element

Send Email from a Flow

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

**Email Alert element**

Sends an email by using a workflow email alert to specify the email template and recipients. The flow provides only the record ID.

**Send Email element**

Sends an email by manually specifying the subject, body, and recipients in the flow.

SEE ALSO:
- Flow Elements

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

<table>
<thead>
<tr>
<th></th>
<th>Process.Plugin Interface</th>
<th>@InvocableMethod Annotation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk operations</td>
<td>Not supported</td>
<td>Supported</td>
</tr>
<tr>
<td>Element name</td>
<td>Class name or the value of the <code>name</code> property.</td>
<td>Class name</td>
</tr>
</tbody>
</table>
| Reusability          | Classes with this interface implemented are available in flows | Classes with this annotation implemented are available in:  
  - Flows  
  - Processes  
  - Rest API |
| Section in the Cloud Flow Designer | Apex Plug-in or the value of the `tag` property. | Apex |

**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
        List<Process.PluginDescribeResult.OutputParameter>{
            new Process.PluginDescribeResult.OutputParameter('accountId',
                Process.PluginDescribeResult.ParameterType.STRING)
        };
    return result;
}

Notice that lookupAccountAnnotation is less than half the length (11 lines) of lookupAccountPlugin (28 lines). In addition, because the annotation supports bulk operations, lookupAccountAnnotation performs one query per batch of interviews. lookupAccountPlugin performs one query per interview.

SEE ALSO:
Flow Elements

View Inputs and Outputs of Other Referenced Flow Versions

While configuring a subflow element, view the variables of a specified version of the referenced flow. Doing so lets you configure draft master and referenced flows at the same time.

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.
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:
Flow Subflow Element

Clone Records with a Fast Create Element

A flow can clone records in your org. First, populate an sObject variable with an existing record’s values. Identify fields that the running user can’t edit, and map all remaining fields to another sObject variable. Then use the second sObject variable in a Fast Create element to clone the record. Before you begin, review Which Fields Are Inaccessible When a Flow Creates or Updates Records?

1. Populate an sObject variable with the values from the existing record.
   For example:
   - Look up the record with a Fast Lookup element.
   - Obtain the record from a Flows action in a process.

2. In an Assignment element, copy the writable field values to a new sObject variable.
   
   \* Note: Make sure that Id isn’t set in the new variable.

3. Add a Fast Create element to your flow. Select the new sObject variable to populate the values of the new clone record.

SEE ALSO:
Copy Field Values from One sObject Variable to Another
Flow Fast Create Element

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

<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:

Flow Collection Variable Resource  
Sample Flow That Populates a Collection Variable
Define the Path That a Flow Takes

Identify which elements the flow executes and in what order by connecting the elements on your canvas together.

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.

IN THIS SECTION:

Remove Connectors from a Flow
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.

SEE ALSO:

Remove Connectors from a Flow
Flow Connectors
Customize What Happens When a Flow Fails

Remove Connectors from a Flow

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. If you’re having trouble selecting a connector, click and drag an area on the canvas that includes the connector.

2. Press DELETE.

SEE ALSO:

Define the Path That a Flow Takes
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 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 it if your flow is distributed through a process.

Other Examples of Error Handling in Flows
Examples of using fault connectors to handle flow errors include requesting corrections from the user and bypassing the error.

SEE ALSO:
Flow Connectors
Flow Elements
Define the Path That a Flow Takes

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 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.

   ```text
   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.

SEE ALSO:
- Flow Text Template Resource
- Flow Send Email Element
- Customize What Happens When a Flow Fails

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

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

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

```html
<FONT FACE="Arial" STYLE="font-size:14px"> 
<B>Something went wrong with this flow.</B>
```
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:
- Flow Screen Element: Display Text Fields
- Flow Text Template Resource
- 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 ({{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
Flow Reference

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

IN THIS SECTION:

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.

Flow Resources
Each resource represents a value that you can reference throughout the flow.

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.

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.

Flow Operators
Operators behave differently, depending on what you’re configuring. In Assignment elements, operators let you change resource values. In flow conditions and record filters, operators let you evaluate information and narrow the scope of a flow operation.

Flow Event Types
Event Type drives the fields that you use to define an event in a flow Wait element. You can use a platform event, which you can fully customize. You can also use an alarm consisting of a date/time value—the base time—and an optional offset from that time.

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.

Flow Properties
A flow’s properties consist of its name, description, interview label, and type. These properties drive the field values that appear on a flow or flow version’s detail page. The properties of a flow and its flow versions are separate.

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.

IN THIS SECTION:

General Settings for Flow Elements
Every flow element has three settings in common: name, unique name, and description.
Flow Activate Session-Based Permission Set Element
Activates a session-based permission set for the running user by manually specifying its name.

Flow Deactivate Session-Based Permission Set Element
Deactivates a session-based permission set for the running user by manually specifying its name.

Flow Apex Plug-In Element
Calls an Apex class that implements the Process.Plugin interface. If you used the Tag property in the PluginDescribeResult class, the Apex class appears under a customized section. Otherwise, it appears under the Apex Plug-ins section.

Flow Assignment Element
Sets or changes values in variables, collection variables, sObject variables, and sObject collection variables.

Flow Call Apex Element
Calls an Apex class's invocable method.

Flow Create Service Report Element
Creates a service report for a service appointment, work order, or work order line item.

Flow Decision Element
Evaluates a set of conditions and routes users through the flow based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

Flow Email Alert Element
Sends an email by using a workflow email alert to specify the email template and recipients. The flow provides only the record ID.

Flow Fast Create Element
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.

Flow Fast Delete Element
Deletes Salesforce records using the ID values that are stored in an sObject collection variable. Or deletes one Salesforce record using the ID value that's stored in an sObject variable.

Flow Fast Lookup Element
Finds Salesforce records to assign their field values to an sObject collection variable. Or finds one Salesforce record to assign its field values to an sObject variable.

Flow Fast Update Element
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. If a record's ID is included in the variable, its field values are updated to match the other values that are stored in the variable.

Flow Generate Work Orders Element
Generates work orders from a maintenance plan.

Flow Loop Element
Iterates through a collection one item at a time, and executes actions on each item's field values—using other elements within the loop.

Flow Record Create Element
Creates one Salesforce record by using individual field values that you specify.

Flow Record Delete Element
Deletes all Salesforce records that meet specified criteria.
### Flow Record Lookup Element
Finds the first Salesforce record that meets specified criteria. Then assigns the record's field values to individual flow variables or individual fields on sObject variables.

### Flow Record Update Element
Finds all Salesforce records that meet specified criteria and updates them with individual field values that you specify.

### Flow Quick Action Element
Calls an object-specific or global quick action that's already been configured in your organization. Only "Create," "Update," and "Log a Call" actions are supported.

### Flow Post to Chatter Element
Posts a message to a specified feed, such as to a Chatter group or a case record. The message can contain mentions and topics, but only text posts are supported.

### Flow Screen Element
Displays a screen to the user who is running the flow, which lets you display information to the user or collect information from the user.

### Flow Send Email Element
Sends an email by manually specifying the subject, body, and recipients in the flow.

### Flow Step Element
Acts as a placeholder when you're not sure which element you need.

### Flow Submit for Approval Element
Submits one Salesforce record for approval.

### Flow Subflow Element
Calls another flow in your organization. Use this element to reference modular flows and simplify the overall architecture of your flow.

### Flow Wait Element
WAits for one or more defined events to occur, which lets you automate processes that require a waiting period.

SEE ALSO:
- Flow Resources
- Cloud Flow Designer

### General Settings for Flow Elements
Every flow element has three settings in common: name, unique name, and 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.</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>
### Flow Activate Session-Based Permission Set Element

Activates a session-based permission set for the running user by manually specifying its name.

**Important:** You can run queries, however, do not make data or object updates in flows that also activate session-based permission sets.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>Appears after you click <strong>Add Description</strong>.</td>
</tr>
<tr>
<td>Permission Set Name</td>
<td>The developer name of the permission set. This parameter accepts <strong>single-value resources of any type</strong>. That value is treated as text.</td>
</tr>
<tr>
<td>Permission Set Namespace</td>
<td>Optional. The permission set’s namespace. This parameter accepts <strong>single-value resources of any type</strong>. That value is treated as text.</td>
</tr>
</tbody>
</table>

**Example:** A junior buyer in your org occasionally requires access to your Contracts object. Create a session-based permission set with access to the object. Then, create a flow that uses the Activate Session-Based Permission Set action. Configure the action to activate the permission set. The junior buyer runs the flow to access contracts during the current user session. The action activates the permission set for the junior buyer during the current session.

**SEE ALSO:**
- [Cloud Flow Designer](#)
- [Flow Deactivate Session-Based Permission Set Element](#)
- [Create a Flow That Can Activate or Deactivate a Session-Based Permission Set](#)

### Flow Deactivate Session-Based Permission Set Element

Deactivates a session-based permission set for the running user by manually specifying its name.

**Editions**

Available in: both Salesforce Classic and Lightning Experience

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

Deactivate a running session-based permission set.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Permission Set Name</td>
<td>The developer name of the permission set. This parameter accepts <strong>single-value resources of any type</strong>. That value is treated as text.</td>
</tr>
</tbody>
</table>

**Editions**

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions
**Permission Set Namespace**

- **Optional. The permission set’s namespace.**
- This parameter accepts **single-value resources of any type.** That value is treated as text.

**SEE ALSO:**

- Cloud Flow Designer
- Flow Activate Session-Based Permission Set Element
- Create a Flow That Can Activate or Deactivate a Session-Based Permission Set

### Flow Apex Plug-In Element

Calls an Apex class that implements the `Process.Plugin` interface. If you used the Tag property in the `PluginDescribeResult` class, the Apex class appears under a customized section. Otherwise, it appears under the Apex Plug-ins section.

**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, ask your developer to 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 48.

#### Inputs

Pass information from the flow to the invoked Apex method. The method determines the available input parameters and their data types.

#### Outputs

Pass information from the invoked Apex method to the flow. The method determines the available output parameters and their data types.

The flow assigns the values to the specified variables when the code is executed.
Usage

Note: If the Apex class creates, updates, or deletes a record, the action isn’t performed until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:
- Invoke Apex Code from a Flow
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows
- Apex Developer Guide: Process Namespace

Flow Assignment Element

Sets or changes values in variables, collection variables, sObject variables, and sObject collection variables.

In each row, specify the Variable whose value you want to change and the new Value for that variable. At run time, the variable assignments occur in the order you specify.

<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.</td>
</tr>
<tr>
<td>Value</td>
<td>The Variable and Value in the same row must have compatible data types.</td>
</tr>
<tr>
<td></td>
<td>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>

Example: Change the value of a customer’s credit score based on how the customer answered questions in the flow.

SEE ALSO:
- Flow Elements
- Operators in Flow Assignment Elements
- Define the Path That a Flow Takes
- Flow Resources
- Cross-Object Field References in Flows
Flow Call Apex Element

Calls an Apex class’s 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.

Inputs

Pass information from the flow to the invoked Apex method. The method determines the available input parameters and their data types.

Outputs

Pass information from the invoked Apex method to the flow. The method determines the available output parameters and their data types.

The flow assigns the values to the specified variables when the method is executed.

Usage

Note: If the invoked method creates, updates, or deletes a record, that action isn’t performed until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:

- Invoke Apex Code from a Flow
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows

Flow Create Service Report Element

Creates a service report for a service appointment, work order, or work order line item.

Inputs

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>Required. The ID of the service appointment, work order, or work order line item that the service report is created for.</td>
</tr>
<tr>
<td>Template ID</td>
<td>Required. The ID of the standard or custom service report template that is used to create the service report.</td>
</tr>
</tbody>
</table>

Usage

Field Service Lightning must be enabled in your org, and you must have the Field Service Standard user permission.
Flow Decision Element

Evaluates a set of conditions and routes users through the flow based on the outcomes of those conditions. This element performs the equivalent of an if-then statement.

Outcomes

Create the outcomes for the decision. To rename the path that the flow takes when none of the other outcome conditions are met, click [Default Outcome]. Set up the conditions.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Identifies the connector for this outcome on the canvas.</td>
</tr>
<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>Conditions</td>
<td>Determines whether the flow takes this outcome’s path.</td>
</tr>
</tbody>
</table>

Example: Using a Decision element, 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 so that it does 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.

SEE ALSO:
- Flow Elements
- Define Flow Conditions
- Operators in Flow Conditions
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows

Flow Email Alert Element

Sends an email by using a workflow email alert to specify the email template and recipients. The flow provides only 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>Select a variable that contains the ID for the record that you want the email to reference. If the email alert uses any merge fields, this record is the starting point for those merge fields. This field accepts single-value variables of any type. The value is treated as text.</td>
</tr>
</tbody>
</table>

**Usage**

⚠️ **Note:** At run time, the email isn’t sent until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

**SEE ALSO:**
- Send Email from a Flow
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows

**Flow Fast Create Element**

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.

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>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. This field accepts any sObject variable or sObject collection variable.</td>
</tr>
</tbody>
</table>

⚠️ **Example:** 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.
Usage

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: At run time, records aren’t created until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:
Create Salesforce Records from a Flow
Clone Records with a Fast Create Element
Flow Elements
Customize What Happens When a Flow Fails
Define the Path That a Flow Takes
Flow sObject Variable Resource
Flow sObject Collection Variable Resource

Flow Fast Delete Element

Deletes Salesforce records using the ID values that are stored in an sObject collection variable. Or deletes one Salesforce record using the ID value that’s stored in an sObject variable.

Tip: Make sure that the sObject variable or collection is populated with ID values before using the Fast Delete element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Identifies the sObject variable or collection that you want to use to delete records. The variable must include the IDs of the records that you want to delete. This field accepts any sObject variable or sObject collection variable.</td>
</tr>
</tbody>
</table>

Usage

Warning:

- Be careful when testing flows that contain delete elements. Even if the flow is inactive, it triggers the delete operation.
- To prevent deleting records by mistake, be as specific in your filter criteria as possible.
- Records are deleted from your 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 pending approval.

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.
**Note:** At run time, the records aren’t deleted until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

**SEE ALSO:**
- Delete Salesforce Records from a Flow
- Flow Elements
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Flow sObject Variable Resource
- Flow sObject Collection Variable Resource

## Flow Fast Lookup Element

Finds Salesforce records to assign their field values to an sObject collection variable. Or finds one Salesforce record to assign its field values to an sObject variable.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up</td>
<td>Identifies the object whose records you want to look up.</td>
</tr>
<tr>
<td>Filter criteria</td>
<td>Narrows down the scope of records that the flow looks up. Specify the filter criteria for selecting the record from the database.</td>
</tr>
<tr>
<td>Tip:</td>
<td>Make sure that your filter criteria sufficiently narrows the search. If you use an sObject variable to store the results, the Fast Lookup element returns only the first record from the filtered results. Value must be compatible with the selected field’s data type.</td>
</tr>
<tr>
<td>Sort results by:</td>
<td>Sorts the filtered results before storing records in the variable. If selected, also select the field that you want to sort the results by and the sort order. Only sortable fields are available.</td>
</tr>
<tr>
<td>Variable</td>
<td>The variable to contain the returned field values. This field accepts any sObject variable or sObject collection variable.</td>
</tr>
<tr>
<td></td>
<td>- To contain the field values for the first returned record, use an sObject variable.</td>
</tr>
<tr>
<td></td>
<td>- To contain the field values for all returned records, use an sObject collection variable.</td>
</tr>
<tr>
<td>Assign null to the variable if no records are found.</td>
<td>Sets the variable to null if no records meet the filter criteria. By default, the variable’s values are left unchanged.</td>
</tr>
</tbody>
</table>
**Field** | **Description**
--- | ---
Specify which of the record’s fields to save in the variable. | Identifies which fields on the records that meet the filter criteria to store in the variable. Values for unselected fields are set to null in the variable.

**Example:**
- Look up a product’s name and description by using the bar code on its product tag.
- Look up all customers who live in a particular city.
- Look up customer transactions on a particular day.

**Considerations for Defining Filter Criteria**
- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR. For example, your filters check whether a case’s Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At runtime, those filters are combined to \((1 \text{ OR } 2) \text{ AND } 3\).
- The available filter operators depend on the data type of the selected fields. For details, see Operators in Flow Record Filters.

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

SEE ALSO:
- Pull Values from Salesforce Records into a Flow
- Flow Elements
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Flow sObject Variable Resource
- Flow sObject Collection Variable Resource
- Cross-Object Field References in Flows
Flow Fast Update Element

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. If a record's ID is included in the variable, its field values are updated to match the other values that are stored in the variable.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable</td>
<td>Identifies the sObject variable or collection that you want to use to update records. This field accepts any sObject variable or sObject collection variable.</td>
</tr>
</tbody>
</table>

**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.

**Usage**

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.

**Note:** At run time, records aren’t updated until the interview's transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

**SEE ALSO:**
- Update Salesforce Records from a Flow
- Flow Elements
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Flow sObject Variable Resource
- Flow sObject Collection Variable Resource

Flow Generate Work Orders Element

Generates work orders from a maintenance plan.

**Inputs**

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>The ID of the maintenance plan from which you want to generate work orders.</td>
</tr>
</tbody>
</table>
Usage

Field Service Lightning must be enabled in your org, and you must have the Field Service Standard user permission. In addition, to generate work orders, you need “Read” access to assets and “Create” access to work orders and service appointments.

You can’t generate more than 2,500 work orders at a time. You can decrease the number of work orders generated by increasing the frequency, decreasing the generation timeframe, or decreasing the number of assets related to the maintenance plan.

Flow Loop Element

Iterates through a collection one item at a time, and executes actions on each item’s field values—using other elements within the loop.

A collection is a list of items that contain, for example, field values from Salesforce records. A loop uses an sObject variable, referred to as the 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Loop through</td>
<td>The collection that you want to loop through.</td>
</tr>
<tr>
<td></td>
<td>This field accepts any sObject variable or sObject collection variable.</td>
</tr>
<tr>
<td>Order</td>
<td>Ascending begins at the start of the collection and moves to the end, while</td>
</tr>
<tr>
<td></td>
<td>Descending begins at the end and moves to the start.</td>
</tr>
<tr>
<td>Loop Variable</td>
<td>The variable that the flow uses to contain the current item’s values during</td>
</tr>
<tr>
<td></td>
<td>a loop iteration.</td>
</tr>
<tr>
<td></td>
<td>• If Loop through is set to a non-sObject collection variable, this field</td>
</tr>
<tr>
<td></td>
<td>accepts a single-value variable with the same data type.</td>
</tr>
<tr>
<td></td>
<td>• If Loop through is set to an sObject collection variable, this field</td>
</tr>
<tr>
<td></td>
<td>accepts an sObject variable with the same object type.</td>
</tr>
</tbody>
</table>

Usage

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

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

SEE ALSO:

Sample Flow That Loops Through a Collection
Define the Path That a Flow Takes
Flow Elements
Flow sObject Collection Variable Resource
Flow Record Create Element

Creates one Salesforce record by using individual field values that you specify.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>The object for which you want to create a record.</td>
</tr>
<tr>
<td>Field values</td>
<td>Identifies the field values for the new record. Each value must be compatible with the selected field's data type.</td>
</tr>
<tr>
<td></td>
<td><strong>Important:</strong> Ensure that all required fields are populated with values; otherwise the flow fails at run time. If you don't know which fields are required, check the object definition.</td>
</tr>
<tr>
<td>Variable</td>
<td>Assigns the ID of the new record to a variable so you can reference it later in the flow.</td>
</tr>
<tr>
<td></td>
<td>This field accepts only <strong>single-value variables of type Text</strong>.</td>
</tr>
</tbody>
</table>

**Example:** A user enters a 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.

**Usage**

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.

**Note:** At run time, the record isn't created until the interview's transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:

Create Salesforce Records from a Flow
Operators in Flow Record Filters
Customize What Happens When a Flow Fails
Define the Path That a Flow Takes
Flow Elements

Flow Record Delete Element

Deletes all Salesforce records that meet specified criteria.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delete</td>
<td>Identifies the object whose records you want to delete.</td>
</tr>
<tr>
<td>Filter Criteria</td>
<td>Narrows down the scope of records that the flow deletes.</td>
</tr>
</tbody>
</table>
Considerations for Defining Filter Criteria

• When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

  For example, your filters check whether a case’s Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At runtime, those filters are combined to \((1 \text{ OR } 2) \text{ AND } 3\).

• The available filter operators depend on the data type of the selected fields. For details, see Operators in Flow Record Filters.

Usage

⚠️ Warning:

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

• To prevent deleting records by mistake, be as specific in your filter criteria as possible.

• Records are deleted from your 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 pending approval.

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.

⚠️ Note: At run time, the record isn’t deleted until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:

Delete Salesforce Records from a Flow
Operators in Flow Record Filters
Customize What Happens When a Flow Fails
Define the Path That a Flow Takes
Flow Elements

Flow Record Lookup Element

Finds the first Salesforce record that meets specified criteria. Then assigns the record’s field values to individual flow variables or individual fields on sObject variables.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Look up</td>
<td>Identifies the object whose records you want to look up.</td>
</tr>
<tr>
<td>Filter criteria</td>
<td>Narrows down the scope of records that the flow looks up. Specify the filter criteria for selecting the record from the database. Value must be compatible with the selected field’s data type.</td>
</tr>
</tbody>
</table>

⚠️ Tip: Make sure that your filter criteria sufficiently narrows the search. The Record Lookup element returns only the first record from the filtered results.
### Field | Description
--- | ---
**Sort results by:** | Sorts the filtered results before storing records in the variable. If selected, also select the field that you want to sort the results by and the sort order. Only sortable fields are available.

**Assign the record’s fields to variables to reference them in your flow.** | Select fields from the returned record, and assign the values to variables in the flow. The values must be compatible with each selected field.

**Assign null to the variable(s) if no records are found.** | Sets the variables to null if no records meet the filter criteria. By default, the variable’s values are left unchanged.

---

**Example:** 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.

### Considerations for Defining Filter Criteria

- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.

  For example, your filters check whether a case’s Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At runtime, those filters are combined to \( (1 \text{ OR } 2) \text{ AND } 3 \).

- The available filter operators depend on the data type of the selected fields. For details, see Operators in Flow Record Filters.

### Usage

Use a **Fast Lookup** element to find:

- A single record and store specified field values in an sObject variable
- Multiple records and store specified field values in an sObject collection

### SEE ALSO:

- [Pull Values from Salesforce Records into a Flow](#)
- [Operators in Flow Record Filters](#)
- [Flow Elements](#)
- [Customize What Happens When a Flow Fails](#)
- [Define the Path That a Flow Takes](#)
- [Cross-Object Field References in Flows](#)
**Flow Record Update Element**

Finds all Salesforce records that meet specified criteria and updates them with individual field values that you specify.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update</td>
<td>Identifies the object whose records you want to update.</td>
</tr>
<tr>
<td>Filter criteria</td>
<td>Narrows down the scope of records that the flow updates.</td>
</tr>
<tr>
<td>Important:</td>
<td>Configure at least one filter, or the flow updates <strong>all</strong> the records for the object. Value must be compatible with the selected field’s data type.</td>
</tr>
<tr>
<td>Update record fields ...</td>
<td>Identifies which fields to update on the records that meet the filter criteria, as well as the new values. The values must be compatible with each selected field.</td>
</tr>
</tbody>
</table>

**Example:** Automatically update Salesforce with data collected from customers, such as new addresses or product preferences.

**Considerations for Defining Filter Criteria**

- When you define multiple filters, the filter logic usually defaults to AND. However, if multiple filters have the same field selected and use the equals operator, the filters are combined with OR.
  
  For example, your filters check whether a case’s Type equals Problem (1), Type equals Feature Request (2), and Escalated equals true (3). At runtime, those filters are combined to (1 OR 2) AND 3.

- The available filter operators depend on the data type of the selected fields. For details, see Operators in Flow Record Filters.

**Usage**

Use Fast Update to:

- Update a single record with all field values from an sObject variable
- Update multiple records with all field values from an sObject collection

**Note:** At run time, the record isn’t updated until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

**SEE ALSO:**

- Update Salesforce Records from a Flow
- Operators in Flow Record Filters
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Flow Elements
Flow Quick Action Element

Calls an object-specific or global quick action that’s already been configured in your organization. Only “Create,” “Update,” and “Log a Call” actions are supported.

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Related Record ID</td>
<td>Only for object-specific actions. The ID of the record from which the action executes.</td>
</tr>
<tr>
<td></td>
<td>For example, the action creates a case that’s associated with a given account. Assign the ID for that account to Related Record ID.</td>
</tr>
<tr>
<td></td>
<td>This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Input Parameter</td>
<td>Varies for each action.</td>
</tr>
<tr>
<td></td>
<td>The action layout determines which parameters are required. Required parameters appear by default and can’t be removed. If a required field has a default or predefined value, that field is optional in object-specific and global actions in the flow. If you later remove the field’s default or predefined value and you didn’t set a value in the flow, the interview fails at run time.</td>
</tr>
<tr>
<td></td>
<td>The value must be compatible with the parameter.</td>
</tr>
</tbody>
</table>

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.

Note: At run time, the record isn’t created or updated until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:
- Create Salesforce Records from a Flow
- Update Salesforce Records from a Flow
- Flow Elements
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows
Flow Post to Chatter Element

Posts a message to a specified feed, such as to a Chatter group or a case record. The message can contain mentions and topics, but only text posts are supported.

Inputs

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Community ID</td>
<td>ID of a community to post to. Valid only if Salesforce Communities is enabled. Required if posting to a user or Chatter group that belongs to a Salesforce.com Community. This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Message</td>
<td>The text that you want to post.</td>
</tr>
<tr>
<td></td>
<td>• 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: @{&quot;!UserId&quot;}.</td>
</tr>
<tr>
<td></td>
<td>• To add a topic, enter #[string], where string is the topic that you want to add. For example: #[Action Required].</td>
</tr>
<tr>
<td></td>
<td>This parameter accepts single-value resources of any type. That value is treated as text and is limited to 10,000 characters.</td>
</tr>
<tr>
<td>Target Name or ID</td>
<td>Reference to the user, Chatter group, or record whose feed you want to post to.</td>
</tr>
<tr>
<td></td>
<td>• To post to a user’s feed, enter the user’s ID or Username. For example: <a href="mailto:jsmith@salesforce.com">jsmith@salesforce.com</a></td>
</tr>
<tr>
<td></td>
<td>• To post to a Chatter group, enter the group’s Name or ID. For example: Entire Organization</td>
</tr>
<tr>
<td></td>
<td>• To post to a record, enter the record’s ID. For example: 001D000000JWBDx</td>
</tr>
<tr>
<td></td>
<td>This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Target Type</td>
<td>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:</td>
</tr>
<tr>
<td></td>
<td>• User—If Target Name or ID is set to a user’s Username, enter this value.</td>
</tr>
<tr>
<td></td>
<td>• Group—If Target Name or ID is set to a Chatter group’s Name, enter this value.</td>
</tr>
</tbody>
</table>
### Input Parameter

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| Visibility      | Specifies whether this feed item is available to all users or internal users only. Valid only if Salesforce Communities is enabled. Valid values are:  
  - allUsers  
  - internalUsers |

### Outputs

<table>
<thead>
<tr>
<th>Output Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feed Item ID</td>
<td>Assigns the created post’s ID to a resource in the flow. This parameter accepts any single-value variables of type Text, Picklist, or Picklist (Multi-Select).</td>
</tr>
</tbody>
</table>

### Usage

**Note:** At run time, the Chatter post isn’t created until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

**SEE ALSO:**
- Flow Elements
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows

### Flow Screen Element

Displays a screen to the user who is running the flow, which lets you display information to the user or collect information from the user.

**IN THIS SECTION:**
- Flow Screen Element: General Info
  - Identifies which navigation buttons are available for a given screen, as well as whether help text is available to the flow user.
- Flow Screen Element: User Input Fields
  - Lets users manually enter information. A flow user input field is a text box, long text area, number, currency, date, date/time, password, or checkbox in a Screen element.
- Flow Screen Element: Choice Fields
  - Lets users select from a set of choices. A flow choice field is a radio button, drop-down list, multi-select checkbox, or multi-select picklist in a Screen element.
Flow Screen Element: Display Text Fields
Displays information to users while they’re running the flow.

SEE ALSO:
Flow Elements
Define the Path That a Flow Takes

Flow Screen Element: General Info
Identifies which navigation buttons are available for a given screen, as well as whether help text is available to the flow user.

Navigation Options

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Drop-down list</td>
<td>Determines which navigation buttons display for this screen. 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.</td>
</tr>
</tbody>
</table>

- **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**.

Show Pause button
Adds the Pause button to the navigation buttons for this screen, if Let Users Pause Flows is enabled in your organization’s Workflow and Approvals settings. Once a user pauses a flow interview, only that user or an administrator can resume the interview.

**Note:** Users can’t resume paused flows from Lightning Experience, so we recommend removing the Pause button from flows that are distributed in Lightning Experience.

Paused Message
The message that’s displayed to flow users when they pause a flow.
### Help Text

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text box</td>
<td>Information for users to see when they click Help for this form.</td>
</tr>
<tr>
<td></td>
<td>This field accepts <strong>single-value resources of any type</strong>. That value is treated as text.</td>
</tr>
</tbody>
</table>

### Usage

If you allow users to pause interviews of this flow:

- Customize the **interview label**
- Enable **Let Users Pause Flows** in your organization’s Process Automation Settings
- Add the Paused Flow Interviews component to the Home page layout for relevant users

**SEE ALSO:**
- Flow Screen Element
- Design Home Page Layouts in Salesforce Classic

### Flow Screen Element: User Input Fields

Let users manually enter information. A **flow user input field** is a text box, long text area, number, currency, date, date/time, password, or checkbox in a Screen element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Label</strong></td>
<td>User-friendly text that displays to the left of the field.</td>
</tr>
<tr>
<td></td>
<td>To format the label, click <strong>Label</strong>.</td>
</tr>
<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>Input Type</td>
<td>Automatically populated based on the type of input field you selected.</td>
</tr>
<tr>
<td>Default Value</td>
<td>Pre-populated value for the input field. If the associated screen isn’t executed, the stored value of the input field is always null. 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.</td>
</tr>
<tr>
<td>Scale</td>
<td>Controls the number of digits to the right of the decimal point. Can’t exceed 17. If you leave this field blank or set to zero, only whole numbers display when your flow runs. Available for only Currency and Number input fields.</td>
</tr>
<tr>
<td>Required</td>
<td>Forces users to enter a value before they can move on to the next screen.</td>
</tr>
<tr>
<td>Validate</td>
<td>Enables input validation on this field.</td>
</tr>
</tbody>
</table>

**EDITIONS**

Available in: both **Salesforce Classic** and **Lightning Experience**

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula Expression</td>
<td>Boolean formula expression that evaluates whether the user entered an acceptable value.</td>
</tr>
<tr>
<td>Error Message</td>
<td>Displays underneath the field if the user didn’t enter an acceptable value. This field accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Help Text</td>
<td>Adds 🔄 next to the input field. In the text box, enter helpful information about this field. This field accepts single-value resources of any type. That value is treated as text.</td>
</tr>
</tbody>
</table>

SEE ALSO:
- Checkbox Input Fields
- Flow Screen Element

Flow Screen Element: Choice Fields

Lets users select from a set of choices. A flow choice field is a radio button, drop-down list, multi-select checkbox, or multi-select picklist in a Screen element.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Displays to the left of the field. To format the label, click 🖨️.</td>
</tr>
<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>Value Data Type</td>
<td>Controls which choices are available in Choice Settings. For example, if you choose Number you can’t use a choice that has a data type of Text. You can’t change the value data type of multi-select choice fields; only text is supported.</td>
</tr>
<tr>
<td>Scale</td>
<td>Controls the number of digits to the right of the decimal point. Can’t exceed 17. If you leave this field blank or set to zero, only whole numbers display when your flow runs. Available for only Currency and Number choice fields.</td>
</tr>
<tr>
<td>Required</td>
<td>Forces users to identify a choice before they progress to the next screen.</td>
</tr>
<tr>
<td>Default Value</td>
<td>The choice that’s preselected for the user. If the associated screen isn’t executed, the stored value of the choice field is always null.</td>
</tr>
<tr>
<td>Choice</td>
<td>The choice options that the user can choose from. Select configured choices, dynamic record choices, or picklist choices. Note: You can’t rearrange choices. The choice’s data type must match Value Data Type.</td>
</tr>
</tbody>
</table>

EDITIONS
Available in: both Salesforce Classic and Lightning Experience
Available in: Enterprise, Performance, Unlimited, and Developer Editions
### Field Description

| Help Text box | Information that users see when they click [ ].
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>This field accepts <strong>single-value resources of any type</strong>. That value is treated as text.</td>
</tr>
</tbody>
</table>

### SEE ALSO:

- Limitations for Multi-Select Choice Fields
- Options for Choice Fields in Flow Screen Elements
- Flow Screen Element

### Options for Choice Fields in Flow Screen Elements

A key part of configuring choice fields in a flow screen is selecting the choice options to display in that field. These options appear as the individual radio buttons or checkboxes or options in a drop-down list. Create choice options with at least one of these resources: choices, dynamic record choices, or picklist choices.

Dynamic record choices and picklist choices are easier to configure and don’t require as much maintenance as choices. We recommend using a flow choice resource (otherwise known as stand-alone choices) only when you can’t use the other two.

<table>
<thead>
<tr>
<th>If you want the user to select...</th>
<th>Use this resource</th>
</tr>
</thead>
<tbody>
<tr>
<td>From a set of filtered records</td>
<td>Dynamic Record Choice</td>
</tr>
<tr>
<td>From a set of values that correspond to an existing picklist or multi-select picklist field</td>
<td>Picklist Choice</td>
</tr>
<tr>
<td>Something that can’t be generated from a record, picklist field, or multi-select picklist field</td>
<td>Choice</td>
</tr>
</tbody>
</table>

### SEE ALSO:

- Flow Screen Element: Choice Fields
- Flow Dynamic Record Choice Resource
- Flow Picklist Choice Resource
- Flow Choice Resource

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions
Flow Screen Element: Display Text Fields

Displays information to users while they’re running the flow.

<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 box</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. This field accepts single-value resources of any type. That value is treated as text.</td>
</tr>
</tbody>
</table>

Example: Welcome users to the flow or display terms and conditions.

SEE ALSO:

Flow Screen Element

Flow Send Email Element

Sends an email by manually specifying the subject, body, and recipients in the flow.

Tip:
- Store the text for the email body in a text template.
- To use an email template that exists in your organization, call an email alert instead.

Specify at least one recipient for the email. You can use both email address parameters, so long as the combined number of addresses is five or fewer.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Body</td>
<td>Text for the body of the email. The email is treated as plain text; HTML formatting isn’t respected. This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Subject</td>
<td>Text for the subject of the email. This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<tr>
<td>Email Addresses (comma-separated)</td>
<td>Optional. Recipients of the email. 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.</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| This parameter accepts single-value resources of any type. That value is treated as text.  
| Email Addresses (collection)  | Optional. Recipients of the email. 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. This parameter accepts collection variables of type Text. |
| Sender Address                | The organization-wide email address that’s used to send the email. Required only if Sender Type is set to OrgWideEmailAddress. This parameter accepts single-value resources of any type. That value is treated as text.                                                                                                                                          |
| 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. (Default)  
|                               | • DefaultWorkflowUser—Email address of the default workflow user.  
|                               | • OrgWideEmailAddress—The organization-wide email address that is specified in Sender Address.                                                                                                                                                                                                                                             |

**Usage**

⚠️ **Note:** At run time, the email isn’t sent until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:

- Send Email from a Flow
- Flow Text Template Resource
- Customize What Happens When a Flow Fails
- Define the Path That a Flow Takes
- Cross-Object Field References in Flows

**Flow Step Element**

Acts as a placeholder when you’re not sure which element you need.

**Usage**

Steps aren’t valid elements for active flows. As a flow admin, you can run a draft flow with Step elements in it. Before you activate the flow, replace the Step elements with other elements or delete them entirely.

Convert a Step element into a Screen element at any time by hovering your mouse over the step and clicking 🎨.
Note:

- If you convert a step that has multiple connectors into a Screen, all its connectors are deleted.
- Once a Step element has been converted, you can’t use its original unique name.

SEE ALSO:

- Flow Screen Element
- Flow Elements
- Define the Path That a Flow Takes

Flow Submit for Approval Element

Submits one Salesforce record for approval.

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

Inputs

Transfer data from the flow to the approval submission.

<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record ID</td>
<td>The ID of the record that you want to submit for approval. This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
<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. This parameter accepts collection variables of type Text that include 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. 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 |
<table>
<thead>
<tr>
<th>Input Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>This parameter accepts single-value resources of any type. That value is treated as text.</td>
<td></td>
</tr>
<tr>
<td>Skip Entry Criteria</td>
<td>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.</td>
</tr>
<tr>
<td>This parameter accepts any single-value resource of type Boolean.</td>
<td></td>
</tr>
<tr>
<td>Submission Comments</td>
<td>Text that you want to accompany the submission. Don’t reference merge fields or formula expressions. Submission comments appear in the approval history for the specified record. This text also appears in the initial approval request email if the template uses the {!ApprovalRequest.Comments} merge field.</td>
</tr>
<tr>
<td>This parameter accepts single-value resources of any type. That value is treated as text.</td>
<td></td>
</tr>
<tr>
<td>Submitter ID</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>This parameter accepts single-value resources of any type. That value is treated as text.</td>
</tr>
</tbody>
</table>

### Outputs
Transfer data from the approval request to the flow. Assignments occur when the approval request is created.

<table>
<thead>
<tr>
<th>Optional Output Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance ID</td>
<td>The ID of the approval request that was submitted.</td>
</tr>
<tr>
<td>This parameter accepts single-value variables of type Text, Picklist, or Picklist (Multi-Select).</td>
<td></td>
</tr>
<tr>
<td>Instance Status</td>
<td>The status of the current approval request. Valid values are “Approved,” “Rejected,” “Removed,” or “Pending”.</td>
</tr>
<tr>
<td>This parameter accepts single-value variables of type Text, Picklist, or Picklist (Multi-Select).</td>
<td></td>
</tr>
<tr>
<td>New Work Item IDs</td>
<td>The IDs of the new items submitted to the approval request. There can be 0 or 1 approval processes.</td>
</tr>
<tr>
<td>This parameter accepts collection variables of type Text.</td>
<td></td>
</tr>
<tr>
<td>Next Approver IDs</td>
<td>The IDs of the users who are assigned as the next approvers.</td>
</tr>
<tr>
<td>This parameter accepts collection variables of type Text.</td>
<td></td>
</tr>
<tr>
<td>Record ID</td>
<td>The ID of the record that the flow submitted for approval.</td>
</tr>
<tr>
<td>This parameter accepts single-value variables of type Text, Picklist, or Picklist (Multi-Select).</td>
<td></td>
</tr>
</tbody>
</table>
Usage

Note: At run time, the approval request isn’t created until the interview’s transaction completes. Transactions complete either when the interview finishes, executes a Screen element, or executes a Wait element.

SEE ALSO:
Flow Elements
Customize What Happens When a Flow Fails
Define the Path That a Flow Takes
Cross-Object Field References in Flows

Flow Subflow Element

Calls another flow in your organization. Use this element to reference modular flows and simplify the overall architecture of your flow.

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.

⚠️ Tip: Create smaller flows that perform common tasks. For example, build utility flows to capture address and credit card information, and authorize a credit card purchase amount. Then call those flows as needed from multiple product-ordering flows.

Assign values to variables in the referenced flow. 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>
<tbody>
<tr>
<td>Target</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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.  
  Source’s data type must be compatible with Target. |

Assign values from the referenced flow’s variables to the master flow’s variables. Variable assignments occur when the referenced flow finishes running.
## 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.

Target’s data type must be compatible with Source.

### Usage

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:

- Open the master flow, and click Run with Latest in the button bar, or
- Append the URL for the master flow with ?latestSub=true

⚠️ **Note**: Only flow admins 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.

### Flow Wait Element

Waits for one or more defined events to occur, which lets you automate processes that require a waiting period.

Define the events that the flow waits for before it proceeds.

<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>
<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>Event Type</td>
<td>Determines whether the flow is waiting for:</td>
</tr>
<tr>
<td></td>
<td>• An absolute time (such as 3 days from now), or</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Event Conditions</td>
<td>Determines the exact event that the flow is waiting for. Parameters vary, based on the selected event type.</td>
</tr>
<tr>
<td>Wait for this event only if additional conditions are met</td>
<td>If selected, the flow waits for this event only when certain conditions are met.</td>
</tr>
<tr>
<td>Waiting Conditions</td>
<td>Determines which conditions must be true for the flow to wait for this event. Available only if Wait for this event only if additional conditions are met is selected. The flow waits for the event only if the waiting conditions evaluate to true. For details, see Define Flow Conditions on page 47.</td>
</tr>
<tr>
<td>Variable Assignments</td>
<td>Assigns the event’s outputs to flow variables. Parameters vary, based on the selected event type.</td>
</tr>
<tr>
<td>Default Path</td>
<td>Determines what the flow does when all the events have unmet waiting conditions. If at least one event doesn’t have waiting conditions, the default path is never executed. The name displays on the Wait element’s default connector. Provide a custom name for this path by replacing the predefined value.</td>
</tr>
</tbody>
</table>

**Usage**

- **Note:**
  - Flows that contain Wait elements must be autolaunched. If a flow includes Wait elements and screens, choice, or dynamic choices, you can’t activate or run it.
  - Before you add a Wait element to your flow, understand the special behavior and limitations. See Limitations for Waiting Flows on page 19 for details.

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 141.

IN THIS SECTION:

What Are Waiting Conditions?

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

SEE ALSO:

Customize What Happens When a Flow Fails
Define the Path That a Flow Takes
Cross-Object Field References in Flows
Flow Elements

What Are Waiting Conditions?

Each event that you define in a flow 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 the events have waiting conditions set, and
• You want the flow to proceed when the waiting conditions for all events 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 141.

SEE ALSO:
- Operators in Flow Record Filters
- Flow Wait Element
- Flow Event Types

Flow Resources

Each resource represents a value that you can reference throughout the flow.

In the Cloud Flow Designer, the Explorer tab displays the resources that are available in the flow. You can create some types of resources from the Resources tab by double-clicking them. Some resources, such as global constants and system variables, are automatically provided by the system. Other resources are provided by the system when you add an element to the flow. For example, when you add a Decision element to your flow, the system creates a resource for each outcome.

Which resources are available depend on the specific field you're setting. Oftentimes you can create resources from within that field by expanding the CREATE NEW section of its drop-down list.

<table>
<thead>
<tr>
<th>Flow Resource</th>
<th>Description</th>
<th>Creatable from the Resources Tab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice</td>
<td>Represents an individual value that can be used in choice screen fields.</td>
<td>Yes</td>
</tr>
<tr>
<td>Collection Variable</td>
<td>Stores multiple updatable values that have the same data type, such as a group of email addresses.</td>
<td>Yes</td>
</tr>
<tr>
<td>Constant</td>
<td>Stores a fixed value.</td>
<td>Yes</td>
</tr>
<tr>
<td>Dynamic Record Choice</td>
<td>Represents a set of choices that's generated from an object's records.</td>
<td>Yes</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</td>
<td></td>
</tr>
<tr>
<td>Flow Resource</td>
<td>Description</td>
<td>Createable from the Resources Tab</td>
</tr>
<tr>
<td>----------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-----------------------------------</td>
</tr>
<tr>
<td>Formula</td>
<td>Calculates a value by using other resources in your flow and Salesforce functions</td>
<td>Yes</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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>Picklist Choice</td>
<td>Represents a set of choices that’s generated from the values of a picklist or multi-select picklist field.</td>
<td>Yes</td>
</tr>
<tr>
<td>Picklist Values</td>
<td>System-provided values for picklist fields in sObject variables and sObject collection variables. Available for only Assignment and Decision elements.</td>
<td></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>
<td></td>
</tr>
<tr>
<td>SO Object Collection Variable</td>
<td>Stores updatable field values for one or more Salesforce records.</td>
<td>Yes</td>
</tr>
<tr>
<td>SO Object Variable</td>
<td>Stores updatable field values for a Salesforce record.</td>
<td>Yes</td>
</tr>
<tr>
<td>System Variable</td>
<td>System-provided values about the running flow interview, such as {!$Flow.CurrentDate}, {!$Flow.CurrentDateTime}, and {!$Flow.FaultMessage}.</td>
<td></td>
</tr>
<tr>
<td>Text Template</td>
<td>Stores formatted text.</td>
<td>Yes</td>
</tr>
<tr>
<td>Variable</td>
<td>Stores a value that can be updated as the flow executes.</td>
<td>Yes</td>
</tr>
<tr>
<td>Wait Event</td>
<td>If you add a Wait element to the flow, its events are available as Boolean resources. If an event’s waiting conditions are met, the resource’s value is True. If the event has no waiting conditions set, the resource’s value is always True.</td>
<td></td>
</tr>
</tbody>
</table>

IN THIS SECTION:

- **Flow Choice Resource**
  Represents an individual value to use in choice screen fields.

- **Flow Collection Variable Resource**
  Stores multiple updatable values that have the same data type, such as a group of email addresses.
Flow Constant Resource
A flow constant represents a fixed value. Unlike variables, this value can't change throughout a flow.

Flow Dynamic Record Choice Resource
Represents a set of choices that's generated from an object's records.

Flow Formula Resource
Calculates a value by using other resources in your flow and Salesforce functions.

Flow Global Constant Resource
Fixed, system-provided values, such as EmptyString, True, and False.

Global Variables in Visual Workflow
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 only in flow formulas.

Flow Picklist Choice Resource
Represents a set of choices that's generated from the values of a picklist or multi-select picklist field.

Flow sObject Collection Variable Resource
Stores updatable field values for one or more Salesforce records.

Flow sObject Variable Resource
Stores updatable field values for a Salesforce record.

System Variables in Flows
System-provided values about the running flow interview, such as {!$Flow.CurrentDate}, {!$Flow.CurrentDateTime}, and {!$Flow.FaultMessage}.

Flow Text Template Resource
Stores HTML-formatted text.

Flow Variable Resource
Stores a value that can be updated as the flow executes.

SEE ALSO:
Cloud Flow Designer

Flow Choice Resource
Represents an individual value to use in choice screen fields.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>A user-friendly label for the choice option.</td>
</tr>
<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>Description</td>
<td>Helps you differentiate this choice option from other resources.</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>
</tbody>
</table>
### DescriptionField

Controls the number of digits to the right of the decimal point. Can’t exceed 17. If you leave this field blank or set to zero, only whole numbers display when your flow runs.

Available for only Currency and Number choices.

### Stored Value

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 `null`.

### Show Input on Selection

Displays a text box below the choice option. This option isn’t available if the choice’s data type is `Boolean`.

These fields appear only if you select Show Input on Selection.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>A user-friendly label for the text box.</td>
</tr>
<tr>
<td>Required</td>
<td>Forces users to enter a value in the text box before they can progress or finish the flow.</td>
</tr>
<tr>
<td>Validate</td>
<td>Evaluates whether the user entered an acceptable value.</td>
</tr>
</tbody>
</table>

These fields appear only if you select Validate.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formula</td>
<td>Boolean formula expression that evaluates whether the user entered an acceptable value.</td>
</tr>
<tr>
<td>Error Message</td>
<td>Displays if the user didn’t enter an acceptable value.</td>
</tr>
</tbody>
</table>

**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.

### Rich Text in Choice Labels

You can configure every choice’s label using the rich text editor, but keep in mind where you want to use the choice.

**Dropdown List and Multi-Select Picklist Fields**

Rich text isn’t supported. If you include a choice with rich text in a dropdown list, at runtime the choice renders the HTML characters in their escaped form.

For example, the choice label `<b>My Label</b>` renders as `<b>My Label</b>` at runtime.

**Radio Button and Multi-Select Checkbox Fields**

- Rich text is supported.
- Don’t use angle brackets (`<` and `>`) except when applying HTML markup to your label. Instead, use square brackets (`[` and `]`). If a choice is used in a field that supports rich text, it treats anything in angle brackets as HTML. If the contents isn’t valid HTML, the content is stripped from the label.
For example, you include the “<My Label>” choice in a radio button or multi-select checkbox field. <My Label> is considered invalid HTML, so at runtime the value is stripped from the label. Rather than display nothing—because the label doesn’t include any other value—the flow displays the choice’s unique name: My_Label.

SEE ALSO:
Flow Resources
Flow Screen Element: Choice Fields
Options for Choice Fields in Flow Screen Elements
Cross-Object Field References in Flows

Flow Collection Variable Resource
Stores multiple updatable values that have the same data type, such as a group of email addresses.

<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 differentiate this collection variable from other resources.</td>
</tr>
<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
A flow constant represents a fixed value. Unlike variables, this value can’t change throughout a flow.

<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 differentiate this constant from other resources.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Determines the types of values that the constant can store.</td>
</tr>
<tr>
<td>Value</td>
<td>The constant’s value. This value doesn’t change throughout the flow.</td>
</tr>
</tbody>
</table>

Flow Dynamic Record Choice Resource

Represents a set of choices that’s generated from an object’s records.

<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 differentiate this resource from other resources.</td>
</tr>
<tr>
<td>Data Type</td>
<td>Data type of the choice’s stored value.</td>
</tr>
</tbody>
</table>
### Field | Description
--- | ---
Create a choice for each | Determines the object whose records you want to use to generate the choices

Filter Criteria | Determines which records are included in the generated list of choices. If you don’t apply any filters, a choice is generated for every record of the selected object. For example, to generate a list of all accounts in San Francisco, use filters to include only accounts whose Billing City is San Francisco.

Choice Label | Determines which field is used as the label for each generated choice. Select a field that enables users to differentiate between the generated choices. **Tip:** Make sure to choose a field that contains data. If the selected field has no value for a given record, the corresponding choice’s label is blank at run time.

Choice Stored Value | Determines which field’s value is stored when the user selects this choice at run time. Value Data Type determines the available options. By default, the stored value is null. The stored value is determined by the most recent user selection of a choice within the generated set.

Sort results by | Controls the order that the choices appear in. When Sort results by is selected, also select the field that you want to order the choices by. Then select which order the choices should appear in.

Limit number of choices to | Controls the number of options that appear in the screen field that uses this dynamic record choice. When Limit number of choices to is selected, also enter the maximum number (up to 200) of choices to include.

Assign the record fields to variables... | Takes field values from the record that the user chose and stores them in flow variables that you can reference later. **Note:** When a multi-select choice field uses a dynamic record 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 record choice, the variable assignments obey the first of those fields.

---

**Example:** In a support flow for a computer hardware manufacturer, users identify a product to find its latest updates. You create a dynamic record 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. Elsewhere in the flow, you want to display the associated product ID and description. To do so, you assign the ID and Description field values from the user-selected record to flow variables.

SEE ALSO:
- Operators in Flow Record Filters
- Flow Screen Element: Choice Fields
- Options for Choice Fields in Flow Screen Elements
- Flow Resources

**Flow Formula Resource**

Calculates a value by using other resources in your flow and Salesforce functions.

<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 differentiate this formula from other resources.</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>Controls the number of digits to the right of the decimal point. Can't exceed 17. If you leave this field blank or set to zero, only whole numbers display when your flow runs. Appears when Value Data Type is Number or Currency.</td>
</tr>
<tr>
<td>Formula</td>
<td>The formula expression that the flow evaluates at run time. The returned value must be compatible with Value Data Type.</td>
</tr>
</tbody>
</table>

Note: Some formula operators are not supported in the Cloud Flow Designer.

SEE ALSO:
- Formula Operators and Functions
- Which Functions Aren’t Supported in Flow Formulas?
- Flow Resources
- Cross-Object Field References in Flows
Flow Global Constant Resource

Fixed, system-provided values, such as EmptyString, True, and False.

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

Example: When you create a Boolean variable, $GlobalConstant.True and $GlobalConstant.False are supported. But when you create a Currency variable, no global constants are supported.

Null vs. Empty String

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).

SEE ALSO:
Flow Resources

Global Variables in Visual Workflow

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 only in flow formulas.

Example: Use {$!User.Id} to easily access the ID of the user who’s running the flow interview.

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>• Enterprise_Server_URL_***—The Enterprise WSDL SOAP endpoint where *** represents the version of the API.</td>
</tr>
<tr>
<td></td>
<td>• Partner_Server_URL_***—The Partner WSDL SOAP endpoint where *** represents the version of the API.</td>
</tr>
<tr>
<td></td>
<td>• Session_ID</td>
</tr>
<tr>
<td>Global Variable</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| $Label          | References custom labels. This global variable appears only if custom labels have been created in your org. The returned value depends on the language setting of the contextual user. The value returned is one of the following, in order of precedence:  
1. The local translation’s text  
2. The packaged translation’s text  
3. The master label’s text |
| $Organization   | References information about your company. |
| $Permission     | References information about the current user’s custom permission access. |
| $Profile        | References information from the current user’s profile, such as license type or name. |
| 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 org. 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 preference and permissions to see the Lightning Experience look and feel, but if they are using a browser that doesn’t support that look and feel, for example, older versions of Internet Explorer, $User.UIThemeDisplayed returns a different value. These merge fields return one of the following values.
Global Variable | Description
--- | ---
• Theme1—Obsolete Salesforce theme
• Theme2—Salesforce Classic 2005 user interface theme
• Theme3—Salesforce Classic 2010 user interface theme
• Theme4d—Modern “Lightning Experience” Salesforce theme
• Theme4t—Salesforce1 mobile Salesforce theme
• PortalDefault—Salesforce Customer Portal theme
• Webstore—Salesforce AppExchange theme

$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.
• 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.

Flow Picklist Choice Resource
Represents a set of choices that’s generated from the values of a picklist or multi-select picklist field.

<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 differentiate this resource from other resources.</td>
</tr>
<tr>
<td>Value Data Type</td>
<td>Data type of the choice’s stored value.</td>
</tr>
<tr>
<td>Object</td>
<td>The object whose fields you want to select from.</td>
</tr>
<tr>
<td>Field</td>
<td>The picklist or multi-select picklist field to use to generate the list of choices.</td>
</tr>
<tr>
<td>Sort Order</td>
<td>Controls the order that the choices appear in. The choices sort based on the translated picklist value for the running user’s language.</td>
</tr>
</tbody>
</table>

Example: In a flow that simplifies the process of creating an account, users identify the company’s industry.

Rather than creating one choice for each industry, you add a picklist choice to the flow and populate a drop-down list with it. When users run this flow, the picklist choice finds all the values in the database for the Industry picklist field (1) on the Account object (2).
On top of being easier to configure than the stand-alone choice resource, picklist choices reduce maintenance. When someone adds new options to the Account Industry picklist, the flow automatically reflects those changes; you don’t have to update the flow.

Limitations

Unlike with dynamic record choices, you can’t:

Filter out any values that come back from the database.
The flow always displays every picklist value for that field—even if you’re using record types to narrow down the picklist choices in page layouts.

Customize the label for each option.
The flow always displays the label for each picklist value.

Customize the stored value for each option.
The flow always stores the API value for each picklist value.

Picklists for Knowledge Articles aren’t supported.

Labels and Values for Translated Fields

When a picklist field has been translated:

- Each choice’s label uses the version of that picklist value in the running user’s language
- Each choice’s stored value uses the version of that picklist value in the org’s default language

SEE ALSO:
Flow Screen Element: Choice Fields
Options for Choice Fields in Flow Screen Elements
Flow Resources
## Flow sObject Collection Variable Resource

Stores updatable field values for one or more Salesforce records.

<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 differentiate this sObject collection variable from other resources.</td>
</tr>
<tr>
<td>Input/Output Type</td>
<td>Determines whether the sObject collection 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>
<tr>
<td></td>
<td>The default value is Private.</td>
</tr>
</tbody>
</table>

**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>Object Type</th>
<th>Type of Salesforce records that the sObject collection represents in the flow.</th>
</tr>
</thead>
</table>

## Usage

After you populate the sObject collection, reference it to create, update, or delete records in the Salesforce database.

Examine every item in the collection by using a Loop element. 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
- Flow Loop Element
- Flow Resources

## Flow sObject Variable Resource

Stores updatable field values for a Salesforce record.

<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 differentiate this sObject variable from other resources.</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>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>The default value of the field is Private.</td>
</tr>
</tbody>
</table>

⚠️ **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>Object Type</th>
<th>Type of Salesforce record that the sObject variable represents in the flow.</th>
</tr>
</thead>
</table>

Available in: both Salesforce Classic and Lightning Experience

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

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

SEE ALSO:

Flow Resources

System Variables in Flows

System-provided values about the running flow interview, such as `{!$Flow.CurrentDate}`, `{!$Flow.CurrentDateTime}`, and `{!$Flow.FaultMessage}`.

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Supported Resource Types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>{!$Flow.CurrentDate}</code></td>
<td>Text, Date, and DateTime</td>
<td>Date when the flow interview executes the element that references the system variable.</td>
</tr>
<tr>
<td><code>{!$Flow.CurrentDateTime}</code></td>
<td>Text, Date, and DateTime</td>
<td>Date and time when the flow interview executes the element that references the system variable.</td>
</tr>
<tr>
<td><code>{!$Flow.FaultMessage}</code></td>
<td>Text</td>
<td>System fault message that can help flow administrators troubleshoot run time issues.</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
Flow Text Template Resource
Stores HTML-formatted text.

<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 differentiate this text template from other resources.</td>
</tr>
<tr>
<td>Text Template</td>
<td>The text for the template. Use HTML to format the text and merge fields to reference information from other resources.</td>
</tr>
</tbody>
</table>

**Example:** You’re designing a flow that registers people for an event. You create a text template that includes a registrant’s name, address, and other information. Then you use the template in an email confirmation that the flow sends when it finishes.

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

Flow Variable Resource
Stores a value that can be updated as the flow executes.

<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 differentiate this variable from other resources.</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>Controls the number of digits to the right of the decimal point. Can’t exceed 17. If you leave this field blank or set to zero, only whole numbers display when your flow runs. Appears only when the Data Type is set to Number or Currency.</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>
</tbody>
</table>

**Editions**
Available in: both Salesforce Classic and Lightning Experience

Available in: Enterprise, Performance, Unlimited, and Developer Editions
### Field

**Description**

- **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 does not 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

Determines the variable value when the flow starts. If you leave this field blank, the value is **null**.

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

### Usage

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

SEE ALSO:

- [Flow Resources](#)
- [Flow Assignment Element](#)
- [Flow sObject Variable Resource](#)
- [Cross-Object Field References in Flows](#)

### 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.

#### IN THIS SECTION:

- **Tips for Cross-Object Field References in Flows**
  
  Cross-object field values are valid wherever you can reference a flow resource or manually enter a value. Keep these implementation tips in mind when you use a cross-object field reference.

- **Cross-Object Field References in Flows: Simple Relationships**
  
  Most relationships are straightforward. For example, *Case.AccountId* links directly to the case’s parent account. If you know that a field relationship ties your object to exactly one other object, use this syntax.

- **Cross-Object Field References in Flows: Polymorphic Relationships**
  
  Some fields have relationships to more than one object. We call these relationships **polymorphic**. For example, if you have queues enabled for cases, a case owner can be either a user or queue. If you’re traversing from a case to its owner ID, add special syntax to identify which object you mean when you say “Owner”.

---

**EDITIONS**

Available in: both Salesforce Classic and Lightning Experience

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

---

107
Example Cross-Object Field References in Flows

This example demonstrates how to update a contract’s owner to be the contract’s account’s owner.

Tips for Cross-Object Field References in Flows

Cross-object field values are valid wherever you can reference a flow resource or manually enter a value. Keep these implementation tips in mind when you use a cross-object field reference.

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.

Unsupported Relationships

The following relationships aren’t supported in cross-object field references.

- Lead ConvertedAccount
- Lead ConvertedContact
- Lead ConvertedOpportunity

Avoiding Null Values

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, using another flow, 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 and Org Limits

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

- Cross-object relationships per object
- DML operations per transaction
Cross-Object Field References in Flows: Simple Relationships

Most relationships are straightforward. For example, `Case.AccountId` links directly to the case’s parent account. If you know that a field relationship ties your object to exactly one other object, use this 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_Case.Account.Id}` references `Id` of the account that’s related to the case record represented by an sObject variable in the flow.

Cross-Object Field References in Flows: Polymorphic Relationships

Some fields have relationships to more than one object. We call these relationships polymorphic.

For example, if you have queues enabled for cases, a case owner can be either a user or queue. If you’re traversing from a case to its owner ID, add special syntax to identify which object you mean when you say “Owner”.

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

```{!sObjectVariable.polymorphicObjectName1:specificObjectName2.fieldName}
```

where:

- `sObjectVariable` is the unique name for the sObject variable that you want to start from.
- `polymorphicObject` is the API name for a polymorphic relationship for `sObjectVariable`’s object type.
- `specificObjectName` is the API name for the object that you want to select from the polymorphic relationship.
- `fieldName` is the name for the field that you want to reference on the last object in the expression. All custom field API names end in `__c`.
For example: `{sObj_Case.Owner:User.Id}` references the ID of the user who owns the case, while `{sObj_Case.Owner:Queue.Id}` references the ID of the queue who owns the case. You can always add the polymorphic reference after several traversals (`{sObj_Case.Account.Owner:User.Id}`) or in the middle of a reference (`{sObj_Case.Owner:User.Manager.Id}`).

**Supported Polymorphic Relationships**

Not every relationship is polymorphic, so we recommend using the polymorphic syntax only when you know that the field can link to multiple objects. The following relationships are supported.

- Case.Source
- FeedItem.CreatedBy
- Object.Owner

Where `Object` lets you set Owner to either a user or a queue. `Group.Owner` and `Queue.Owner` aren’t supported.

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 Cross-Object Field References in Flows**

This example demonstrates how to update a contract’s owner to be the contract’s account’s owner.

**Example:**

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 element to find the contract’s account.
4. Use a Decision element to verify that the owner is set on the account.
5. Use an Assignment element to assign the owner to the contract.
6. Use a Fast Update element to update the contract’s owner.

**Available in:** both Salesforce Classic and Lightning Experience.
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.

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><img src="image" alt="Unlabeled" /></td>
<td>Identifies which element to execute next.</td>
</tr>
<tr>
<td>Decision</td>
<td>X &gt; Y</td>
<td>Identifies which element to execute when the criteria of a Decision outcome are met.</td>
</tr>
<tr>
<td>Wait event</td>
<td>1 day after X</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><img src="image" alt="FAULT" /></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</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</td>
<td>Identifies which element to execute after a Loop element finishes iterating through a collection.</td>
</tr>
</tbody>
</table>

SEE ALSO:
Flow Elements

Flow Operators

Operators behave differently, depending on what you’re configuring. In Assignment elements, operators let you change resource values. In flow conditions and record filters, operators let you evaluate information and narrow the scope of a flow operation.

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).
**Example Supported Data Types**

<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. |
| **add** | What you enter or select for Value is added as a new item at the end of the collection. | 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. | Before the Assignment:  
  • `{!collText}` is Yellow, Green, Blue  
  • `{!varPicklist}` is Red  
Assignment: `{!collText} add {!varPicklist}`.  
After the Assignment: `{!collText}` is Yellow, Green, Blue, Red. |

**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>
</table>
| **equals** | The number that you enter or select for Value replaces the value of Variable. | Currency, Number | Before the Assignment: `{!varCurrency}` is 10.  
Assignment: `{!varCurrency} equals 7`.  
After the Assignment: `{!varCurrency}` is 7. |
| **add** | The number that you enter or select for Value is added to the value of Variable. | Currency, Number | Before the Assignment: `{!varCurrency}` is 10.  
Assignment: `{!varCurrency} add 7`.  
After the Assignment: `{!varCurrency}` is 17. |
| **subtract** | The number that you enter or select for Value is subtracted from the value of Variable. | Currency, Number | Before the Assignment: `{!varCurrency}` is 10.  
Assignment: `{!varCurrency} subtract 7`.  
After the Assignment: `{!varCurrency}` is 3. |

**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>
</table>
| equals   | The date that you enter or select for Value replaces the value of Variable. | • Date  
• Date/Time | Before the Assignment: `{!varDate}` is 1/16/2016. Assignment: `{!varDate} equals 1/15/2016`. After the Assignment: `{!varDate}` is 1/15/2016. |
| 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 | 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).

⚠️ **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. |
### Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Supported Data Types</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>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 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>Boolean, Currency, Date, Date/Time, Multi-Select Picklist, Number, Picklist, Text</td>
<td>Before the Assignment: <code>{!varMSP}</code> is Blue; Green. This value includes two separate selections. Assignment: <code>{!varMSP} add Yellow.</code> After the Assignment: <code>{!varMSP}</code> is Blue; GreenYellow. This value includes two 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>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>
</tbody>
</table>

### sObject

Replace an sObject variable with a new value (equals).

<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>The sObject that you select for Value replaces the value of Variable. sObject with the same object type</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>Before the Assignment: Assignment: <code>{!account1} equals </code>{!account2}.`</td>
</tr>
</tbody>
</table>
**Text**

Replace a text resource with a new value (equals) or concatenate a value onto the end of the original value (add).

**Note:** Before values are assigned or added to a text resource, they’re converted into string values.

<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. |
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.

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

**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.</td>
<td>Boolean</td>
</tr>
<tr>
<td></td>
<td>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></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.
## Operator

### True if...

<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 record 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>
<tr>
<td>does not equal</td>
<td>The collection that's selected for Resource doesn't match the collection that's selected for Value</td>
<td>Collection</td>
</tr>
<tr>
<td>equals</td>
<td>The collection that's selected for Resource matches the collection that's selected for Value</td>
<td>Collection</td>
</tr>
</tbody>
</table>
### Supported Data Types

- Otherwise, only collection variables with the same data type are supported.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<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>
<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.
**Operator** | **True if...** | **Supported Data Types**
--- | --- | ---
does not equal | The value for Resource doesn’t match what’s entered or selected for Value | • Date  
|  |  | • Date/Time

equals | The value for Resource matches what’s entered or selected for Value | • Date  
|  |  | • Date/Time

greater than | The value of the Resource is a later date or time than what’s entered or selected for Value | • Date  
|  |  | • Date/Time

greater than or equal | 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 | • Date  
|  |  | • Date/Time

less than | The value of the Resource is an earlier date or time than what’s entered or selected for Value | • Date  
|  |  | • Date/Time

less than or equal | 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 | • Date  
|  |  | • Date/Time

is null | Resource isn’t populated with a value | Boolean

was set | 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 | Boolean

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

⚠️ **Note:** These operators treat the resource’s value as a text value.

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
</table>
| contains | The value for Resource contains what’s entered or selected for Value | • Boolean  
|  |  | • Currency  
|  |  | • Date  
|  |  | • Date/Time  
|  |  | • Multi-select  
|  |  | • Picklist  
|  |  | • Number  
|  |  | • Picklist  
|  |  | • Text  

For example, if the value of {!varPicklist} is yellow-green, the condition {!varPicklist} contains green evaluates to true.
<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>• Boolean&lt;br&gt;• Currency&lt;br&gt;• Date&lt;br&gt;• Date/Time&lt;br&gt;• Multi-select Picklist&lt;br&gt;• Number&lt;br&gt;• Picklist&lt;br&gt;• Text</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource matches what’s entered or selected for Value</td>
<td>• Boolean&lt;br&gt;• Currency&lt;br&gt;• Date&lt;br&gt;• Date/Time&lt;br&gt;• Multi-select Picklist&lt;br&gt;• Number&lt;br&gt;• Picklist&lt;br&gt;• Text</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>

**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>
</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
### Supported Data Types

<table>
<thead>
<tr>
<th>Operator</th>
<th>True if...</th>
<th>Supported Data Types</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>acceptable values, use the contains operator in a flow condition. If only &quot;green&quot; is an acceptable value, create a formula that uses the INCLUDES() function.</td>
<td>Picklist, 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, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist, Text</td>
</tr>
<tr>
<td>equals</td>
<td>The value for Resource exactly matches what’s entered or selected for Value</td>
<td>Boolean, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist, Text</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>

### 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</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</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>
<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</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>
<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</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>
<tr>
<td>ends with</td>
<td>The end 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>Operator</td>
<td>True if...</td>
<td>Supported Data Types</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>----------------------</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, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist, Text</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>

**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

**Editions**

Available in: both Salesforce Classic and Lightning Experience

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

<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>

⚠️ Tip: Flow treats null as a different value than false. If you filter for records whose checkbox field is null, no records are returned.

## 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>
<tbody>
<tr>
<td>does not equal</td>
<td>Doesn’t match what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>equals</td>
<td>Matches what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>greater than</td>
<td>Is larger than what’s entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>greater than or equal</td>
<td>Is larger than what’s entered or selected for Value or is the same</td>
<td>Currency, Number</td>
</tr>
</tbody>
</table>
### Supported Data Types

<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>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>Is smaller than what's entered or selected for Value</td>
<td>Currency, Number</td>
</tr>
<tr>
<td>less than or equal</td>
<td>Is smaller than what's entered or selected for Value or is the same.</td>
<td>Currency, Number</td>
</tr>
</tbody>
</table>

### 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, 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>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, 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>

### 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>
<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>
</tbody>
</table>

Flow Reference: Visual Workflow Guide
<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>starts with</td>
<td>Begins with what’s entered or selected for Value</td>
<td>Boolean, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist, Text</td>
</tr>
</tbody>
</table>

**Multi-Select Picklist Fields**

When you select a multi-select picklist field under Field, these operators are available.

**Tip:** Be careful when using these operators to filter records based on a multi-select picklist field. Even if two resources have the same items in a multi-select picklist, they can be mismatched if these cases differ.

- The spacing before or after the semi-colon. For example, one resource’s value is “red; green; blue” and the other’s value is “red;green;blue”
- The order of the items. For example, one resource’s value is “red; green; blue” and the other’s value is “red; blue; green”

For best results, use the INCLUDES function in a flow formula.

<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, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist, Text</td>
</tr>
<tr>
<td>equals</td>
<td>Matches what’s entered or selected for Value</td>
<td>Boolean, Currency, Date, Date/Time, Multi-select Picklist, Number, Picklist</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>ends with</td>
<td>Ends with what’s entered or selected for Value</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>

Flow Event Types

Event Type drives the fields that you use to define an event in a flow Wait element. You can use a platform event, which you can fully customize. You can also use an alarm consisting of a date/time value—the base time—and an optional offset from that time.

The following applies to both types of alarms.

The base time, which is 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:

Flow Wait Event Type: Platform Events

If you’ve defined platform events in your org, you can choose that event as the Event Type in your wait event.
Flow Wait Event Type: 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.

Flow Wait Event Type: 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:
Flow Wait Element

Flow Wait Event Type: Platform Events

If you’ve defined platform events in your org, you can choose that event as the Event Type in your wait event.

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 parameters for a platform event are the fields available on that event’s definition.

For an example of a flow that waits for a platform event, see Sample Flow That Waits for a Platform Event.

Event Outputs

Platform events return one output value: one sObject value for the entire published event.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event</td>
<td>To reference data from the platform event in your flow, pass the event into an sObject variable. When you create the sObject variable, for Object select the event definition’s API name.</td>
<td>{!vendorResponse}</td>
</tr>
</tbody>
</table>

For example, to reference Expected Delivery Date from a Vendor Response event, pass the Vendor Response data to the {!vendorResponse} sObject variable. Then reference {!vendorResponse.Expected_Delivery_Date__c} to get the specific field value.
Flow Wait Event Type: 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>
Delivered

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: 
Flow Event Types

Flow Wait Event Type: 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>Object Type</td>
<td>The API name of the object whose field you want to base the alarm on. See Supported Objects. on page 134</td>
<td>Contract</td>
</tr>
<tr>
<td></td>
<td>You must manually enter a string. You can’t use a merge field or flow resource for this value.</td>
<td></td>
</tr>
<tr>
<td>Base Date/Time Field</td>
<td>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.</td>
<td>EndDate</td>
</tr>
<tr>
<td></td>
<td>Manually enter a string.</td>
<td></td>
</tr>
<tr>
<td>Record ID</td>
<td>ID of the record that the alarm is based on. The record’s object type must match Object Type.</td>
<td>{!ContractId}</td>
</tr>
<tr>
<td></td>
<td>You can enter a string, merge field, or flow resource.</td>
<td></td>
</tr>
<tr>
<td>Offset Number</td>
<td>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.</td>
<td>-3</td>
</tr>
</tbody>
</table>

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Offset Unit

Optional. The unit to offset Base Date/Time Field. Required Days 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.

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.

<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>

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
- Contact
- Contract
- ContractLineItem
- DandBCompany
- DuplicateRecordItem
- DuplicateRecordSet
- EmailMessage
- Entitlement
- EntitlementContact
- EnvironmentHubMember
- 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
• QuickText
• Quote
• QuoteLineItem
• Reply
• SSOSSession
• SSOSSessionActivity
• 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
• WorkRewardFundType
• WorkThanks
• WorkUpgradeAction
• WorkUpgradeCustomer
• WorkUpgradeUser
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.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td>Requires user interaction, because it has one or more screens, steps, choices, or dynamic choices.</td>
</tr>
<tr>
<td></td>
<td>This flow type doesn’t support wait elements.</td>
</tr>
<tr>
<td></td>
<td>A flow can be implemented with a custom button, custom link, direct URL, Visualforce page, or Salesforce1 action.</td>
</tr>
<tr>
<td>Autolaunched Flow</td>
<td>Doesn’t require user interaction.</td>
</tr>
<tr>
<td></td>
<td>This flow type doesn’t support screens, steps, choices, or dynamic choices.</td>
</tr>
<tr>
<td></td>
<td>An autolaunched flow can be implemented any way that a flow can, as well as with a process action, workflow action (pilot), or Apex code.</td>
</tr>
<tr>
<td>User Provisioning Flow</td>
<td>Provisions users for third-party services. A user provisioning flow can only be implemented by associating it with a connected app when running the User Provisioning Wizard. Provisions users for third-party services. For example, use this flow type to customize the user provisioning configuration for a connected app to link Salesforce users with their Google Apps accounts.</td>
</tr>
</tbody>
</table>

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.”
Supported in Translation Workbench

<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
- Flow and Flow Version Fields
- User Provisioning for Connected Apps

Flow Properties

A flow’s properties consist of its name, description, interview label, and type. These properties drive the field values that appear on a flow or flow version’s detail page. 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 to update the properties for a flow or a flow version.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>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 runtime user interface. You can edit the name for inactive flows and flow versions.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>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.</td>
</tr>
<tr>
<td>Type</td>
<td>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 137. If the type is Login Flow, you can’t update the type after the flow has been saved.</td>
</tr>
</tbody>
</table>
### Property | Description
--- | ---
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:
- Save a Flow
- Flow and Flow Version Fields

## Sample Flows

Sometimes showing is better than telling. Check out these sample flows to get a feel for how to work with advanced things like Wait elements and collection variables.

### IN THIS SECTION:
- **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.
- **Sample Flows That Wait for Events**
  Configure a flow to wait for events in one of three ways.
- **Sample Flow That Loops Through a Collection**
  Transfer ownership of accounts from one user to another by using sObject variable collections and loops. The flow already has the required user IDs.

### 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:
Flow Collection Variable Resource
Add Values to a Collection Variable

Sample Flows That Wait for Events
Configure a flow to wait for events in one of three ways.

IN THIS SECTION:
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.
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 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 a Platform Event
You’re designing a flow that places a supply order and waits for shipment confirmation from the vendor. Then it assigns an installation task the day after the supplies are expected to be delivered.

SEE ALSO:
Flow Wait Element

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 consists 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 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.

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 renews or the contract ends. The flow sends a reminder email for whichever date 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.

**Tip:** Every alarm event consists 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).

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).
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:
- Flow Wait Element
- Flow Wait Event Type: Relative Time Alarms
- Flow Wait Element
- Flow Wait Event Type: Relative Time Alarms
- What Are Waiting Conditions?

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:
- Flow Wait Element
- Flow Wait Event Type: Absolute Time Alarms

Sample Flow That Waits for a Platform Event

You’re designing a flow that places a supply order and waits for shipment confirmation from the vendor. Then it assigns an installation task the day after the supplies are expected to be delivered. The vendor that you buy supplies from has set up a platform event for you to subscribe to. This event, called Vendor Response, includes the order number, order status, and expected delivery date.

Note: This flow is part of a larger example. It gets launched by a process that starts when a Printer Status platform event occurs. For details about the process, see Sample Process: Printer Management.

The Order Printer Supplies flow starts when the Printer Management process launches it. The process populates the following variables in the flow.

- `{!assetId}`—The asset’s ID
- `{!assetOwner}`—The asset’s owner
- `{!inkManufacturer}`—The manufacturer of the printer’s ink
- `{!inkNeeded}`—Whether the printer needs more ink
- `{!inkType}`—Specific type of ink that the printer uses
- `{!paperNeeded}`—Whether the printer needs more paper
- `{!paperSize}`—Paper size that the printer uses
- `{!serialNumber}`—The asset’s serial number

First, the flow determines whether to order ink or paper. Based on the decision, it invokes Apex code to order ink or paper from the vendor. Then it waits for the vendor to send a platform event of type Vendor Response that says the order has been shipped. When Salesforce receives the specified event, the flow resumes and creates a task for the asset’s owner to install the new supplies.
Decision Element

The decision includes two outcomes: Ink and Paper. The Ink outcome is true if the variable `{!inkNeeded}` is true. The Paper outcome is true if the variable `{!paperNeeded}` is true.

### Ink Outcome

- Name: Ink
- Unique Name: Ink

<table>
<thead>
<tr>
<th>Resource</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{!inkNeeded}</td>
<td>equals</td>
<td>{$GlobalConstant.True}</td>
</tr>
</tbody>
</table>

### Paper Outcome

- Name: Paper
- Unique Name: Paper

<table>
<thead>
<tr>
<th>Resource</th>
<th>Operator</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>{!paperNeeded}</td>
<td>equals</td>
<td>{$GlobalConstant.True}</td>
</tr>
</tbody>
</table>

Apex Elements

The flow includes two Apex elements that submit a supply order with a vendor but provide different information to it based on whether the flow executed the Ink outcome or Paper outcome. All the variables used for input values (like `{!serialNumber}` and `{!paperSize}`) are set when a process launches the flow.
The first Apex element provides information about which ink to order.

The second Apex element provides information about which paper to order.

In both Apex elements, after the class submits the order, it returns an order number. The flow stores that value in the {!orderNumber} variable to use in the Wait element.

Wait Element

After the Apex class submits the supply order, the flow waits for confirmation that the order has been shipped. That confirmation is received through the Vendor Response platform event.

The flow waits for a specific Vendor Response. The order number must be the same as the order number that the Apex class provided. And the order status must be Shipped.

When the correct event occurs and the flow resumes, the flow stores the event’s data in an sObject variable. That way, you can reference the expected delivery date to calculate when the supplies are scheduled to be installed.

Record Create Element

When the flow resumes, it creates a task for the asset owner to install the new supplies.

For the task’s field values, the flow uses these resources.

• {!installDate} — A formula that calculates the day after the event’s expected delivery date.
Sample Flow That Loops Through a Collection

Transfer ownership of accounts from one user to another by using sObject variable collections and loops. 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.

<table>
<thead>
<tr>
<th>ActivityDate</th>
<th>InstallDate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Description</td>
<td>TaskDescription</td>
</tr>
<tr>
<td>OwnerId</td>
<td>AssetOwner</td>
</tr>
<tr>
<td>Priority</td>
<td></td>
</tr>
<tr>
<td>Status</td>
<td>Not Started</td>
</tr>
<tr>
<td>Subject</td>
<td>Install ink on printer</td>
</tr>
<tr>
<td>WhatId</td>
<td>AssetId</td>
</tr>
</tbody>
</table>
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).

Manage Your Flows

Use the flow detail page to do anything with your flow outside of designing it—such as activating a flow, testing it, or viewing its properties.

To visit a flow’s detail page, from Setup, enter Flows in the Quick Find box, select Flows, and then click a flow name.

IN THIS SECTION:

Flow and Flow Version Fields
View information about a flow and its versions on the flow detail page, like its name and URL.

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

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

Activate or Deactivate a Flow Version
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.
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.

Delete a Flow Version
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.

Let Users Pause Flows
Enable your users to pause a flow interview that they can’t finish yet by customizing your organization’s process automation settings. 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 the necessary information.

SEE ALSO:
Visual Workflow
Limits for Visual Workflow
Considerations for Managing Flows

Flow and Flow Version Fields
View information about a flow and its versions on the flow detail page, like its name and URL.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active Version</td>
<td>Identifies which version is active.</td>
</tr>
<tr>
<td>Description</td>
<td>The description for the flow or flow version</td>
</tr>
<tr>
<td>Flow Name</td>
<td>The name for the flow. It appears in the run time user interface.</td>
</tr>
<tr>
<td>Name</td>
<td>The name for the flow version. It becomes the Flow Name when this version is active.</td>
</tr>
<tr>
<td>Namespace Prefix</td>
<td>The flow’s namespace prefix, if it was installed from a managed package. The Cloud Flow Designer can’t open flows that are installed from managed packages.</td>
</tr>
<tr>
<td>Type</td>
<td>Determines which elements and resources are supported in the flow or flow version, as well as the ways that the flow can be distributed. For details, see Flow Types on page 137.</td>
</tr>
<tr>
<td>Status</td>
<td>Identifies whether the flow version is active or not.</td>
</tr>
<tr>
<td>Unique Name</td>
<td>Lets you refer to the flow from other parts of Salesforce, such as in Visualforce page.</td>
</tr>
<tr>
<td>URL</td>
<td>The relative URL that you can use to run the flow, such as from a custom button or Web tab.</td>
</tr>
</tbody>
</table>
### Property Description

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>The number of the flow version.</td>
</tr>
</tbody>
</table>

**SEE ALSO:**
- Manage Your Flows
- 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, enter `Flows` in the Quick Find box, then select `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:**
- Considerations for Designing Flows
- Manage Your Flows
- Activate or Deactivate a Flow Version

### EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

### USER PERMISSIONS

To open a flow in the Cloud Flow Designer:
- Manage Force.com 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, enter Flows in the Quick Find box, then select 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 or Deactivate a Flow Version
- Customize What Happens When a Flow Fails
- Considerations for Running Flows
- Manage Your Flows

Activate or Deactivate a Flow Version

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, enter Flows in the Quick Find box, then select Flows.
2. Click the name of the flow.
3. Click **Activate** or **Deactivate** next to the relevant version of the flow.

SEE ALSO:
- Considerations for Managing Flows
- 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, enter **Flows** in the Quick Find box, then select **Flows**.
   
   If there are waiting interviews for any of your flows, the Paused and Waiting Interviews related list appears underneath the list of flows.

2. For each interview that you want to delete, click **Del**.

### Delete a Flow Version

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, enter **Flows** in the Quick Find box, then select **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:
- Considerations for Managing Flows
- Manage Your Flows
Let Users Pause Flows

Enable your users to pause a flow interview that they can’t finish yet by customizing your organization’s process automation settings. 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 the necessary information.

1. From Setup, enter Process Automation Settings in the Quick Find box, then select Process Automation 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:
Flow Screen Element: General Info

Distribute Your Flow

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

IN THIS SECTION:
Flow Runtime Experiences
Depending on how a flow is distributed, users see either the Classic runtime or Lightning runtime UI when they run the flow. Like its name suggests, Lightning runtime looks and feels like Lightning Experience.

Distribute a Flow to Internal Users
Enable your internal users to run your flow through a custom action, the flow URL, a Lightning page, a Visualforce page, or a custom Lightning component.

Distribute a Flow to External Users
Let external users run your flow by adding the flow to a Lightning community or external app or page. For finer control over how your flow behaves in external contexts, use a custom Lightning component or Visualforce page. Flows in custom Lightning components use Lightning runtime, and flows in Visualforce pages use Classic runtime.

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.
Flow Runtime Experiences

Depending on how a flow is distributed, users see either the Classic runtime or Lightning runtime UI when they run the flow. Like its name suggests, Lightning runtime looks and feels like Lightning Experience.

Here's the same flow rendered in Classic runtime (left) and Lightning runtime (right).

Which Runtime Experience Do My Users See?

Flows that run from a Visualforce page always use Classic runtime. Flows that run from a Lightning page always use Lightning runtime. All other methods depend on whether Lightning runtime has been enabled in your org’s Process Automation settings.

This table summarizes which runtime experience your users see based on how you distribute the flow.

<table>
<thead>
<tr>
<th>Flow Distribution Method</th>
<th>When Lightning Runtime for Flows is</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not selected</td>
<td>Selected</td>
</tr>
<tr>
<td>Visualforce page</td>
<td>Classic runtime</td>
<td>Classic runtime</td>
</tr>
<tr>
<td>Custom button</td>
<td>Classic runtime</td>
<td>Lightning runtime</td>
</tr>
<tr>
<td>Custom link</td>
<td>Classic runtime</td>
<td>Lightning runtime</td>
</tr>
<tr>
<td>Web tab</td>
<td>Classic runtime</td>
<td>Lightning runtime</td>
</tr>
<tr>
<td>Direct link</td>
<td>Classic runtime</td>
<td>Lightning runtime</td>
</tr>
</tbody>
</table>
**Flow Distribution Method**

<table>
<thead>
<tr>
<th></th>
<th>When Lightning Runtime for Flows is</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Not selected</td>
</tr>
<tr>
<td>Lightning page</td>
<td>Lightning runtime</td>
</tr>
</tbody>
</table>

SEE ALSO:

- Choose Your Org’s Runtime Experience for URL-Based Flows
- Considerations for Running Flows

## Distribute a Flow to Internal Users

Enable your internal users to run your flow through a custom action, the flow URL, a Lightning page, a Visualforce page, or a custom Lightning component.

**IN THIS SECTION:**

- **Embed a Flow in a Lightning Page**
  
  To easily distribute a flow to Lightning Experience or Salesforce1 users, embed it in a Lightning page.

- **Create a Flow Action (Beta)**
  
  To easily distribute a flow to Lightning Experience or Salesforce1 users, create a flow action and add it to the appropriate page layout. Flows aren’t supported for global actions.

- **Add a Flow to the Utility Bar**
  
  Want your flow to be accessible from any page in your app? Add it to the utility bar in your Lightning app. The utility bar gives your users quick access to commonly used tools.

- **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 for Internal Users**
  
  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.

- **Embed a Flow in a Custom Lightning Component**
  
  To customize how your flow gets and receives data, add it to a custom Lightning component. Then distribute that component through a custom action, Lighting tab, or Lightning page.
Embed a Flow in a Lightning Page

To easily distribute a flow to Lightning Experience or Salesforce1 users, embed it in a Lightning page.

Available in: Lightning Experience
Available in: Enterprise, Performance, Unlimited, and Developer Editions

If you’re not yet familiar with the types of Lightning pages you can customize, check out the Lightning App Builder module in Trailhead. If your org uses Lightning for Outlook or Lightning for Gmail, you can also create custom email application panes.

1. Open a Lightning page in the Lightning App Builder.
2. From the Lightning Components pane on the left, drag the Flow component onto the Lightning page canvas.
3. Configure the component.

**Flow**

Only active flows of type Flow are available. Flows that were built in the Desktop Flow Designer aren’t supported.

**Layout**

By default, flows display in one column.

**Input variables**

If you see other properties, they are the flow’s input variables. Variables appear only if they allow input access.

**Pass record ID into this variable**

This option is available only for Text input variables in Record pages. For simplicity, we recommend passing the ID to only one variable.

For example, when this component is embedded in an Opportunity Record page, at runtime the component passes the opportunity’s ID into the selected input variable.

4. Save the page.
5. Hang on, you’re not done yet! To make your page available to your users, activate it. You can activate the page from the Save dialog when you save it for the first time, or later using the **Activation** button.
6. Test that the flow is working correctly, and then roll the Lightning page out to your users.

Considerations and Limitations for Flows in Lightning Pages

Here are some things to keep in mind when you add a flow component to a Lightning page.

Lightning pages always use Lightning runtime, so also review the **Lightning Runtime Limitations**.

**Running Flows from a Lightning Page**

When a user opens a Lightning page that has a flow component, the flow runs when the page loads. Make sure that the flow doesn’t perform any actions – such as create or delete records – before the first screen.

**Input Variable Limitations**

- These variables aren’t supported.
  - Collection variables
  - sObject variables
sObject collection variables

- The component supports only manually entered values for input variables.
- Text input variables accept a maximum length of 4,000 characters.

Deployment Considerations
Change sets and the Metadata API deploy all flows as inactive, which users can’t run. If you deploy a Lightning page (known as FlexiPage in the API) that contains a flow component, make sure to activate the flow.

SEE ALSO:
- Limits and Considerations for Visual Workflow
- Lightning Pages
- Lightning Page Considerations and Limitations
- Considerations for Two-Column Flows

Create a Flow Action (Beta)
To easily distribute a flow to Lightning Experience or Salesforce1 users, create a flow action and add it to the appropriate page layout. Flows aren’t supported for global actions.

Available in: Lightning Experience

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

**Note:** This release contains a beta version of flow actions, which means it’s a high-quality feature with known limitations. Flow actions aren’t generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can’t guarantee general availability within any particular time frame or at all. Make your purchase decisions only based on generally available products and features. You can provide feedback and suggestions for flow actions in the IdeaExchange.

1. From the management settings for the object for which you want to create an action, go to Buttons, Links, and Actions.
2. Click **New Action**.
3. For Action Type, select **Flow (Beta)**.
4. Select the flow to use in your action.
   - The flow must be active and of type “Flow”.
5. Enter a label for the action.
   - Users see this label, rather than the flow name, as the name of the action. We recommend entering the name of the flow as the action label.
6. If necessary, change the name of the action.
   - This name is used in the API and managed packages. It must begin with a letter and use only alphanumeric characters and underscores, and it can’t end with an underscore or have two consecutive underscores. Unless you’re familiar with working with the API, we suggest not editing this field.
7. Type a description for the action.
   - The description appears on the detail page for the action and in the list on the Buttons, Links, and Actions page. The description isn’t visible to your users. If you’re creating several actions on the same object, we recommend using a detailed description.
8. Optionally, change the action icon to a static resource in your org.
   Custom images used for action icons must be less than 1 MB in size.

9. Save the action.

10. Hang on, you’re not done yet! To make your action available to your users, add it to a page layout.

Want the action to send the record’s ID to your flow? Make sure that the flow has a variable with these settings.

<table>
<thead>
<tr>
<th>Variable Setting</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unique Name</td>
<td>recordId (case sensitive)</td>
</tr>
<tr>
<td>Data Type</td>
<td>Text</td>
</tr>
<tr>
<td>Input/Output Type</td>
<td>• Input Only</td>
</tr>
<tr>
<td></td>
<td>• Input and Output</td>
</tr>
</tbody>
</table>

Note: If you delete an action, the action is removed from all layouts that it’s assigned to. If you deactivate a flow referenced in an action, the action doesn’t appear at runtime.

IN THIS SECTION:
Flow Action Considerations (Beta)
Keep these considerations in mind before using flow actions.

Flow Action Considerations (Beta)
Keep these considerations in mind before using flow actions.

Note: This release contains a beta version of flow actions, which means it’s a high-quality feature with known limitations. Flow actions aren’t generally available unless or until Salesforce announces its general availability in documentation or in press releases or public statements. We can’t guarantee general availability within any particular time frame or at all. Make your purchase decisions only based on generally available products and features. You can provide feedback and suggestions for flow actions in the IdeaExchange.

Flow Type
Flow actions support only flows that include screens. Make sure the flow that you want to use has a type “Flow”.

Flow Status
The flow must be active. If you later deactivate the flow, the action doesn’t appear at runtime.

Action Types
Flow actions are available only as object-specific actions.

Action Title
The title that appears on the modal is the action’s label instead of the flow name. We recommend entering the flow name as the action label.

Input Variables
Flow actions send the record ID to your flow automatically if the flow includes a Text input variable called recordId. The variable name is case-sensitive.
Deploying Flow Actions

Packaging and other deployment options aren’t supported for flow actions.

Add a Flow to the Utility Bar

Want your flow to be accessible from any page in your app? Add it to the utility bar in your Lightning app. The utility bar gives your users quick access to commonly used tools.

Available in: Lightning Experience

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

1. From Setup, enter App Manager in the Quick Find box, then select App Manager.
2. Edit an existing Lightning app or click New Lightning App. You can also upgrade a custom Classic app to a Lightning app.
4. Configure the utility item properties and the component properties.
   Flow
   Only active flows of type Flow are available. Flows that were built in the Desktop Flow Designer aren’t supported.
   Layout
   By default, flows display in one column.
   Input variables
   If you see other properties, they are the flow’s input variables. Variables appear only if they allow input access.
   Pass record ID into this variable
   This option is available only for Text input variables in Record pages. For simplicity, we recommend passing the ID to only one variable.
   For example, when this component is embedded in an Opportunity Record page, at runtime the component passes the opportunity’s ID into the selected input variable.
5. Save your changes.

To verify your changes, click the App Launcher and select the app that you added the flow to.
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, enter Flows in the Quick Find box, then select 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:

   https://yourDomain.my.salesforce.com/flow/MyFlowName

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

   https://yourDomain.my.salesforce.com/flow/namespace/MyFlowName

5. To set the initial values of your flow’s variables, append ?variable1=value1&variable2=value2 to the URL.
6. Distribute the flow URL.
   Here are some examples:
   • Create a custom button or link, and add it to a page layout.
   • Create a Web tab, and add it to the appropriate profiles.

IN THIS SECTION:

Choose Your Org’s Runtime Experience for URL-Based Flows
Are you distributing a flow via a URL? That includes things like direct URLs and custom buttons, as well as links in Setup. You can flip one switch to upgrade all those flows to Lightning runtime.

Render Two-Column Screens from a Flow URL
When you distribute a flow using a URL, you can control whether to display the screens with one column or two columns. Two-column screens are supported only for orgs that have enabled Lightning runtime.

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 to redirect users to another page within Salesforce when they click Finish, use the retURL parameter in the flow URL.
Choose Your Org’s Runtime Experience for URL-Based Flows

Are you distributing a flow via a URL? That includes things like direct URLs and custom buttons, as well as links in Setup. You can flip one switch to upgrade all those flows to Lightning runtime.

We have two flavors of runtime experience for your flow users. *Classic runtime* looks more like a standard Visualforce page. *Lightning runtime* fits right in with Lightning Experience. To see a comparison of the two runtime experiences, check out Flow Runtime Experiences.

To render all URL-based flows in Lightning runtime:

1. From Setup, enter Process Automation Settings in the Quick Find box, then select Process Automation Settings.
2. Select Enable Lightning Runtime for Flows.
3. Save your changes.

This setting also lets you control whether a flow displays in one or two columns if you distribute the flow via a URL or via a Lightning page.

When enabled, flows use Lightning runtime when they’re run from:

- A direct link
- A custom button or link
- The Run button in the Cloud Flow Designer
- A Run link on the flow list page
- The Run button on a flow detail page

SEE ALSO:
- Flow Runtime Experiences
- Render Two-Column Screens from a Flow URL

Render Two-Column Screens from a Flow URL

When you distribute a flow using a URL, you can control whether to display the screens with one column or two columns. Two-column screens are supported only for orgs that have enabled Lightning runtime.

Prerequisites

Enable Lightning runtime so that your flows respect the specified layout.

1. From Setup, go to Process Automation Settings.
2. Select Enable Lightning Runtime for Flows.

Format

To display a flow’s screens in two columns:

```
/flow/flowName?flowLayout=twoColumn
```
Examples
This example displays a “Case Management” flow in two columns.

/flow/Case_Management?flowLayout=twoColumn

This example displays a “User Info” flow in two columns and sets the varUserFirst and varUserLast variables (both of type Text) to the running user’s FirstName and LastName field values.

/flow/User_Info?varUserFirst={!$User.FirstName}&varUserLast={!$User.LastName}&flowLayout=twoColumn

SEE ALSO:
Choose Your Org’s Runtime Experience for URL-Based Flows
Considerations for Two-Column Flows

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 and sObject collection variables using URL parameters. If the flow uses Lightning runtime, you also can’t set the values for picklist variables, multi-select picklist variables, and collection variables of any data type.
• The variable 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 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.

Format
To set the initial value of a flow variable:

/flow/flowName?variableName=value

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

/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:

/flow/flowName?variable1Name=value1&variable2Name=value2
To set the initial values for items in a collection variable:

```
/flow/flowName?collection=value1&collection=value2
```

## Valid Values

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Acceptable Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>• Merge field of type Checkbox</td>
</tr>
<tr>
<td></td>
<td>• True values: true or 1</td>
</tr>
<tr>
<td></td>
<td>• False values: false or 0</td>
</tr>
<tr>
<td>Currency</td>
<td>Merge field of type Number or a numeric value</td>
</tr>
<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>Multi-Select</td>
<td>Merge field of any type or a string in this format: value1; value2</td>
</tr>
<tr>
<td>Picklist</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>Picklist</td>
<td>Merge field of any type or a string</td>
</tr>
<tr>
<td>Text</td>
<td>Merge field of any type or a string</td>
</tr>
</tbody>
</table>

## 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.

```
/flow/Case_Management?varID={!Case.CaseNumber}
```

The following example sets the varUserFirst and varUserLast variables (both of type Text) to the running user’s FirstName and LastName field values.

```
/flow/User_Info?varUserFirst={!$User.FirstName}&varUserLast={!$User.LastName}
```

The following example is a flow URL that is used in a custom button on a contact page layout. When a user clicks that button, the flow launches and adds text values from the contact as items in the {!collNames} text collection variable.

```
/flow/Contact_Info?collNames={!Contact.FirstName}&collNames={!Contact.LastName}
```
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 to redirect users 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:

\[/flow/flowName?retURL=url\]

where `url` is a relative URL (the part of the URL that comes after `https://yourInstance.salesforce.com/` or `https://yourInstance.lightning.force.com`).

URL Options

You can’t redirect flow users to a URL that’s external to your Salesforce org.

⚠️ Tip: Use Salesforce Classic URLs. Lightning Experience URLs always redirect to the home page in Lightning Experience.

- For Salesforce Classic URLs, Salesforce redirects your users to the right page in whichever Salesforce experience they’ve enabled — Lightning Experience or Salesforce Classic. If the page doesn’t exist in Lightning Experience, Salesforce redirects the user to the page in Salesforce Classic.
- For Lightning Experience URLs, Salesforce always redirects your users to the home page in Lightning Experience (`one/one.app`), even if the user has Salesforce Classic enabled. Users who don’t have permission to access Lightning Experience see an error message.
- If your URL redirects users to a web tab, Salesforce renders the web tab in Salesforce Classic.
- Web tabs in Lightning Experience can redirect only to Visualforce pages.

<table>
<thead>
<tr>
<th>Redirect Destination</th>
<th>Relative URL</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chatter</td>
<td>_ui/core/chatter/ui/ChatterPage</td>
<td>_ui/core/chatter/ui/ChatterPage</td>
</tr>
<tr>
<td>Home page</td>
<td>home/home.jsp</td>
<td>home/home.jsp</td>
</tr>
<tr>
<td>List view</td>
<td>objectCode?cf=listViewId</td>
<td>006?cf=00BD0000005lwec</td>
</tr>
<tr>
<td>Object home page, such as Accounts home</td>
<td>objectCode/o</td>
<td>001/o</td>
</tr>
<tr>
<td>Specific record, such as a contact, report, dashboard, user, profile, or Chatter post</td>
<td>recordId</td>
<td>0DSB0000005KZ7V</td>
</tr>
<tr>
<td>Visualforce page</td>
<td>apex/pageName</td>
<td>apex/myVisualforcePage</td>
</tr>
<tr>
<td>Web tab</td>
<td>servlet/servlet.Integration?lid=webTabId</td>
<td>servlet/servlet.Integration?lid=01rD0000000A88h</td>
</tr>
</tbody>
</table>

Limitations

- You can’t use a flow variable as the value for the `retURL` parameter. If you want to use a flow variable to redirect a user, such as to a specific record, distribute the flow by using Visualforce.
• retURL can cause nested top and side navigation bars to render on the destination page.

Examples

This flow URL redirects users to Accounts home, which exists in both Lightning Experience and Salesforce Classic.

```
/flow/myFlow?retUrl=001/o
```

When Lightning Experience users finish the flow interview, Salesforce redirects them to http://yourInstance.lightning.force.com/one/one.app#/sObject/Account/home. When Salesforce Classic users finish the flow interview, Salesforce redirects them to http://yourInstance.salesforce.com/001/o. Either way, Salesforce redirects users to Accounts home in their respective experience.

This flow URL redirects users to a Visualforce page that exists only in Salesforce Classic.

```
/flow/myFlow?retUrl=apex/myPage
```

When users finish the flow interview, Salesforce redirects them to http://yourInstance.salesforce.com/apex/myPage in Salesforce Classic. When they navigate away from the Visualforce page, Salesforce reverts to their original experience.


This flow URL sets the varUserFirst and varUserLast variables (both of type Text) to the running user’s FirstName and LastName field values. When the flow interview finishes, the user is redirected to the home page for whichever Salesforce experience is enabled.

```
/flow/User_Info?varUserFirst={!$User.FirstName}
&varUserLast={!$User.LastName}&retURL=home/home.jsp
```

SEE ALSO:

- Distribute Your Flow
- Troubleshoot Flow URLs
- Set Flow Variables with the Flow URL
Embed a Flow for Internal Users

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, enter Flows in the Quick Find box, then select Flows.
   b. Click the name of the flow.
   c. Copy the unique name of the flow.

2. From Setup, enter Visualforce Pages in the Quick Find box, then select Visualforce Pages.

3. Define a new Visualforce page, or open an existing one.

4. Add the <flow:interview> component somewhere between the <apex:page> tags.

5. Set the name attribute to the unique name of the flow.
   For example:

   ```xml
   <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 Visualforce Pages.
   b. Click Security next to your Visualforce page.
   c. Move all the appropriate profiles from Available Profiles to Enabled Profiles by using the add and remove buttons.
   d. Click Save.

8. Add the Visualforce page to your 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, 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.
Set Flow Variable Values from a Visualforce Page

After you embed your flow in a Visualforce page, 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

If you reference a variable that doesn’t allow input access, attempts to set the variable are ignored. Compilation can 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>

**Setting Variable Values without a Controller**

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

```xml
<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 by passing in data from a record. This example sets the initial value of `myVariable` to the Visualforce expression `{!account}` when the interview starts.

```xml
<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

For 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
class MyCustomController {
    public Account apexVar {get; set;}
    
    public MyCustomController() {
        apexVar = [
            SELECT Id, Name FROM Account
            WHERE Name = 'Acme'
            LIMIT 1
        ];
    }
}

<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 Acme.

```apex
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">
```
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.

```java
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);
    }
}
```

Here’s a similar example that sets the value for `accVar` to a new account when the interview starts.

```java
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

If you reference a variable that doesn’t allow output access, attempts to get the variable are ignored. Compilation can 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.

```apex
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>
```

This example uses an Apex class to get the values that are stored in a string collection variable (emailsCollVar) in the flow. Then it 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: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
<!-- The data table iterates over the variable set in the "value" attribute and sets 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" cellspacing="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:column>
        <apex:facet name="header">Employees</apex:facet>
        <apex:outputText value="{"account['NumberOfEmployees']""/>
    </apex:column>
</apex:dataTable>
```

Depending on the contents of the sObject collection variable in your flow, here’s what that data table looks like.
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 org.

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
<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.

```xml
<apex:page controller="myFlowController">
    <h1>Congratulations!</h1>
    This is your new page.
    <flow:interview name="flowname" finishLocation="{"pageA}"/>
</apex:page>
```

If you use a standard controller to display a record on the same page as the flow, users who click **Finish** start a new flow interview. They see the first screen of the flow, without the record, because the `id` query string parameter isn’t preserved in the page URL. If needed, configure the `finishLocation` to route users back to the record.

**Embed a Flow in a Custom Lightning Component**

To customize how your flow gets and receives data, add it to a custom Lightning component. Then distribute that component through a custom action, Lightning tab, or Lightning page.

To embed a flow in your Lightning component, add the `<lightning:flow>` component to it. In the JavaScript controller, identify which flow to start.

```xml
<aura:component>
    <aura:handler name="init" value="{!this}" action="{!c.init}" />
</aura:component>
```
IN THIS SECTION:

Get Flow Variable Values to a Lightning Component
Flow variable values can be displayed or referenced in a Lightning component. Once you've embedded your flow in a custom Lightning component, use the `onstatuschange` action to get values from the flow's output variables. Output variables are returned as an array.

Set Flow Variable Values from a Lightning Component
When you embed a flow in a Lightning component, give the flow more context by initializing its variables, sObject variables, collection variables, or sObject collection variables. In the component's controller, create a list of maps, then pass that list to the `startFlow` method.

Control a Flow's Finish Behavior in a Lightning Component
By default, when a flow user clicks Finish, the component starts a new interview and the user sees the first screen of the flow again. However, you can shape what happens when the flow finishes by using the `onstatuschange` action. To redirect to another page, use one of the `force:navigateTo*` events such as `force:navigateToObjectHome` or `force:navigateToUrl`.

Get Flow Variable Values to a Lightning Component
Flow variable values can be displayed or referenced in a Lightning component. Once you've embedded your flow in a custom Lightning component, use the `onstatuschange` action to get values from the flow's output variables. Output variables are returned as an array.

- **Note:** The variable must allow output access. For each flow variable, output access is controlled by:
  - The Input/Output Type variable field in the Cloud Flow Designer
  - The `isInput` field on `FlowVariable` in the Metadata API

If you reference a variable that doesn’t allow output access, attempts to get the variable are ignored.

- **Example:** This example uses the JavaScript controller to pass the flow’s `accountName` and `numberOfEmployees` variables into attributes on the component. Then, the component displays those values in output components.

```html
<aura:component>
  <aura:attribute name="accountName" type="String" />
  <aura:attribute name="numberOfEmployees" type="Decimal" />

  <p><lightning:formattedText value="{!v.accountName}" /></p>
  <p><lightning:formattedNumber style="decimal" value="{!v.numberOfEmployees}" /></p>
</aura:component>
```
<aura:handler name="init" value="{!this}" action="{!c.init}"/>
<lightning:flow aura:id="flowData" onstatuschange="{!c.handleStatusChange}" />
</aura:component>

```javascript
({
    init : function (component) {
        // Find the component whose aura:id is "flowData"
        var flow = component.find("flowData");
        // In that component, start your flow. Reference the flow's Unique Name.
        flow.startFlow("myFlow");
    },

    handleStatusChange : function (component, event) {
        if(event.getParam("status") === "FINISHED") {
            // Get the output variables and iterate over them
            var outputVariables = event.getParam("outputVariables");
            var outputVar;
            for(var i = 0; i < outputVariables.length; i++) {
                outputVar = outputVariables[i];
                // Pass the values to the component's attributes
                if(outputVar.name === "accountName") {
                    component.set("v.accountName", outputVar.value);
                } else {
                    component.set("v.numberOfEmployees", outputVar.value);
                }
            }
        }
    },
})
```

Set Flow Variable Values from a Lightning Component

When you embed a flow in a Lightning component, give the flow more context by initializing its variables, sObject variables, collection variables, or sObject collection variables. In the component’s controller, create a list of maps, then pass that list to the startFlow method.

**Note:** You can set variables only at the beginning of an interview, and the variables you set must 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

If you reference a variable that doesn’t allow input access, attempts to set the variable are ignored.

For each variable you set, provide the variable’s name, type, and value. For type, use the flow data type.

<table>
<thead>
<tr>
<th>Flow Variable Type</th>
<th>Type</th>
<th>Valid Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>String</td>
<td>String value or equivalent expression</td>
</tr>
<tr>
<td>Number</td>
<td>Number</td>
<td>Numeric value or equivalent expression</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency</td>
<td>Numeric value or equivalent expression</td>
</tr>
</tbody>
</table>
### Flow Variable Type

<table>
<thead>
<tr>
<th>Type</th>
<th>Valid Values</th>
</tr>
</thead>
</table>
| Boolean   | • True values: `true`, `1`, or equivalent expression  
             • False values: `false`, `0`, or equivalent expression |
| Picklist  | String value or equivalent expression |
| Multipicklist | String value or equivalent expression |
| Date      | "YYYY-MM-DD" or equivalent expression |
| Date/Time | "YYYY-MM-DDThh:mm:ssZ" or equivalent expression |
| sObject   | Map of key-value pairs or equivalent expression |

Example: This JavaScript controller sets values for a number variable, a date collection variable, and a couple of sObject variables.

```javascript
// Find the component whose aura:id is "flowData"
var flow = component.find("flowData");
var inputVariables = [
  { name : "numVar", type : "Number", value: 30 },
  { name : "dateColl", type : "String", value: [ "2016-10-27", "2017-08-01" ] }
],
// Sets values for fields in the account sObject variable. Id uses the value of the component's accountId attribute. Rating uses a string.
{ name : "account", type : "SObject", value: { 
  "Id" : component.get("v.accountId"),
  "Rating" : "Warm"
} },
// Set the contact sObject variable to the value of the component's contact attribute. We're assuming the attribute contains the entire sObject for a contact record.
{ name : "contact", type : "SObject", value: component.get("v.contact") }
];
flow.startFlow("myFlow", inputVariables);
```
Example: Here's an example of a component that gets an account via an Apex controller. The Apex controller passes the data to the flow's sObject variable through the JavaScript controller.

```xml
<aura:component controller="AccountController" >
    <aura:attribute name="account" type="Account" />
    <aura:handler name="init" value="{"this}" action="{"!c.init}"/>
    <lightning:flow aura:id="flowData"/>
</aura:component>

public with sharing class AccountController { 
    @AuraEnabled
    public static Account getAccount() {
        return [SELECT Id, Name, LastModifiedDate FROM Account LIMIT 1];
    }
}

{"init : function (component) {
    // Create action to find an account
    var action = component.get("c.getAccount");

    // Add callback behavior for when response is received
    action.setCallback(this, function(response) {
        if (state === "SUCCESS") {
            // Pass the account data into the component's account attribute
            component.set("v.account", response.getReturnValue());
            // Find the component whose aura:id is "flowData"
            var flow = component.find("flowData");
            // Set the account sObject variable to the value of the component's account attribute.
            var inputVariables = [
                {
                    name : "account",
                    type : "SObject",
                    value: component.get("v.account")
                }
            ];

            // In the component whose aura:id is "flowData, start your flow
            // and initialize the account sObject variable. Reference the flow's Unique Name.
            flow.startFlow("myFlow", inputVariables);
        } else {
            console.log("Failed to get account date.");
        }
    });

    // Send action to be executed
    $A.enqueueAction(action);
}
Control a Flow’s Finish Behavior in a Lightning Component

By default, when a flow user clicks Finish, the component starts a new interview and the user sees the first screen of the flow again. However, you can shape what happens when the flow finishes by using the onstatuschange action. To redirect to another page, use one of the force:navigateTo* events such as force:navigateToObjectHome or force:navigateToUrl.

Note: To control what happens when an autolaunched flow finishes, check for the FINISHED_SCREEN status.

```aura:component access="global">
    <aura:handler name="init" value="{!this}" action="{!c.init}" />
    <lightning:flow aura:id="flowData" onstatuschange="{!c.handleStatusChange}" />
</aura:component>
```

```javascript
handleStatusChange : function (component, event) {
    if(event.getParam("status") === "FINISHED") {
        // Redirect to another page in Salesforce, or
        // Redirect to a page outside of Salesforce, or
        // Show a toast, or...
    }
}
```

**Example:** This function redirects the user to a case created in the flow by using the force:navigateToSObject event.

```javascript
handleStatusChange : function (component, event) {
    if(event.getParam("status") === "FINISHED") {
        var outputVariables = event.getParam("outputVariables");
        var outputVar;
        for(var i = 0; i < outputVariables.length; i++) {
            outputVar = outputVariables[i];
            if(outputVar.name === "redirect") {
                var urlEvent = $A.get("e.force:navigateToSObject");
                urlEvent.setParams({
                    "recordId": outputVar.value,
                    "isredirect": "true"
                });
                urlEvent.fire();
            }
        }
    }
}
```

Distribute a Flow to External Users

Let external users run your flow by adding the flow to a Lightning community or external app or page. For finer control over how your flow behaves in external contexts, use a custom Lightning component or Visualforce page. Flows in custom Lightning components use Lightning runtime, and flows in Visualforce pages use Classic runtime.

For example, set up a self-service tool for your community to help visitors generate custom sales quotes.
IN THIS SECTION:

Embed a Flow in a Community
The easiest way to distribute a flow externally is with the Flow component in Community Builder. Add the Flow component to a page in your Lightning community.

Create a Custom Flow Lightning Component for External Users
To control how a flow behaves in external contexts and use Lightning runtime, create a custom Lightning component. Then distribute the component externally, such as through a Lightning community or a Lightning Out-enabled app.

Embed a Flow for External Users
Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

Embed a Flow in a Community

The easiest way to distribute a flow externally is with the Flow component in Community Builder. Add the Flow component to a page in your Lightning community.

Note:
- Users can’t resume paused flows from Lightning communities, so we recommend removing the Pause button from flows that are distributed in Lightning communities.
- Flows in Lightning communities are supported only through the Lightning Flow component.
- Flow creators can overwrite error messages with their own content.

1. Open a community in the Community Builder, then navigate to the page that you want to add the flow to.
2. From the Components panel, drag the Flow component onto the page.
3. In the property editor, configure the component.
   - **Flow**
     Only active flows of type Flow are available. Flows that were built in the Desktop Flow Designer aren’t supported.
   - **Layout**
     By default, flows display in one column.
   - **Input variables**
     If you see other properties, they are the flow’s input variables. Variables appear only if they allow input access.
   - **Pass record ID into this variable**
     This option is available only for Text input variables in Record pages. For simplicity, we recommend passing the ID to only one variable.
     For example, when this component is embedded in an Opportunity Record page, at runtime the component passes the opportunity’s ID into the selected input variable.
4. Save your page.
Create a Custom Flow Lightning Component for External Users

To control how a flow behaves in external contexts and use Lightning runtime, create a custom Lightning component. Then distribute the component externally, such as through a Lightning community or a Lightning Out-enabled app.

Create a custom Lightning component that uses `lightning:flow`. In the JavaScript controller, identify which flow to start. Mark the component with `access="global"` to make it usable outside of your org.

```xml
<aura:component access="GLOBAL">
  <aura:handler name="init" value="#{this}" action="#{c.init}" />
  <lightning:flow aura:id="flowData" />
</aura:component>

{{
  init : function (component) {
    // Find the component whose aura:id is "flowData"
    var flow = component.find("flowData");
    // In that component, start your flow. Reference the flow's Unique Name.
    flow.startFlow("myFlow");
  },
}}
```

**Lightning Communities**

To expose your custom Lightning component in a community, make sure it implements the `forceCommunity:availableForAllPageTypes` interface. Then, in the Community Builder, drag the component onto your community page.

**External App or Page**

To expose your custom Lightning component in an app outside of Salesforce servers, use Lightning Out. Create a Lightning dependency app to describe the component dependencies.

```xml
<aura:application access="GLOBAL" extends="ltng:outApp">
  <aura:dependency resource="lightning:flow"/>
</aura:application>
```

In the app or page hosting your flow, add a script to load the Lightning Out JavaScript library. Add another script to create a `lightning:flow` component.

```xml
...<script src="https://myDomain.my.salesforce.com/lightning/lightning.out.js"></script>
<script>
$L Lightning.use("c:lightningOutApp", function() {
  $Lightning.createComponent("lightning:flow", {},
    "flowContainer",
    function (component) {
      component.startFlow("Survey_customers", inputVariables);
    }
})
```
Embed a Flow for External Users

Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

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, enter Flows in the Quick Find box, then select Flows.
   b. Click the name of the flow.
   c. Copy the unique name of the flow.

2. From Setup, enter Visualforce Pages in the Quick Find box, then select Visualforce Pages.

3. Define a new Visualforce page, or open an existing one.

4. Add the `<flow:interview>` component somewhere between the `<apex:page>` tags.

5. Set the `name` attribute to the unique name of the flow.

   For example:

   ```html
   <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 Visualforce Pages.
   b. Click Security next to your Visualforce page.
   c. Move all the appropriate profiles from Available Profiles to Enabled Profiles by using the add and remove buttons.
   d. Click Save.
8. 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.

**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 flow, the latest version is always run.

**IN THIS SECTION:**

- **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.

- **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.

- **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.

- **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.

**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 “Create a Process” 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 pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

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 pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

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, enter Flow Triggers in the Quick Find box, then click Flow Triggers.

- Only active, autolaunched flows can be launched by flow triggers. However, if a flow trigger is in test mode, 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.

Tip: The pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

1. From Setup, enter Flow Triggers in the Quick Find box, then select Flow Triggers.
2. Click New Flow Trigger.
3. Select the same object as the workflow rule, and then click Next.
4. Configure the flow trigger.

<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 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>Field</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------</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 pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

Before you begin, create:

- An autolaunched flow
- A workflow rule
- A flow trigger that launches the autolaunched flow

1. From Setup, enter Workflow Rules in the Quick Find box, then select Workflow Rules.
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. Move the flow trigger to Selected Actions by using the right arrow.
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".

```
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

```
{
  "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.

```
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.

```
<apex:page controller="FlowController">
  <apex:outputLabel id="text">v1 = {!output}</apex:outputLabel>
</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**

```java
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 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**

```java
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 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:**

**Considerations for Deploying Flows with Change Sets**

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.

**Considerations for Deploying Flows with Packages**

Flows can be included in both managed and unmanaged packages. Before you deploy one, understand the limitations and behaviors of packages that contain flows.

---

**Considerations for Deploying Flows with Change Sets**

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.
  - Post to Chatter
  - Send Email
  - Submit for Approval

For example, if you deploy a flow that includes a Submit for Approval element, manually add the referenced approval process.
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 pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

SEE ALSO:
Change Sets

Considerations for Deploying Flows with Packages

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.
  - Post to Chatter
  - Send Email
  - Submit for Approval

For example, if you deploy a flow that posts to a particular Chatter group, manually add the referenced Chatter group to the package.

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

- If you register your namespace after you referenced a flow in a Visualforce page or Apex code, don’t forget to add the namespace to the flow name. Otherwise, the package will fail to install.
- If someone installs a flow from a managed package, error emails for that flow’s interviews don’t include any details about the individual flow elements. The email is sent to the user who installed the flow.
• Flow triggers aren’t packageable.
  
  The pilot program for flow trigger workflow actions is closed. If you’ve already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn’t enable the pilot in your org, use the Flows action in Process Builder instead.

• 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:
  
  Considerations for Installed Flows
  Create a Package

Why Did My Flow Interview Fail?

To troubleshoot a failed flow interview, use the flow fault email. You can also set up temporary Screen or Send Email elements to identify the problem.

IN THIS SECTION:

  Emails About Flow Errors
  Every time a flow interview fails, the admin who created the associated flow gets an email. The email includes the error message from the failure and details about every flow element that the interview executed.

  Add Temporary Elements to a Flow
  Add Screen or Send Email elements to the flow so you can check what the resources’ values are at any given time. Once you’ve solved the problem, delete temporary Screen elements.

  Troubleshoot Flow URLs
  If you’re distributing a flow and the custom button, custom link, or a direct flow URL isn’t working as expected, verify the referenced flow. In addition, verify its variables if you’re passing values into a flow from the URL.

Emails About Flow Errors

Every time a flow interview fails, the admin who created the associated flow gets an email. The email includes the error message from the failure and details about every flow element that the interview executed.

Note: This release contains a beta version of the flow error email that is production quality but has known limitations. You can provide feedback and suggestions for the flow error email on the IdeaExchange.

If the interview failed at multiple elements, the admin receives multiple emails, and the final email includes an error message for each failure. If a flow uses fault connectors, its interviews can fail at multiple elements.

Example:

An error occurred at element Apex_Plug_in_1.
List index out of bounds: 0.
An error occurred at element Fast_Delete_1.
DELETE --- There is nothing in Salesforce matching your delete criteria.

An error occurred at element Email_Alert_1.
Missing required input parameter: SObjectRowId.

IN THIS SECTION:

Limitations of Emails About Flow Errors (Beta)
The email about errors in flow interviews has some limitations for Screen, Lookup, Create, and Subflow elements—as well as some general limitations.

SEE ALSO:
Limitations of Emails About Flow Errors (Beta)
Customize What Happens When a Flow Fails
Why Did My Flow Interview Fail?

Limitations of Emails About Flow Errors (Beta)
The email about errors in flow interviews has some limitations for Screen, Lookup, Create, and Subflow elements—as well as some general limitations.

Note: This release contains a beta version of the flow error email that is production quality but has known limitations. You can provide feedback and suggestions for the flow error email on the IdeaExchange.

General
- If the user who started the flow doesn’t have a first name, null replaces the user’s first name in the “How the Interview Started” section.
- Variable assignments display in this pattern: {!variable} (prior value) = field/variable (new value). If the variable had no prior value, the parentheses display as empty. For example: {!varStatus} () = Status (Delivered)
- If you install a flow from a managed package, error emails for that flow’s interviews don’t include any details about the individual flow elements. The email is sent to the user who installed the flow.

Screen elements
Password fields display in plain text, just like if you reference a password field in a Display Text field.

Lookup elements
The email displays lookup elements as “query” elements. Record Lookup displays as Record Query, and Fast Lookup displays as Fast Query.

Subflow elements
- The merge field annotation ( {!variable} as opposed to just variable) is missing for variables in a referenced flow. For example, when an interview enters a subflow and gives details about the inputs, the subflow’s variable is subVariable instead of {!subVariable}.
- If the error occurs in a referenced flow, the email gets sent to the author of the master flow, but the subject references the name of the referenced flow.
• If you see multiple “Entered flow ReferencedFlowName version ReferencedFlowVersion” messages with no “Exited ReferencedFlowName version ReferencedFlowVersion” messages in between them, the flow user navigated backwards. To prevent this scenario, adjust the navigation options in the first screen of the referenced flow so that the user can’t click Previous.

SEE ALSO:
- Emails About Flow Errors
- Why Did My Flow Interview Fail?

Add Temporary Elements to a Flow

Add Screen or Send Email elements to the flow so you can check what the resources’ values are at any given time. Once you’ve solved the problem, delete temporary Screen elements.

1. Create a single text template that contains the values of all resources and the fault message.
2. For each fault path, use the text template to configure one of the following elements.
   - A Screen element that uses the text template in a Display Text Field. (Only if the flow’s type supports Screen elements.)
   - A Send Email element that uses the text template as the body and your email as the recipient.
   - A Post to Chatter element that uses the text template as the message. Consider creating a Chatter group specifically for flow errors.
3. Test the flow.

Example: Here’s a text template for the Calculate Discounts on Opportunities flow in the Cloud Flow Designer Workbook.

<table>
<thead>
<tr>
<th>RESOURCE VALUES for “Calculate Discounts on Opportunities”</th>
</tr>
</thead>
<tbody>
<tr>
<td>opptyID: {!opptyID}</td>
</tr>
<tr>
<td>AccountID: {!AccountID}</td>
</tr>
<tr>
<td>AccountRevenue: {!AccountRevenue}</td>
</tr>
<tr>
<td>Full_Discount outcome: {!Full_Discount}</td>
</tr>
<tr>
<td>Partial_Discount outcome: {!Partial_Discount}</td>
</tr>
<tr>
<td>Discount: {!Discount}</td>
</tr>
<tr>
<td>ERROR</td>
</tr>
<tr>
<td>{$Flow.FaultMessage}</td>
</tr>
</tbody>
</table>

After each element in the flow, add a temporary Post to Chatter element. Each Post to Chatter element is configured to use:
- The text template as the post’s message
- The “Flow Troubleshooting” Chatter group as the post’s target

Configure the Record Lookup and Record Update elements’ fault connecters so that they route to the Post to Chatter elements.
This way, the flow posts to the Chatter group after the Record Lookup, Decision, Assignment, and Record Update elements are executed. Each post provides insight into the values of each resource throughout the flow. If the flow fails, the error is included in the Chatter posts.

After you identify and fix the problem with the flow, remove the temporary elements.

SEE ALSO:

Why Did My Flow Interview Fail?

Troubleshoot Flow URLs

If you’re distributing a flow and the custom button, custom link, or a direct flow URL isn’t working as expected, verify the referenced flow. In addition, verify its variables if you’re passing values into a flow from the URL.

To make sure that the URL can find the right flow, verify that:

- The flow that the URL references hasn’t been deleted or deactivated.
- The flow name is spelled and capitalized correctly. It must be an exact, case-sensitive match to the flow’s Unique Name.
If your flow URL references a specific flow version, verify that the version hasn’t been deleted or deactivated.

If you’re using the URL to pass values into the flow and the URL can’t access the variable, the parameter that references the variable is ignored.

To make sure that the URL can find the right flow variable, verify that each variable you’re trying to pass values into:

- Is spelled and capitalized correctly. It must be an exact, case-sensitive match to the variable.
- Allows input access. The variable’s Input/Output Type must be set to “Input Only” or “Input and Output.”
- Hasn’t been renamed in the flow.
- Hasn’t been removed from the flow.
- Isn’t an sObject variable or an sObject collection variable.

In addition, make sure the value that you’re trying to pass into the variable is compatible with the variable’s data type and is correctly formatted.

SEE ALSO:
- Set Flow Finish Behavior with a Flow URL
- Set Flow Variables with the Flow URL
- Why Did My Flow Interview Fail?

**Visual Workflow Terminology**

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
- Visual Workflow
INDEX

<flow:interview>
  usage 169, 172

A
Absolute Alarm Event
  example 145
Actions
  custom actions 160
  flow actions 160
Activate session-based permission set
  send from flow 61
Activate session-based permission set element 61
Alarm events
  absolute time 132
  relative time 133
Apex
  call from a flow 48
Apex Plug-in element 62
Approvals
  submit from a flow 85
Assignment element
  adding 63
  configuring 63
  operators 112

C
Call Apex element 64
Chatter
  post from a flow 77
Choice 93
Clone records, flow
  Fast Create element 51
Cloud Flow Designer
  overview 31
  system requirements 32
Collection
  example 140, 149
Collection variable
  example 139
  populate with values 52
Components
  flow variables, get 176
  flow, finish behavior 180
  flow, output variables 176
  using 177
Conditions
  flow 47
Constant 96
Create a Flow 30
Create Service Report Element 64
Cross-object references
  in flows 107–110

D
Deactivate session-based permission set
  send from flow 61
Deactivate session-based permission set element 61
Decision element
  adding 65
  configuring 65
Dynamic Record Choice 96

E
Elements
  Apex plug-in 62
  assignment 63
  Call Apex 64
  create service report 64
  decision 65
  email alert 65
  fast create 66
  fast delete 67
  fast lookup 68
  fast update 70
  fault connector 54
  fault connector, best practice 55–56
  fault connector, default behavior 54
  fault connector, examples 57
  Generate Work Orders 70
  loop 71
  Post to Chatter 77
  quick actions 76
  record create 72
  record delete 72
  record lookup 73
  record update 75
  screen 78–81, 83
  Send Email 61, 83
  step 84
  subflow 60, 87
  Submit for Approval 85
Email
  send from flow 48, 83
Email Alert element 65
Event types, flow
  absolute time alarm 132
  platform event 131

F
Fast Create element 66
Fast Delete element 67
Fast Lookup element 68
Fast Update element 70
Fault Connector element
  best practice 55–56
  default behavior 54
  examples 57
flow
  activating 153
Flow
  absolute time alarms 132
  accessibility 29
  administration considerations 27
  building blocks 30
  call Apex 48
  call email alert workflow action 65
  Classic runtime 156
  clone record, Fast Create 51
  conditions 47
  connectors, types of 53, 111
  considerations 4
  considerations, Cloud Flow Designer 14
  considerations, large flows 15
  considerations, two-column layout 17
  creating 30
  creating records 40
  delete waiting interviews 154
  deleting 154
  deleting records 45
  delivering to users 155
  delivering to users, external 180–181, 183
  delivering to users, internal 157–159, 168, 175, 182
  design considerations 12–15, 17
  design considerations, formula 18
  differences from Process Builder 2
  differences from Workflow 2
  editing properties 138, 151
  elements overview 58
  embedding in community pages 181
  embedding in Lightning components 175, 182
Flow (continued)
  embedding in Lightning page 158–159
  embedding in Visualforce pages 168, 183
  formula limitations 18
  global variables 99
  in change sets 190
  in Lightning pages, considerations 158
  inaccessible fields 22, 25
  Invoke quick action 76
  launching from processes 184
  launching from workflow rules 185
  Lightning runtime 156
  Lightning runtime, enable 163
  limits 5–6
  looking up records 39
  managing 150
  managing connectors 37
  managing elements 37
  managing resources 37
  modifying 152
  opening 152
  operators 111
  operators, assignment 112
  operators, conditions 118
  operators, record filter 125
  passing values into 39
  pause, enable 155
  platform event 131
  process action 184
  querying records 39
  reference cross-object field values 107–110
  relative time alarms 133
  resources overview 91
  run time considerations 28
  runtime experiences 156
  saving 36
  searching 34
  send email 48, 61, 83
  setting start element 35
  settings 24, 155, 163
  sharing 155, 157–159, 168, 175, 181–183
  submit record for approval 85
  testing 153
  time-based 19
  time-based considerations 19
  translation workbench 17
  troubleshooting 192, 194
  troubleshooting, emails 192–193
  type 137
Flow (continued)
  updating records 42
  wait 88
  wait event types 130
  wait, waiting conditions 90
  workflow action 185
  workflow action considerations 185
Flow collection values
  getting 172
  setting 169
Flow condition
  operators 118
Flow connectors
  adding 37
  editing 37
  removing 37
Flow constant values
  getting 172
  setting 169
Flow Designer
  user interface 32
Flow elements
  adding 37
  editing 37
  removing 37
Flow event types
  absolute time alarm 132
  platform event 131
Flow examples
  absolute alarm event 145
  loop through a collection 140, 149
  platform event 146
  populate a String collection variable 139
  relative alarm event 141, 143
  wait for many events, all 141
  wait for many events, any 143
  wait for one event 145–146
Flow Fast Create element
  enforce field-level security 24
  ignore read-only fields 24
  read-only fields 24
Flow Fast Update element
  enforce field-level security 24
  ignore read-only fields 24
  read-only fields 24
Flow finish location
  setting via URL 163, 166
Flow formula resource
  create 98
Flow formula resource (continued)
  limitations 18
Flow interviews
  delete 154
Flow record filters
  operators 125
Flow resources
  adding 37
  editing 37
  formula, create 98
  formula, limitations 18
  removing 37
Flow screen
  validate user input 46
Flow sObject variable values
  getting 172
  setting 169
Flow trigger
  associated with workflow rule 187
  considerations 185
  defining 185
  test mode 185
Flow URL
  setting finish location 163, 166
  setting flow variable values 164
  troubleshooting 195
Flow variable values
  getting 172
  setting 169
  setting via URL 164
Flows in packages 190

G
  Generate Work Orders Element 70
  Global actionQuick action
    Invoke in flow 76
  Global constants 99
  Global variables
    in flows 99

L
  Lightning App Builder
    considerations for adding Flow components 158
    embedding flows 158–159
  Lightning communities
    embedding flows 181
  Lightning components
    embedding flows 175, 182
Limits
flow 5–6
Loop element
  adding 71
  configuring 71
  example 140, 149

M
multi-select choice fields 18

O
Object-specific action
  Invoke in flow 76
Operators
  flow assignment element 112
  flow condition 118
  flow record filters 125

P
Palette, searching 35
Picklist Choice 101
Platform Event
  example 146
Populate variable with values
  collection variable 52
Post to Chatter element 77
Process Builder
  differences from flow 2
  differences from Workflow 2

Q
Quick action element 76

R
Record Create element 72
Record Delete element 72
Record Lookup element 73
Record Update element 75
Relative Alarm Event
  example 141, 143
Resources
  choice 82, 93
  Collection variable 95
  constant 96
  dynamic record choice 82, 96
  global constants 99
  picklist choice 82, 101
  sObject collection 103
  sObject variable 104

Resources (continued)
  system variables 105
  text template 106
  variable 106
  variable in referenced flow 50

S
Saving a flow 36
Screen
  choice fields 82
  choice resources 82
Screen element
  about multi-select choice fields 18
  adding 78
  choice fields 81
  configuring output fields on the Field Settings tab 83
  configuring the Field Settings tab 80, 83
  configuring the General Info tab 79
  configuring user input fields on the Field Settings tab 80
  editing 78
  settings 79–80, 83
Searching
  a flow 34
  the Palette 35
Send Email element 83
SObject variable 104
sObject Variable, flow
  copy values to new sObject variable 25
Step element 84
Subflow element 60, 87
Submit for Approval element 85
System variables 105

T
Test mode for flow triggers 185
Text Template 106
Translation
  workbench overview 17
Troubleshooting
  flow URL 195
  flows 192
  flows, add temporary elements 194
  flows, emails about errors 192–193
  flows, fault email 192–193

V
Variable 106
Variable in referenced flow 50
Visual Workflow
  administration considerations 27
  considerations 4
  considerations, Cloud Flow Designer 14
  considerations, large flows 15
  considerations, two-column layout 17
  design considerations 12–15, 17
  design considerations, formulas 18
  design considerations, time-based flows 19
  formula limitations 18
  run time considerations 28
Visualforce
  embedding flows 168, 183
  getting flow variable values 172
  setting flow variable values 169
Voice
  create permission set 161

W
  Wait element
    example 141, 143, 145–146
  Wait event types
    flow 130
  Wait events
    relative time 133
  Wait in a flow
    configure 88
    event types 130
    waiting conditions 90
  Waiting conditions
    flow 90
Workflow
  differences from Process Builder 2
  differences from Visual Workflow 2
  flow trigger considerations 185
  flow triggers 185