

Cloud Flow Designer Guide

Version 46.0, Summer '19





© Copyright 2000–2019 salesforce.com, inc. All rights reserved. Salesforce is a registered trademark of salesforce.com, inc., as are other names and marks. Other marks appearing herein may be trademarks of their respective owners.

CONTENTS

Cloud Flow Designer Guide	1
Which Tool Should I Use?	2
Limits and Considerations	4
Limits	5
Flow Usage-Based Entitlements	2
Best Practices	3
Design Considerations	4
Management Considerations	31
Runtime Considerations	3
Accessibility	6
Create a Flow	6
Take a Tour	57
Building Blocks	8
Set the Start Element	9
Save Options	9
Common Tasks	0
Samples	6
Flow Reference	3
Elements	3
Resources	4
Cross-Object Field References	2
Connectors	6
Operators	7
Event Types	0
Types	6
Properties	8
Manage Your Flows	0
Flow and Flow Version Fields)1
Open and Modify	2
Test	2
Activate or Deactivate	3
Delete an Interview	4
Delete a Version	4
Prepare for Paused Interviews	5
Distribute Your Flows	3
Runtime Experiences	3
Internal Users	5
External Users	7
Systems	0

Contents

Other Orgs
Troubleshoot Your Flows
Error Emails
Debug Your Flows
Add Temporary Elements
Troubleshoot URLs
Terminology
Index

CLOUD FLOW DESIGNER GUIDE

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 Flow Builder.

Flow Limits and Considerations

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.

Flow Reference

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

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. To test the flow and observe what happens as it runs, use the debug option in Cloud Flow Designer. Or add temporary Screen or Send Email elements to the flow.

Flow Terminology

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Which Automation Tool Do I Use?

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

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 Flow Builder. 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 Flow Builder. 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.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Processes and flows are available in: **Essentials, Professional, Enterprise, Performance, Unlimited,** and **Developer** Editions

Approvals and Workflow are available in **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Getting Information from Users or Customers and Then Doing Something with It

If you need to build a wizard to collect information, Flow Builder 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.

	Process Builder	Flow Builder	Workflow	Approvals
Complexity	Multiple if/then statements	Complex	A single if/then statement	A single if/then statement
Visual designer	~	~		
Starts when	 Record is changed Invoked by another process Platform event message is received 	 User clicks button or link User accesses Lightning page, Community page, Visualforce page, or custom tab User accesses item in a utility bar 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	V	~	~	
Supports user interaction		~		
		Supported Actions		
Call Apex code	~	~		
Create records	*	~	Tasks only	Tasks only
Invoke processes	~			

	Process Builder	Flow Builder	Workflow	Approvals
Delete records		~		
Launch a flow	~	~		
			~	
			(Pilot) ¹	
Post to Chatter	~	~		
Send email		~		
	~		~	~
	(Email alerts only)		(Email alerts only)	(Email alerts only)
Send custom	-			
notification	•			
Send outbound messages without code			~	*
Submit for approval	~	~		
Update fields	Any related record	Any record	The record or its parent	The record or its parent

¹The 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.

Flow Limits and Considerations

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

IN THIS SECTION:

Flow Limits

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

Flow Usage-Based Entitlements

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce will contact you to discuss additions to your contract.

Flow Best Practices

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

Considerations for Designing Flows

When you design flows, keep the guidelines in mind.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Considerations for Managing Flows

When managing flows, consider the administration and activation limits.

Considerations for Running Flows

When you test or distribute a flow, keep the allocations and guidelines in mind.

Flow Accessibility

Flows are 508-compliant with a few exceptions.

SEE ALSO:

Considerations for Debugging Flows in Cloud Flow Designer Cloud Flow Designer Considerations and Limitations for Flows in Lightning Pages Considerations for Translating Flows

Flow Limits

Per-Org Limit

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

	Professional Editions	Unlimited, Performance, or Developer Editions
Versions per flow	50	50
Executed elements at runtime per flow	2,000	2,000
Active flows	5 per flow type	2,000 per flow type
Total flows	5 per flow type	4,000 per flow type
Flow interviews that are resumed or groups of scheduled actions (from processes) that are executed per hour	1,000	1,000
Total resume events (defined in flow Pause elements) or schedules (defined in processes) based on a record field value	20,000	20,000

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.

Essentials or Enterprise,

Cloud Flow Designer Guide

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:

Cloud Flow Designer Flow Limits and Considerations

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.

Per-Transaction Limit	Value
Total number of SOQL queries issued	100
(All executions of Get Records elements, and executions of Update Records or Delete Records elements that use filter conditions)	
Total number of records retrieved by SOQL queries	50,000
(All executions of Get Records elements, and executions of Update Records or Delete Records elements that use filter conditions)	
Total number of DML statements issued	150
(Create Records, Update Records, and Delete Records executions)	
Total number of records processed as a result of DML statements	10,000

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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 Pause elements span multiple transactions. A transaction ends when a flow interview pauses for an event. When the flow interview resumes, a new transaction begins. Everything after the Pause

element is executed as part of a batch transaction that includes other resumed interviews. The batch includes interviews executed by the same user ID, have the same execution time, and have the same flow version ID.

SEE ALSO:

Apex Developer Guide : Execution Governors and Limits Flow Limits

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.

Distribution Method	Transaction starts when
Process Builder ¹	A record is created or updated.
Flow URL	The URL is accessed.
Custom button or link	The button or link is clicked.
Visualforce page	The page is accessed.
Interview.start() method	If the method starts via a before or after trigger, the transaction starts when a record is created or updated. Otherwise, the transaction starts when the method (or a parent method) is invoked.

Distribution Method	Transaction starts when
	The start() method shares its limits with other operations in the transaction and other methods in the class.
REST API (Custom Actions or Flows resource)	When the REST call is made. Depending on how the REST call is implemented, the limits can be shared with other operations.

¹The same also applies if the flow is distributed through a workflow rule. The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use the Flows action in Process Builder instead.

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

When Does a Flow's Transaction End?

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

The transaction ends when:

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

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

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

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

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

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

SEE ALSO:

Apex Developer Guide : Triggers and Order of Execution

Flow Bulkification in Transactions

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

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



Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

• 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.



EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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 Usage-Based Entitlements

Like feature licenses, usage-based entitlements don't limit what you can do in Salesforce; they add to your functionality. If your usage exceeds the allowance, Salesforce will contact you to discuss additions to your contract.

To see your org's usage-based entitlements and usage from Setup, enter *Company* in the Quick Find box, then select **Company Information**. See the Usage-based Entitlements related list.

Per-Org Usage-Based Entitlement	What's Counted	Essentials and Professional Editions	Performance and Developer Editions	Enterprise and Unlimited Editions
Maximum paused and waiting flow interviews	Paused flow interviews and groups of scheduled actions from processes	30,000	50,000	50,000

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Per-Org Usage-Based Entitlement	What's Counted	Essentials and Professional Editions	Performance and Developer Editions	Enterprise and Unlimited Editions
Maximum flow interviews with UI per month	Interviews of flow types that can have screen elements, such as screen flows, user provisioning flows, Field Service Snap-in flows, contact request flows, and surveys built in Survey Builder	2,000	20,000,000	20,000,000
Maximum flow interviews without UI per month	Interviews of flow types that can't have screen elements, such as autolaunched flows, transaction security flows, and processes built in Process Builder	10,000,000	10,000,000,000	10,000,000,000

If you have questions about increasing your allocation, contact your Salesforce account executive.

SEE ALSO:

Usage-based Entitlements

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 the guidelines in mind.

Deleting Variables

If you delete an sObject variable or sObject collection variable, 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 the { !Opportunity.Probability } sObject variable, 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

- Record Delete element
- Record Lookup element
- Fast Lookup element
- Dynamic Record Choice resource

External Objects

External object records are used in, but don't trigger, flows.

- A platform event or event on external objects can kick off a flow.
- A flow can edit, create, delete, and reference external object records.

Tracking Additional Information About a Flow Interview

To store additional information about an interview when it's saved as a Salesforce record, build a custom object that references the interview's GUID. An interview is assigned an 18-character Salesforce ID only when it's paused and saved as a Salesforce record. Each interview, whether in-flight or paused, has a GUID.

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.

Considerations for Flow Stages

Before you add stages to your flow, understand how stage references and default active stages work, as well as considerations for troubleshooting stages.

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 flows.

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 Flow Limits and Considerations 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:
 - English (US)
 - French (France)
 - German (Germany)
 - Spanish (Spain)
 - Japanese (Japan)
 - Chinese (Traditional)
 - Chinese (Simplified)

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

- Has a default outcome labeled "False".

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.

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.

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.

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.
- Filter the Explorer tab contents to one type of element or resource by clicking <a>.. To remove the filter, click <a>, and select SEARCH ALL.
- 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:

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

- 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 Q, 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 *Q* and select what you want to see.
- To remove the filter, click *Q* and select SEARCH ALL.

SEE ALSO:

Tour the Cloud Flow Designer User Interface

Considerations for Flow Stages

Before you add stages to your flow, understand how stage references and default active stages work, as well as considerations for troubleshooting stages.

Stage References

When you reference a stage merge field in a display text field or other label, it resolves to the stage's label. Everywhere else, a stage merge field resolves to the stage's fully qualified name: namespace.flowName:stageName or flowName:stageName.

Whenever possible, use the stage merge field to refer to stages, such as {!myStage}. When you reference a stage in a subflow, use the fully qualified name.

Default Active Stages

When you mark a stage resource **Active by Default**, the flow automatically sets values for the system variables. Use this setting when a stage applies to every branch of the flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

At runtime, the default active stages are sorted in ascending order. How the flow uses the default active stages to update \$Flow.ActiveStages and \$Flow.CurrentStage depends on whether the flow is a master flow or a referenced flow.

Master flows

The default active stages are added to \$Flow.ActiveStages in ascending order. \$Flow.CurrentStage is set to the default active stage with the lowest order.

Referenced flows

The default active stages are inserted in <code>\$Flow.ActiveStages</code> in ascending order. <code>\$Flow.CurrentStage</code> isn't automatically updated.

• When \$Flow.CurrentStage is included in \$Flow.ActiveStages, the default active stages are inserted in \$Flow.ActiveStages after \$Flow.CurrentStage.

For example, Flow1 sets \$Flow.ActiveStages to "1,2,3,4" and \$Flow.CurrentStage to "3." It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, \$Flow.ActiveStages becomes "1, 2, 3, **A**, **B**, **C**, 4." \$Flow.CurrentStage is still "3."

• When \$Flow.CurrentStage isn't included in \$Flow.ActiveStages, the default active stages are added to the end of \$Flow.ActiveStages.

For example, Flow1 sets \$Flow.ActiveStages to "1, 2, 3, 4" and doesn't set \$Flow.CurrentStage. It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, \$Flow.ActiveStages becomes "1, 2, 3, 4, **A**, **B**, **C**." \$Flow.CurrentStage remains unset.

• When \$Flow.CurrentStage is duplicated in \$Flow.ActiveStages, the default active stages are appended after the first occurrence.

For example, Flow1 sets \$Flow.ActiveStages to "1, 2, 2, 3, 4" and \$Flow.CurrentStage to "2." It then uses a Subflow element to call Flow2. Flow2's default active stages are "A, B, C." When Flow2 starts, \$Flow.ActiveStages becomes "1, 2, **A**, **B**, **C**, 2, 3, 4." \$Flow.CurrentStage remains "2."

Troubleshooting Stages

The flow error email doesn't specify the values of \$Flow.ActiveStages and \$Flow.CurrentStage at the start of an interview. To confirm what the initial values are, add temporary elements to display the initial values, such as in a screen display text field.

SEE ALSO:

Show Users Progress Through a Flow with Stages Flow Stage Resource

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.



EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

 SCREEN INPUT FIELDS 		
User1_username		
▼ Description		
Properties	Usage	
Used by:	Coll es S D	ect_Usernam _User_1_s_I

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

- SCREENS	
💼 Collect_Userna 🥒 🖮	Q
Confirmation	-

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.

Palette	Resources	Explorer		
Drag and drop elements onto the canvas				
Q, Fir	nd in Palette			
- DRAFT	T TOOLS			
	Step			
- USER	INTERFACE			
	Screen			
- LOGIC	;			
÷	Decision			
=	Assignment			
C	Loop			
- Zz	Wait			
- DATA				
	Record Creat	e		
5	Record Upda	te	•	
Ea	Record Look	qu		
6	Record Delet	e	•	

Best Practices for Designing Flows for Translation

Review these best practices if you plan to translate flows.

- Keep your labels as short as possible. The translated label can't exceed 1,000 characters (or 255 characters for definition name and version name). If you have a long label for a display text field, consider breaking it up into multiple fields.
- When 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.
- Avoid using logic that references translated values.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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:

Flow Limits and Considerations 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

- Don't use the same choice in multiple multi-select choice fields on the same screen.

SEE ALSO:

Choice Screen Fields

Limitations for Flow Formulas

When you create a formula resource or add validation to a screen input field, understand the formula limitations in flows.

- Flow formulas can't contain more than 3,000 characters.
- A formula returns null if:
 - The value that the formula returns doesn't match its data type
 - The formula contains an unsupported function

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

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

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

• 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

• 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 50,000 interviews waiting at a given time.
- If the user who started the interview is deactivated when Salesforce tries to execute an event path, the interview fails to resume.

Transactions and Waiting Interviews

A transaction ends as soon as a flow interview begins to wait for an event. When the flow interview resumes, a new transaction begins. Everything after the Wait element is executed as part of a batch transaction that includes other resumed interviews.

Interviews aren't resumed independently. They're grouped into a single batch that starts resuming within one hour after the first interview enters the batch. Actions that execute as a result of the grouped interviews are also executed in that transaction. The batch can have other interviews that resume at the same time, have the same flow definition version ID, and are executed by the same user ID. This behavior can cause you to exceed your Apex governor limits if the resumed interview executes DML operations or SOQL queries through:

- 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 Flow Limits on page 5.

If a Wait element precedes a flow element that executes DML operations or SOQL queries:

- Ensure that your flows don't let a single user execute DML operations or SOQL queries between Wait elements that can exceed limits.
- Consider using multiple Wait elements so that the DML operations and SOQL queries are performed in multiple transactions.

EDITIONS

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

Alarm events are available in: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

Platform events are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions • 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.

Testing Events

Interviews that are waiting for platform events don't support Apex tests.

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 resumes the interview within one hour.

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

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 resumes the interview on 2/13/2014 and sends the email.
- The close date is updated to 2/10/2014 before the interview resumes. Result: Salesforce reschedules the relative time alarm and the interview resumes on 2/3/2014.
- The close date is updated to a date in the past. Result: Salesforce recalculates the relative time alarm and resumes the interview shortly after you save the record.
- If a relative time alarm references a null date field when the interview executes the Wait element, Salesforce resumes the interview within an hour.
- If a relative time alarm references a date field that's that has a non-null value when the flow interview executes the Wait element and it's updated to null before the alarm is processed, Salesforce resumes the interview within an hour after the date field is updated.
- If a waiting interview has a relative time alarm and the referenced record or object is deleted, the alarm is removed from the queue. If the interview has no other events to wait for, the interview is deleted.
- You can't archive a product or price book that's referenced in a relative or absolute time alarm in a waiting interview.
- Lead Convert Limitations
 - You can't convert a lead that has associated relative time alarms in waiting interviews.
 - If Validation and Triggers from Lead Convert is enabled, existing operations on leads after a Wait element aren't executed during lead conversion.
 - If a campaign member based on a lead is converted before a waiting interview that's associated with that record finishes, Salesforce still executes the interview.

SEE ALSO:

Platform Events Developer Guide : Considerations for Defining and Publishing Platform Events Considerations for Designing Flows Flow Limits and Considerations 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?

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

Control What Happens When a Flow Tries to Set Values for Read-Only Fields

When 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.

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.

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 Lightning Platform*. To determine which other fields aren't editable, review the running user's permissions.

If the Variable Is Populated by	The Variable Includes
A Fast Lookup element	Id and any other read-only fields that you choose to include.
An Assignment or Record Lookup element	Any read-only fields that you choose to include.
A process or a workflow rule	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.

How Did Read-Only Fields Get in My sObject Variable?

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

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

EDITIONS

Available in: both Salesforce

Classic and Lightning

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 Lightning Platform* : 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.

Linea Demoissione Needed		Experience
User Per	Available in: Essentials ,	
To eail process automation settings:		Protessional, Enterprise, Performance, Unlimited,
		and Developer Editions

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

	When Filter inaccessible fields from flow requests ${f is}$		
	Selected	Not Selected (Recommended)	
Result when the running user doesn't have edit access to all fields	The operation partially succeeds. The flow filters read-only fields out of the operation. The fields that the user can edit are updated. The fields that the user can't edit aren't updated. The flow doesn't execute the fault path.	The operation fails. No fields in the operation are updated. The flow executes the fault path if there is one.	

	When Filter inaccessible fields from flow requests $\ensuremath{\text{is}}$		
	Selected	Not Selected (Recommended)	
Notification when one or more fields aren't updated	No notification is sent to the user or admin to indicate that some fields weren't updated.	The admin receives a flow error email with full details.	
Compared to Record Create and Record Update elements	Inconsistent	Consistent	

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

1. From Setup, enter Automation in the Quick Find box, then select Process Automation Settings.

2. Select or deselect Filter inaccessible fields from flow requests.

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

- Example: 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 { ! sObjectVar2.field}, where sObjectVar2 is the name of the new variable and *field* is the field on that variable.
 - Operator—Select equals.
 - Value—Select { ! sObjectVar1.field}, where sObjectVar1 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

Sector of the se 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.

Variable	Operator	Value
{!myCaseVar_final.ld}	equals	{!myCaseVar_original.ld}
{!myCaseVar_final.IsEscalated}	equals	{!myCaseVar_original.lsEscalated}
{!myCaseVar_final.Status}	equals	{!myCaseVar_original.Status}

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.

Loop 1 C Loop Through Cases	Next element	Assignment 2 Clone sObject Variable
End of loop		
Fast Update		Assignment 3 Add New sObject to Ne

- 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.

Variable	Operator	Value
{!myCaseColl_updated}	add	{!myCaseLoopVar_final}

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 Flow Limits and Considerations

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 either the user who installed the flow or the Apex exception email recipients.
- If you install a flow from an unmanaged package that has the same name but a different version number as a flow in your 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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 Lightning Bolt Solutions, Change Sets, and Packages Considerations for Deploying Flows with Packages *ISVforce Guide*: Installing a Package Control Who Receives Flow and Process Error Emails

Considerations for Running Flows

When you test or distribute a flow, keep the allocations and guidelines in mind.

Date/Time Values

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.

Testing Flows

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

Distributing Flows

- Flow users always run the active version of a flow. As a flow admin, you run the latest version of the flow.
- 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.

Setting Input Variables

When a process or flow launches another flow, that flow's input variables can be assigned values during the launch. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of null is converted to an empty string.

Flow Interviews

A flow interview is an instance of a flow, much like a record is an instance of an object. Depending on how you've configured the corresponding flow, the flow interview can do many things, including look up and manipulate Salesforce data. In an interview, you can pass data into variables and other resources. The data can come from a variety of sources, such as Salesforce records that the flow queried, information that a user entered in a screen input field, or something you manually entered.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions 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. In addition to the Record and Fast elements, the Post to Chatter, Submit for Approval, and Quick Actions elements also create and update records.

When an interview is in flight, the data in the interview isn't saved to the Salesforce database. If the flow executes an element that creates or updates records (such as Fast Update or Post to Chatter), only the information configured in that element is saved to the Salesforce database.

When an interview executes a Wait element or is paused by a user, all the interview data is serialized and saved to the database as a Paused Flow Interview record. When the interview is resumed, the Paused Flow Interview record is deleted.

When a user session expires, in-progress flow interviews are interrupted and can't be resumed. If the flow executed actions before the session ended, such as a Record Create or Post to Chatter element, those actions aren't rolled back. But other progress through the interview, such as what the user entered on the screen, is lost.

🚺 Tip:

- Set your session timeout settings to log out users after an appropriate period of time.
- Encourage your users to pay attention during interviews for alerts about their sessions expiring soon.
- Remind users to avoid running flows during release upgrades. A typical upgrade takes about 5 minutes.

Note: Paused or waiting flow interviews aren't affected by expired user sessions.

Lightning Flow 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

In number input fields, users can't enter more than 16 digits, including digits before and after a decimal point.

IN THIS SECTION:

What Happens When a Flow Finishes?

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

SEE ALSO:

Test a Flow Flow Interviews Flow Limits and Considerations Flow Runtime Experiences

What Happens When a Flow Finishes?

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

Distribution Method	Default Finish Behavior	Override Options
URL (Direct URL, Web tab, custom button, custom link)	Starts new interview	 Add a local action to the flow Set the retURL parameter
Lightning page	Starts new interview	Add a local action to the flow
Lightning Community page	Starts new interview	Add a local action to the flow
Flow action	Closes dialog	Add a local action to the flow
Utility bar	Starts new interview	Add a local action to the flow
lightning:flow Lightning component	Starts new interview	 Add a local action to the flow Set the onstatuschange action
flow:interview Visualforce component	Starts new interview	 Add a local action to the flow Set the finishLocation attribute

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

SEE ALSO:

Redirect Flow Users with a Local Action

Set Flow Finish Behavior with a Flow URL

Control a Flow's Finish Behavior by Wrapping the Flow in a Custom Aura Component

Configure the finishLocation Attribute in a Flow

Flow Accessibility

Flows are 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:

Flow Limits and Considerations

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 a flow 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

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.
- **Run with Latest** runs the most recent save of the flow you're working in. Each Subflow element runs the latest version of the referenced flow, even if that version isn't active. This option appears only if the open flow contains a Subflow element.
- Debug lets you enter values for the flow's input variables and display debug details while running the 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, Resources, and Explorer Tabs (3)

- From the Palette tab, add new elements, like Screens and Record Creates, to your flow.
- From the Resources tab, create variables, stages, choices, and other resources to use in your flow.
- 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

Flow Building Blocks

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

		How Many Records to Create	
		One Multiple	+ New Resource
Start Enter Your Co	ontac	How to Set the Record Fields	PICKLIST VALUES
Screen		Use all values from a record variable Use separate variables, resources, and literal values	Agriculture Text
+			A _a Apparel Text
2	1	Create a Record of This Object	A _a Banking Text
Does User I	Exist	*Object Lead	A _a Biotechnology Text
Get Recor	rds	Set Field Values for the Lead	Aa Chemicals Text
		Field	Aa Text
	False	Email	← An Construction
		Field	- Text
About Your Mortg User Foun	nd? Create Lead	Industry	
Screen Decision	n Create Records	Field	Value
		Company	← {!Company}
			3
Calculate Quote Show Quo	ote		
Apex Action Screen			

- 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

Set a Flow's Start Element

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

- 1. Hover over the starting element in your flow.
- 2. Click 💿.

SEE ALSO:

Save a Flow

Save a Flow

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

Initial save

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

Quick save

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

Save As

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

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

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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 Data in a Flow

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

Validate Users' Inputs with Flow Formulas

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

Show Users Progress Through a Flow with Stages

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

Define Flow Conditions

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

Redirect Flow Users with a Local Action

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

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.

Extend Your Flow with Lightning Components

Sometimes your flow needs to do more than what the Cloud Flow Designer provides out of the box. To build a richer flow screen or perform an action without going through the server, add Lightning components to your flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Extend Your Flow with Apex

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 or 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.

Upload Files Directly from a Flow

Let users upload files from a flow by adding the forceContent:fileUpload Lightning component to a flow screen.

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.

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.

	Add	Edit	Remove
Element	Drag from the Palette tab and drop it on to the canvas.	Double-click, or hover over it and click \mathscr{P} .	Hover over it and click 💼 .
Resource	From the Resources tab, double-click.	From the Explorer tab, double-click or hover over it and click 🥜 .	From the Explorer tab, hover over it and click 💼 .
Connector	Click the node at the bottom of an element on the canvas and drag a line anywhere onto the target element.	n/a	Select it and press the DELETE key.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

SEE ALSO:

Flow Elements Flow Resources

Flow Connectors

Working with Data in a Flow

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

For each of those operation, you have at least two options to choose from in the Cloud Flow Designer. Understand the differences and decide which method 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.

Integrate with External Systems from a Flow

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

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.

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.

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

	Can store values in	To map field values to flow variables	Number of records it looks up
Record Lookup	 Variables sObject variables 	 Identify each field that you want to store. 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.
Fast Lookup	 sObject variables sObject collection variables 	 Identify the flow variable in which you want to store all field values. 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.

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.
- Section 2.1. In the section of the lookup elements. Example: Here's how you'd store a user's email and name by using each of the lookup elements.

Record Lookup	Field	Variable
	Id	{!varUserId}
	Email	{!varUserEmail}
	FirstName ▼	{IsvarUser.FirstName}
	LastName ▼	{IsvarUser.LastName}

Fast Lookup	Variable 🕯	<pre>{!UserFields}</pre>
		Assign null to the variable if no n
		Specify which of the record's fields to save
		Fields
		ld
		Email
		FirstName
		LastName

SEE ALSO: Flow Fast Lookup Element Flow Record Lookup Element Working with Data in a Flow

Integrate with External Systems from a Flow

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

Platform Events

Deliver secure and scalable custom notifications within Salesforce or from external sources by using platform events. To publish event messages from your flow, add a Record Create or Fast Create element. To subscribe to messages, add a Wait element.

Lightning Components

Connect to a database that's behind your firewall without going through the Salesforce server by calling a Lightning component's client-side controller. All Lightning components that are available as flow actions appear in the Cloud Flow Designer as Local Action elements.

External Services

Connect to any external system without writing a line of code. You tell us which endpoint and schema you want to use, and we generate Apex classes for you. The Apex classes appear in the Cloud Flow Designer. External Services supports only the Interagent hyper-schema using JSON and Swagger / Open API 2.0.

Apex

If you want more control, write your own Apex code to integrate with an external system. To make your Apex code available in the Cloud Flow Designer, use either the @InvocableMethod annotation or the Process.Plugin interface.

EDITIONS

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

Platform events, External Services, and Apex are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Lightning components are available in: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

Integration Option	Declarative	Server-Side	Client-Side	Synchronous	Asynchronous
Platform events	✓	×			✓
Lightning components			v	~	~
External Services	v	<		~	
Арех		~		~	~

SEE ALSO:

Deliver Custom Notifications with Platform EventsExternal ServicesExtend Your Flow with ApexPerform Flow Actions LocallyLightning Aura Components Developer Guide : Create Flow Local Actions Using Aura Components

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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

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.

Sector 2 (1) 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.

Record Create	Contactid
	Description (!textCaseDescription)
	Status New T
	Subject Low Customer Satisfaction
	Accountid
	You can set any field on the record, but the Record Create element doesn't know which fields are required for this object.
Fast Create	Variable * {!svarCase}
	Assumes {!svarCase} is already populated with the right fields.

Quick Action (of type Create)	Contact ID	{!varContactid}
	Subject 👻	Low Satisfaction Score
	Status	New
	Description 💌	{!textCaseDescription}
	These four fields are the only fields that you from the action layout. Contact ID is require	can set for this element, because they're the only ones available ed by the associated action layout, so it's required in this element.

SEE ALSO:

Flow Fast Create Element Working with Data in a Flow Flow Quick Action Element Flow Record Create 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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.)

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

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

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.

Record Update	Update * Opportunity v that meet the following criteria:
	Field Operator Value
	Id equals V {!varOpportunityId}
	Add Row
	Update record fields with variable, constant, input, or other values.
	Field Value
	StageName Closed Lost 🗸
	CloseDate {!\$Flow.CurrentDate}
	You can update any field on the record, but the Record Update element doesn't know which fields are required for this object.
Fast Update	Variable * {!svarOpportunity} Assumes {!svarOpportunity} is already populated with the right fields.
Quick Action (of type Update)	Close Date {!\$Flow.CurrentDate} Related Record ID {!varOpportunityId} Stage Closed - Lost
	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 Data 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

	To identify records to delete
Record Delete	In the same element, use filter criteria.
Fast Delete	In another element, populate an sObject variable or sObject collection variable with the ID of the record to be deleted.

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.

Record Delete	Delete * Quote vitat meet the following criteria:
	Field Operator Value
	OpportunityId equals {!varOpportunityId}
	Status does not equal Approved
	he flow finds all quotes that are associated with a specific opportunity and haven't been approved, and nen deletes them.

Fast Delete	Variable * {!svarQuotesUnnecessary}
	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 Data 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 m to switch between the plain text editor and the rich text editor. Using the rich text editor saves the content as HTML.

Sexample:

• Validate the format of an email address:

REGEX({!Email_Address},"[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}")

• Validate the format of a zip code:

```
REGEX({!Zipcode}, "\\d{5}(-\\d{4})?")
```

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Experience

Available in: both Salesforce Classic and Lightning

Available in: Essentials.

Professional, Enterprise, Performance, Unlimited,

and **Developer** Editions

Show Users Progress Through a Flow with Stages

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

First, define all the possible stages in your flow with stage resources. When you configure a stage, you set the stage's label, order, and whether it's active by default. Then, throughout the flow identify which of the stages are relevant to the flow user by setting stage system variables.

- The \$Flow.ActiveStages system variable identifies all the stages that are relevant to the flow's current path.
- The \$Flow.CurrentStage system variable identifies which stage the flow is at. Make sure that this stage is included in \$Flow.ActiveStages.
- Example: The Online Purchase flow includes stages for users to review their cart, enter shipping details, enter payment details, and confirm their order. The stages display at runtime using a custom Lightning component.

At this point, \$Flow.ActiveStages contains the Review Cart, Shipping Details, Payment Details, and Order Confirmation stages, and \$Flow.CurrentStage is set to Review Cart.

Online Purchase			
Review Cart	Shipping Details	Payment Details	Order Confirmation

IN THIS SECTION:

Plan the Stages in Your Flow

Before you start adding stages to your flow, plan out all the possible stages for your flow. If your flow includes decisions, you might want different stages for different branches of your flow.

Define the Stages in Your Flow

After you have identified the stages for each branch of your flow, configure the stages.

Identify the Relevant Stages in Your Flow

Throughout your flow, identify which stages are relevant to the user by assigning values to the stage system variables.

Represent Your Flow's Stages Visually

The standard flow runtime doesn't represent a flow's stages. However, you can add a custom component to visually represent the stages.

SEE ALSO:

Considerations for Flow Stages

Plan the Stages in Your Flow

Before you start adding stages to your flow, plan out all the possible stages for your flow. If your flow includes decisions, you might want different stages for different branches of your flow.

- 1. List all the possible stages for your flow.
- 2. Determine in which order the stages occur.
- 3. Identify which stages are active by default.

This step is optional, but when you identify the default active stages, you don't have to populate \$Flow.ActiveStages and \$Flow.CurrentStage with an Assignment element at the beginning of your flow.

Example: Your flow has five sections: Review Cart, Shipping Details, Billing Details, Payment Details, and Order Confirmation. The corresponding stages are in the same order.

If the user's billing details are the same as the shipping details, the flow skips the Billing Details section. The other sections are required for every permutation of the flow. So configure Review Cart, Shipping Details, Payment Details, and Order Confirmation to be active by default.

Stage	Order	Active by Default
Review Cart	1	Selected
Shipping Details	2	Selected
Billing Details	3	Not selected
Payment Details	4	Selected
Order Confirmation	5	Selected

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

SEE ALSO:

Considerations for Flow Stages

Define the Stages in Your Flow

After you have identified the stages for each branch of your flow, configure the stages.

- 1. On the Resources tab, double-click **Stage**.
- 2. Enter the stage's label and order, and specify whether it's active by default.

SEE ALSO:

Flow Stage Resource Considerations for Flow Stages

Identify the Relevant Stages in Your Flow

Throughout your flow, identify which stages are relevant to the user by assigning values to the stage system variables.

The system variable \$Flow.ActiveStages identifies the stages that are relevant to the flow's current branch. \$Flow.CurrentStage identifies which stage the flow is in. To update which stages are referenced in the system variables, use an Assignment or Subflow element.

• To add stages to \$Flow.ActiveStages, use an Assignment element with one of these operators.

Operator	Description
add	Adds stages to the end of \$Flow.ActiveStages.
add at start	Adds stages to the beginning of \$Flow.ActiveStages.

For example, this assignment row adds a stage to the end of \$Flow.ActiveStages.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

 Variable
 Operator
 Value

 {!\$Flow.ActiveStages}

 add
 {!myStage}
 (!myStage)
 (!myStage)

- To add a stage to \$Flow.ActiveStages in between two other stages, define it as a default active stage in another flow. Then use a Subflow element to call the second flow.
- To remove stages from \$Flow.ActiveStages, use an Assignment element with one of these operators.

Operator	Description
remove after first	Removes all stages after the first instance of a specified stage.
remove all	Removes all instances of the specified stages.
remove before first	Removes all stages before the first instance of a specified stage.
remove first	Removes the first instance of a specified stage.
remove position	Removes the stage at a specified position.

For example, if \$Flow.ActiveStages contains three items—stage1, stage2, stage3—this assignment removes stage2 because it's in position 2.

Variable		Operator		Value	
{!\$Flow.ActiveStages}	-	remove position	•	2	-

Or suppose that <code>\$Flow.ActiveStages</code> contains four items—stage1, stage2, stage3, stage1. This assignment results in <code>\$Flow.ActiveStages</code> containing three items—stage2, stage3, stage1.

Variable		Operator	Value
{!\$Flow.ActiveStages}	-	remove first ▼	{!stage1}

• To change what's selected as <code>\$Flow.CurrentStage</code>, use an Assignment element with the **equals** operator. Make sure that the selected stage is included in <code>\$Flow.ActiveStages</code>.

Variable	Operator	Value	
{!\$Flow.CurrentStage}	▼ equals	▼ {!myStage}	•

• To count the number of active stages and assign that number to a variable, use an Assignment element with the **equals count** operator.

Variable	Operator	Value
{!VarNum}	equals count 👻	{!\$Flow.ActiveStages}

To reference a stage in another flow, enter the fully qualified stage name: *flowName:stageName* or *namespace.flowName:stageName*. At run time, the assignment works only if a Subflow element calls the stage's flow.

SEE ALSO:

System Variables in Flows Flow Assignment Element Operators in Flow Assignment Elements Considerations for Flow Stages

Represent Your Flow's Stages Visually

The standard flow runtime doesn't represent a flow's stages. However, you can add a custom component to visually represent the stages.

To visually represent the stages, you can add a Lightning component to your flow's screens or create a custom lightning:flow component.

- Lightning component screen field—When you map a stage to a Lightning component attribute, the flow passes the stage's label into the attribute.
- **lightning:flow component**—The onstatuschange attribute in the standard lightning:flow component returns the names and labels for the flow's active stages and current stage.

SEE ALSO:

Lightning Aura Components Developer Guide : Display Flow Stages with an Aura Component Lightning Aura Components Developer Guide : Display Flow Stages By Adding a Progress Indicator to a Flow Screen

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.

Column Header	Description
Resource	Flow resource whose value you want to evaluate.
Operator	The available operators depend on the data type selected for Resource. For details, see Operators in Flow Conditions on page 177.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

Column Header	Description
Value	The Variable and Value in the same row must have compatible data types.
	Options:
	• Select an existing flow resource, such as a variable, constant, or user input.
	Select CREATE NEW to create a flow resource.
	Manually enter a literal value or merge field.
	Note: When you add or subtract a number from a date value, the date adjusts in days, not hours.

2. Identify the logic between the conditions.

Option	Description	
All conditions must be true (AND)	If one of the conditions is false, the flow evaluates the next outcome's conditions.	
One condition must be true (OR)	If one of the conditions is true, the flow immediately takes this outcome's path.	
Advanced logic	Custom logic.	
(Combination of ANDs and ORs)	When you select this option, provide the customized \texttt{Logic} by entering up to 1000 characters. Use:	
	Numbers to refer to each condition	
	• AND, OR, or NOT to identify which combination of conditions must true	
	Parentheses to group parts of the string together	
	Tip: If you enter <i>AND</i> , it's the same as if you selected All conditions must be true (AND) . If you enter <i>OR</i> , it's the same as if you selected One condition must be true (OR) . If you enter any other logic, make sure that you include a number for each condition.	
	For example, for 1 AND NOT (2 OR 3), the flow evaluates whether the first condition is true and neither the second nor third condition is true.	

SEE ALSO:

What Are Waiting Conditions? Flow Wait Element Flow Decision Element

Redirect Flow Users with a Local Action

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

Note: Local actions that fire force or lightning events might not work properly when you run the flow from:

- The Run, Run Latest, and Debug buttons in the Cloud Flow Designer
- The Run links on the flow detail or flow list pages
- Web tabs
- Custom buttons and links

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

EDITIONS

Available in: Lightning Experience

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

Example: This flow creates a lead using information entered in the Lead Info screen. Then it executes the Open Lead local action, which passes the lead ID into a Lightning component by using the Record ID attribute. The component uses a Lightning event to open the created lead.



Inputs Outputs		
Assign elements or values from your flow to the action's inputs.		
Target	Source	
Record ID 👻	{!leadId}	
Add Row		

Let's look at the Lightning component that the Local Action element calls: c:navigateToRecord.

Component Markup

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

Design Resource

The recordId attribute is declared in the design resource so that it's configurable in the Local Action element.

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

Client-Side Controller

When the Local Action element is executed, the flow calls the invoke method, which uses the force:navigateToSObject event to navigate to the created record.

```
({ invoke : function(component, event, helper) {
    // Get the record ID attribute
    var record = component.get("v.recordId");
    // Get the Lightning event that opens a record in a new tab
    var redirect = $A.get("e.force:navigateToSObject");
    // Pass the record ID to the event
    redirect.setParams({
        "recordId": record
        });
    // Open the record
    redirect.fire();
    }
    })
```

SEE ALSO:

What Happens When a Flow Finishes?

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

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

Extend Your Flow with Lightning Components

Sometimes your flow needs to do more than what the Cloud Flow Designer provides out of the box. To build a richer flow screen or perform an action without going through the server, add Lightning components to your flow.

IN THIS SECTION:

Build Rich Screens with Screen Components

Use screen components to unlock the look, feel, and functionality of your flow screens.

Perform Flow Actions Locally

When you use Lightning components as flow actions, you can perform actions in the browser rather than going through the Salesforce server.

Build Rich Screens with Screen Components

Use screen components to unlock the look, feel, and functionality of your flow screens.

Either install a screen component from an external library, such as Unofficial Flow, or have a developer build one for you. For example, with a screen component you can:

- Customize the navigation in your flow
- Display data in a table
- Let flow users choose an image or use a slider
- Use a branded header instead of the default flow header

Note: Before you add Lightning components to a flow screen, make sure that:

- The flow's type is Screen Flow.
- Users can run the flow only in Lightning runtime. For example, don't distribute the flow using a Visualforce component.
- Your org has My Domain enabled and deployed.

Once you have the appropriate screen component in your org, add it to your flow screen.

- 1. In the Cloud Flow Designer, open an existing screen or add a new Screen element to the canvas.
- 2. Click the Add a Field tab.
- 3. Double-click Lightning Component.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

General Info Add a Field Field Settings		
# Number		
Currency		
17 Date		
🛞 Date/Time		
Password		
Checkbox		
- CHOICES		
●○ Radio Buttons		
👕 Dropdown List		
✓ MULTI-SELECT CHOICES		
✓ Multi-Select Checkboxes		
Multi-Select Picklist		
- OUTPUTS		
Display Text		
+ EXTENSIONS		
✓ Lightning Component		

- 4. In the preview pane, click [Lightning Component] to open the field's settings.
- 5. For Lightning Component, select the appropriate screen component.
- 6. To pass data between the flow and the Lightning component, use the Inputs and Outputs tabs.

IN THIS SECTION:

Retain Previously Entered Values in Flow Lightning Component Screen Fields

By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. Your users can then navigate back and forth in your flows and not lose the information they entered.

SEE ALSO:

Available Screen Components Lightning Aura Components Developer Guide : Customize Flow Screens Using Aura Components Perform Flow Actions Locally

Retain Previously Entered Values in Flow Lightning Component Screen Fields

By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. Your users can then navigate back and forth in your flows and not lose the information they entered.

- In the Inputs tab for your screen component, set the attribute to any value. You can enter a literal value, leave the value blank, or set the value to a variable that's compatible with the attribute's data type.
- 2. In the Outputs tab for your screen component, map the attribute to a variable that's compatible with the attribute's data type.

For example, for the Toggle screen component, map its Value attribute to a Boolean variable. For the Email screen component, map its Value attribute to a Text variable.

When configured in both the Inputs and Outputs tabs, an attribute at runtime retains whatever value it has when the user navigates to another screen.

Example: Suppose that you use the Email screen component in an email address confirmation screen. In the Email component, the Value attribute represents the email address. Configure the Inputs tab to set the Value attribute to the value of the {!existingEmail} variable, which is set earlier in the flow to the user's email address. Configure the Outputs tab to map the Value attribute to the {!confirmedEmail} variable.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

Inputs Outputs	Inputs Outputs
Set the component attributes using values from your flow.	Set flow variables using attributes from your component.
Attribute Value Label Confirm your email v Value v {!existingEmail} v i ii Add Row	Attribute Variable Value Value Add Row

When Madison Rigsby runs the flow and sees the email confirmation screen, the Value attribute is already set to mrigsby@salesforce.com per the Inputs tab setting.

Madison changes that value to *madison.rigsby@salesforce.com* and clicks **Next**. When Madison clicks **Previous** to return to the email confirmation screen, she sees the email address value as madison.rigsby@salesforce.com.

Perform Flow Actions Locally

When you use Lightning components as flow actions, you can perform actions in the browser rather than going through the Salesforce server.

Either install a local action from an external library, such as Unofficial Flow, or have a developer build one for you. For example, with local actions, a flow can:

- Open a related article in another window or tab. •
- When the flow creates a record, open the record in another browser tab. •
- When the flow finishes, close the browser or console tab.
- Get data from a database behind your firewall.

Note: To add local actions to a flow, make sure that:

- The flow's type is Screen Flow.
- Users can run the flow only in Lightning runtime. For example, don't distribute the flow using a Visualforce component.
- Your org has My Domain enabled and deployed.

To add a local action to a flow:

- 1. From the palette in Cloud Flow Designer, drag the corresponding Local Action element onto the canvas.
- 2. To pass data between the flow and the Lightning component, use the Inputs and Outputs tabs.

SEE ALSO:

Flow Local Action Element Integrate with External Systems from a Flow Lightning Aura Components Developer Guide : Create Flow Local Actions Using Aura Components Build Rich Screens with Screen Components

Extend Your Flow with Apex

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 or 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.

	Process.Plugin Interface	@InvocableMethod Annotation
Apex data	Doesn't support:	Doesn't support:
type support	• Blob	Generic Object
	Collection	Generic sObject

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

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

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

	Process.Plugin Interface	@InvocableMethod Annotation
	sObjectTime	 Sets Maps Enums The Cloud Flow Designer doesn't support mapping an Apex method's input or output parameters to an sObject collection variable.
Bulk operations	Not supported	Supported
Element name in the Cloud Flow Designer	Class name or the value of the name property.	Class name
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.	Арех
More Details in the Lightning Platform Apex Code Developer's Guide	Passing Data to a Flow Using the Process.Plugin Interface	InvocableMethod Annotation and InvocableVariable Annotation

Example: To illustrate the difference between these two implementation methods, here are two classes that do the same thing: get an account name from a flow and return that account's ID.

This class implements the @InvocableMethod annotation.

```
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.

```
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 dropdown list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the dropdown 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.

Option	Description	
Select a Version number in the left pane.	The Inputs and Outputs tabs display the variables in the selected version of the referenced flow.	
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.	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

• Manage Flow

SEE ALSO:

Flow Subflow Element

before you activate the master flow.

When you configure subflow input and output assignments, you can specify variables from any version of the referenced flow. This way, you can develop both the master flow and referenced flow in parallel, while keeping another version of the referenced flow active for its users. When you *save* the master flow, however, the Cloud Flow Designer validates against the currently active version of the referenced flow. If that flow doesn't have an active version, the latest version is validated. If you see validation messages about variables that couldn't be found or that were configured differently in the referenced flow, you can still save the flow. Nevertheless, resolve all validation errors

Upload Files Directly from a Flow

Let users upload files from a flow by adding the forceContent:fileUpload Lightning component to a flow screen.

Note: Lightning component fields are supported only in Lightning runtime.

- 1. From a Screen element, click the Add a Field tab, and double-click Lightning Component.
- 2. In the preview pane, select [Lightning Component].
- 3. For the Lightning Component dropdown on the Field Settings tab, select forceContent:fileUpload.
- **4.** Pass values between the flow and the component by using the Inputs and Outputs tabs. For details about each attribute, hover over **i**.

Screen	×
Use screens to collect user input or display output. Customize the screen by ad	lding and configuring fields to display to the user.
General Info Add a Field Field Settings	[Screen Name]
Unique Name * uploadReceipts	uploadReceipts
Lightning Component forceContent.fileUpload 💌 i	
Inputs Outputs	
Set the component attributes using values from your flow.	
Attribute Value	
File Upload Label Upload Supporting Rev	
Related Record ID {lexpense.Id}	
Allow Multiple Files 🔻 (!\$GlobalConstant.Tru 💌 i 🕮	
Accepted Formats v .pdf, .jpg, .jpeg, .png v i	
Add Row	
ОК	Cancel

Now users can attach files during a flow on any device.

Mobile device:

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow



Desktop:

Log An Expense		
Name: Lunch Amount: 20.00 Date: 10/10/2017 Upload Supporting R	eceipts	
	Or drop files	Next
	Upload Files	
----------------------------	--------------	------
Airline.jpg 53 KB		- 0
hotel-receipt.jpg 17 KB		- 0
2 of 2 files uploaded		Done

SEE ALSO:

Flow Screen Component: File Upload Flow Screen Component: File Upload

Add Values to a Collection Variable

After you create a collection variable, populate it with values to reference throughout your flow. You can't use a Record Lookup or Fast Lookup element to populate a collection variable, but there are some workarounds.

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

To add values that are stored in	Do this	For more information
A screen field	Add the field's entered or stored value to a collection variable by using an Assignment element	Choice fieldsInput fieldsOutput fieldsAssignments
A variable	Add the variable's stored value to a collection variable by using an Assignment element	VariablesAssignments
An sObject variable	Add one of the sObject variable's stored field values to a collection variable by using an Assignment element	sObject variablesAssignments
An sObject collection variable	Loop through the sObject collection variable. Within the loop, add one of the loop	sObject collection variablesLoops

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

To add values that are stored in	Do this	For more information
	variable's stored field values to a collection variable by using an Assignment element	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.

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:

Flow Connectors Customize What Happens When a Flow Fails

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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.

Control Who Receives Flow and Process Error Emails

When a process or flow interview fails, a detailed error email is sent to the admin who last

modified the process or flow. But perhaps that admin isn't the best person to read and act on the details of what was executed and what went wrong. Instead, you can choose to send error emails to the Apex exception email recipients, which you specify and control in Setup.

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Available in: both Salesforce Classic and Lightning Experience

• 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

Control Who Receives Flow and Process Error Emails

When a process or flow interview fails, a detailed error email is sent to the admin who last modified the process or flow. But perhaps that admin isn't the best person to read and act on the details of what was executed and what went wrong. Instead, you can choose to send error emails to the Apex exception email recipients, which you specify and control in Setup.

User Permissions Needed

To adit process automation settings:	Customize Application
TO EQIL PROCESS AUTOMATION SETTINGS:	

EDITIONS

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

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

Note: Process and flow error emails include the data that's involved in the process or flow, including user-entered data.

- 1. From Setup, enter *Automation* in the Quick Find box, then select **Process Automation Settings**.
- 2. For Send Process or Flow Error Email to, select who gets the error emails.
 - User Who Last Modified the Process or Flow (default)
 - Apex Exception Email Recipients—Sends emails to the addresses set on the Apex Exception Email page in Setup. Choose this option to control who receives process or flow error emails in your org.

3. Save your changes.

SEE ALSO:

Emails About Flow Errors What Happens When an Apex Exception Occurs? What Happens When a Process Fails? 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 Survey Customers flow from the Create a Satisfaction Survey project on *Trailhead*.



Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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}.

Target	Source
Body	{!allVariableValues}
Subject	An error occurred in the Customer Satis 💌 🧵
Email Addresses (comma-separated)	flowadmin@salesforce.com 💌 i

3. From each element that can fail, draw a fault connecter 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

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

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:

Display Text Screen 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 four 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 Flows That Display Stages

These Online Purchase flows display stages as sections on a progress indicator. Each sample flow displays stages differently based on how the flow is configured.

Sample Flow That 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Fast Lookup Sa Find SF Employees	1
Loop C Loop Through Employees	Assignment Add Email to Collection
Send Email Femail SF Employees	4

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 four 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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.

Record ID	{!contract.ld}
Base Date/Time Field	EndDate
Object Type	Contract
Offset Number	
Offset Unit	Days

You want to use the same Wait element for every reminder, so after a flow interview sends one email reminder, it returns to the Wait element. But first, to ensure that the interview doesn't send the same email again and again, use *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.

Waiting Conditions			
☑ Wait for this event only if additi	onal conditions are met		
If these conditions aren't met wher waits for the other defined events. element's default path.	n the interview hits this Wait e If the conditions aren't met fo	lement, the interview doesn't wait fo r all of the defined events, the interv	r this event. Instead, it iew takes the Wait
Resource	Operator	Value	
{!twoWeeksVisited}	▼ equals	▼ {!\$GlobalConstant.Fal	se} 💌

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.

Record ID	{!contract.ld}
Base Date/Time Field	EndDate 4
Object Type	Contract
Offset Number	-7
Offset Unit	Days

For the second event, the flow checks whether the Boolean variable { !oneWeekVisited} is set to false. If it isn't, the flow doesn't wait for this event.

Cloud Flow Designer Guide

▼ Waiting Condi	tions				
🗹 Wait for this	s event only if additional con	ditions are met			
If these conditi waits for the ot element's defa	ons aren't met when the inte her defined events. If the co ult path.	rview hits this Wait nditions aren't met	t element, the for all of the o	interview doesn't wait for this ev defined events, the interview take	vent. Instead, it es the Wait
Reso	urce	Operator		Value	
{!one	WeekVisited}	equals	-	{!\$GlobalConstant.False}	-

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.



EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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).

Record ID	{!accountId}
Base Date/Time Field	Renewal_Date_c
Object Type	Account
Offset Number	-7
Offset Unit 💌	Days

The base time for the second event ("Week before contract expires") is the value of Contract.EndDate (4) on the record whose ID is stored in { ! contractId} (5). The offset is -7 days (6).

Record ID	{!contractId}
Base Date/Time Field	EndDate
Object Type	Contract
Offset Number 💌	-7
Offset Unit 💌	Days

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).

Record Update	ait	Send Email
Activate	Wait 2 1 day after activated →	Request
Contract	FAULT	feedback
Se /	nd Email Send fault email	

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).

Name * 1 day after activated					
Unique Name * X1_day_after_activated	i				
Event Type Alarm: Absolute Time	▼				
▼ Event Conditions					
Define the event that you want to wait for.					
Target	Source				
Base Time	{!\$Flow.CurrentDateTime}				
Offset Number 💌					
Offset Unit 💌	Days 🔽 🗍				



Available in: both Salesforce Classic and Lightning Experience

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.

EDITIONS

Available in both Salesforce Classic and Lightning Experience

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



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.

Name * Ink			
Unique Name Ink			i
Resource	Operator	Value	
{!inkNeeded}	▼ equals	▼ {!\$GlobalConstant.True}	-

Name * Paper			
Unique Name * Paper			i
Resource	Operator	Value	
{!paperNeeded}	▼ equals	▼ {!\$GlobalConstant.True}	-

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 lnk 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.

Serial Number	{!serialNumber} ▼
Ink Manufacturer	{!inkManufacturer}
Ink Type 💌	{!inkType}

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

	Serial Number	{!serialNumber}	•
Paper Size	-	{!paperSize}	•

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.

Order Number	-	{!orderNumber}	-

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.

Order Number 🔻	{!orderNumber}	-
Order Status 💌	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.

Vendor Response	-	{!vendorResponse}	-

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.

- { ! taskDescription } A text template that gives more details about the installation.
- { !assetOwner } Provided by the process that launches the flow
- {!assetId}—Provided by the process that launches the flow

ActivityDate 💌	{!installDate}
Description	{!taskDescription}
Ownerld 💌	{!assetOwner}
Priority	High
Status	Not Started
Subject 👻	Install ink on printer
WhatId	{!assetId}

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience



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).

Sample Flows That Display Stages

These Online Purchase flows display stages as sections on a progress indicator. Each sample flow displays stages differently based on how the flow is configured.

IN THIS SECTION:

Sample Flow That Displays Stages as Breadcrumbs

This Online Purchase flow shows visitors what parts of the flow they have completed by displaying all stages up to the current stage. This flow displays only the stages that the user has visited.

Sample Flow That Displays All the Active Stages

This Online Purchase flow shows visitors all active stages and the current stage so that they know what to expect throughout this flow.

Sample Flow That Displays Stages as Breadcrumbs

This Online Purchase flow shows visitors what parts of the flow they have completed by displaying all stages up to the current stage. This flow displays only the stages that the user has visited.

Example

This flow includes stages for users to review their cart, enter shipping details, enter billing details, enter payment details, and confirm their order. Since we're displaying the stages as breadcrumbs, only the first stage is active by default.

Stage Label	Unique Name	Order	Active by Default
Review Cart	Review_Cart	0	Yes
Shipping Details	Shipping_Details	1	No
Billing Details	Billing_Details	2	No
Payment Details	Payment_Details	3	No
Order Confirmation	Order_Confirmation	4	No



Available in: both Salesforce Classic and Lightning Experience

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

When the flow starts, Review Cart is automatically set to \$Flow.CurrentStage and is the only stage in \$Flow.ActiveStages.

Each time the flow moves to a different stage, an Assignment element resets the current stage and adds the new stage to the active stages.



Note: This sample uses a Lightning component to display the flow's stages. For details, see Represent Your Flow's Stages Visually.

The first screen displays only one active stage, which is also the user's current stage: Review Cart.



Next, the flow moves to a new stage: Shipping Details. To make sure that the active stages and current stage respect the change, the flow updates the system variables with an assignment.

Variable	Operato	r	Value	
{!\$Flow.CurrentStage}	▼ equals	•	{!Shipping_Details}	•
{!\$Flow.ActiveStages}	▼ add	•	{!Shipping_Details}	•

\$Flow.ActiveStages now contains the Review Cart and Shipping Details stages, and \$Flow.CurrentStage is set to the Shipping Details stage.

Often, a user's shipping details and billing details are the same. On the Shipping Details screen, the user can indicate that the billing address is different.



The flow uses the value of the Different Billing Address checkbox to determine where to go next. If the shipping and billing details are the same, the flow continues to the Payment Details assignment. If the billing and shipping details are different, the flow moves to the Billing Details assignment.

To make sure that the active stages and current stage respect the change, the flow updates the system variables with an assignment.

Variable		Operator		Value	
{!\$Flow.CurrentStage}	•	equals	-	{!Billing_Details}	-
{!\$Flow.ActiveStages}	•	add	•	{!Billing_Details}	•

Now \$Flow.ActiveStages contains the Review Cart, Shipping Details, and Billing Details stages, and \$Flow.CurrentStage is set to the Billing Details stage.



After the shipping and billing details are complete, the flow moves to the Payment Details stage. To make sure that the active stages and current stage respect that change, the flow updates the system variables with an assignment.

Variable		Operator		Value	
{!\$Flow.CurrentStage}	•	equals	-	{!Payment_Details}	•
{!\$Flow.ActiveStages}	•	add	•	{!Payment_Details}	•

\$Flow.ActiveStages contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different), and Payment Details stages. The \$Flow.CurrentStage system variable is set to the Payment Details stage.



Finally, the flow moves to the last stage: Order Confirmation. To make sure that the active stages and current stage respect the change, the flow updates the system variables with an assignment.

Variable		Operator		Value	
{!\$Flow.CurrentStage}	•	equals	-	{!Order_Confirmation}	•
{!\$Flow.ActiveStages}	•	add	•	{!Order_Confirmation}	•

\$Flow.ActiveStages now contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different),
Payment Details, and Order Confirmation stages. The \$Flow.CurrentStage system variable is set to the Order Confirmation stage.



Sample Flow That Displays All the Active Stages

This Online Purchase flow shows visitors all active stages and the current stage so that they know what to expect throughout this flow.

Example

This flow includes stages for users to review their cart, enter shipping details, enter billing details, enter payment details, and confirm their order. To give users an idea of the steps they'll go through in the flow, we're displaying all the applicable stages when the flow starts. Every user goes through the Review Cart, Shipping Details, Payment Details, and Order Confirmation stages, so those are all active by default.

Not all users enter billing details, because a user's shipping and billing details might be the same. To insert an optional stage in the flow's active stages, we'll create another flow and reference it by using a Subflow element

Stage Label	Unique Name	Order	Active by Default
Review Cart	Review_Cart	0	Yes
Shipping Details	Shipping_Details	1	Yes
Payment Details	Payment_Details	2	Yes
Order Confirmation	Order_Confirmation	3	Yes

When the flow starts, Review Cart is automatically set to \$Flow.CurrentStage, and \$Flow.ActiveStages contains Review Cart, Shipping Details, Payment Details, and Order Confirmation.

Each time the flow moves to a different stage, an Assignment element resets the current stage.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience



Note: This sample uses a Lightning component to display the flow's stages. For details, see Represent Your Flow's Stages Visually.

The first screen displays all active stages and the user's current stage: Review Cart.

Review Cart Shipping Details Payment Details Order Confirmation

Next, the flow moves to a new stage: Shipping Details. To make sure that the current stage respects the change, the flow updates the system variable with an assignment. <code>\$Flow.CurrentStage</code> is set to the Shipping Details stage.

ſ	Variable Operator		Operator	Value				
	{!\$Flow.CurrentStage}	¥	equals	•		{!Shipping_Details}	Ŧ	

Often, a user's shipping details and billing details are the same. On the Shipping Details screen, the user can indicate that the billing address is different.



The flow uses the value of the Different Billing Address checkbox to determine where to go next. If the shipping and billing details are the same, the flow continues to the Payment Details assignment. If the billing and shipping details are different, the flow uses a Subflow element to reference the Billing Details flow.

The Billing Details flow includes an optional stage for users to enter billing details between shipping and payment details.

Stage Label	Unique Name	Order	Active by Default
Billing Details	Billing_Details	1	Yes

When a referenced flow starts, its default active stages are automatically inserted in \$Flow.ActiveStages after the current stage.

When the Billing Details flow starts, \$Flow.CurrentStage is Shipping Details. The Billing Details stage is inserted into \$Flow.ActiveStages immediately after the current stage. Now \$Flow.ActiveStages contains the Review Cart, Shipping Details, Billing Details, Payment Details, and Order Confirmation stages.

The flow uses an assignment to set the current stage to Billing Details.

	Assignment Billing Details	Screen Billing	
Variable {!\$Flow.CurrentStage}	Operator	Value	

The \$Flow.CurrentStage system variable is set to the Billing Details stage.

~	\rangle \checkmark	Billing Details	Payment Details	Order Confirmation

After the shipping and billing details are complete, the flow moves to the Payment Details stage. To make sure that the current stage respects that change, the flow updates the system variable with an assignment.

Variable	Operator			Value	
{!\$Flow.CurrentStage}	•	equals	•	{!Payment_Details}	

The \$Flow.CurrentStage system variable is set to the Payment Details stage.



Finally, the flow moves to the last stage: Order Confirmation. To make sure that the current stage respects the change, the flow updates the system variable with an assignment.

Variable		Operator		Value	
{!\$Flow.CurrentStage}	•	equals	•	{!Order_Confirmation}	•

\$Flow.ActiveStages now contains the Review Cart, Shipping Details, Billing Details (if the billing and shipping details are different),
Payment Details, and Order Confirmation stages. The \$Flow.CurrentStage system variable is set to the Order Confirmation stage.



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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Flow Reference

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 Call Apex Element

Calls an Apex class's invocable method.

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 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 Local Action Element

Executes JavaScript by calling a Lightning component's client-side controller. For example, a local action can look up third-party data without going through Salesforce servers, automatically open another URL, or fire a toast message.

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.

Cloud Flow Designer Guide

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.

Provided Flow Screen Components

Salesforce provides several screen components that extend the types of input fields available in screens.

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 org. 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.

Field	Description	Available in: both Salesforce
Name	Helps you identify the element on the canvas.	Classic and Lightning
Unique Name	Automatically populated if empty when you fill out the Name field and press TAB. The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an	Available in: Essentials , Professional, Enterprise , Performance, Unlimited , and Developer Editions
	underscore, and not contain two consecutive underscores.	
	An a serie of the series of th	

Description Appears after you click Add Description.

Flow Activate Session-Based Permission Set Element

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

Specify a session-based permission set for activation.

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

Field	Description
Permission Set	The developer name of the permission set.
Name	This parameter accepts single-value resources of any type . That value is treated as text.
Permission Set	Optional. The permission set's namespace.
Namespace	This parameter accepts single-value resources of any type . That value is treated as text.

EDITIONS

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

💿 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:

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. Deactivate a running session-based permission set.

Field	Description
Permission Set	The developer name of the permission set.
Name	This parameter accepts single-value resources of any type . That value is treated as text.
Permission Set	Optional. The permission set's namespace.
Namespace	This parameter accepts single-value resources of any type . That value is treated as text.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

SEE ALSO:

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

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 Lightning Platform 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.

EDITIONS

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

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

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:

Extend Your Flow with Apex Customize What Happens When a Flow Fails Define the Path That a Flow Takes Cross-Object Field References in Flows

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 Extend Your Flow with Apex on page 63.

EDITIONS

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

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

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.

EDITIONS

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:

Extend Your Flow with Apex 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.		Available in: both Salesforce Classic and Lightning Experience
Column Header	Description	Available in: Essentials,
Variable	Variable whose value you want to change.	Professional, Enterprise, Performance, Unlimited
Operator	The available operators depend on the data type selected for the Variable.	and Developer Editions
Value	The Variable and Value in the same row must have compatible data types.	
	Options:	
	 Select an existing flow resource, such as a variable, constant, or user input. 	
	Select CREATE NEW to create a flow resource.	

Manually enter a literal value or merge field.

Second the flow. 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**

٠

- Define the Path That a Flow Takes
- Flow Resources
- Cross-Object Field References in Flows

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.

Field	Description
Name	Identifies the connector for this outcome on the canvas.
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Conditions	Determines whether the flow takes this outcome's path.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

Field	Description
Record ID	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.

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Variable	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.

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.

Field	Description	Avo
Variable	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.	Pro Per and
	This field accepts any sObject variable or sObject collection variable.	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 deploy or retrieve a version, you can specify the version number. For example, sampleFlow-3 specifies version 3 of the flow whose unique name is sampleFlow. If you don't specify a version number, the flow is the latest version.
- In API version 43.0 and earlier, this field included the version number. In API version 44 and later, this field no longer includes the version number.
- In API version 44.0, we recommend upgrading your flows to flow metadata file names without version numbers and discontinue using the FlowDefinition object to activate or deactivate a flow. Then use the Flow object to activate or deactivate a flow.
- If you deploy with flow definitions, the active version numbers in the flow definitions override the status fields in the flows. For example, the active version number in the flow definition is version 3, and the latest version of the flow is version 4 with the status field as Active. After you deploy your flow, the active version is version 3.

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.

Field	Description
Look up	Identifies the object whose records you want to look up.
Filter criteria	Narrows down the scope of records that the flow looks up. Specify the filter criteria for selecting the record from the database.
	Tip: 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
	Value must be compatible with the selected field's data type.
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.
Variable	The variable to contain the returned field values.
	This field accepts any sObject variable or sObject collection variable.
	• To contain the field values for the first returned record, use an sObject variable.
	• To contain the field values for all returned records, use an sObject collection variable.
Assign null to the variable if no records are found.	Sets the variable to null if no records meet the filter criteria. By default, the variable's values are left unchanged.
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.



- 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 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

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.

Field	Description
Variable	Identifies the sObject variable or collection that you want to use to update records.
	This field accepts any sObject variable or sObject collection variable.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.

Ø

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 Local Action Element

Description

Executes JavaScript by calling a Lightning component's client-side controller. For example, a local action can look up third-party data without going through Salesforce servers, automatically open another URL, or fire a toast message.

Tip: To display a Lightning component to your flow users, add it as a Lightning component screen field instead.

Pass values from the component's attributes into flow variables. When the component's controller finishes executing, control returns to the flow, and these

Component the lightning:availableForFlowActions interface.

values are assigned to the specified variables.

Set the component's attributes using values from the flow.

Select the Lightning component to use as an action. The component must implement

E	n	IT	\cap	NI	C
-	$\boldsymbol{\nu}$		IU	1 N.	2

Available in: both Salesforce Classic and Lightning Experience

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

Note: For attributes to appear in the Inputs and Outputs tab, they must be defined in the component's design resource. If an
attribute is defined in the design resource, it's available as both an input and an output.

Usage

Field

Inputs

Outputs

Lightning

When a flow executes a Local Action element, it calls the invoke method in the selected component's client-side controller. When the method finishes execution, the next element in the flow is executed.

SEE ALSO:

Extend Your Flow with Lightning Components

Integrate with External Systems from a Flow

Lightning Component Screen Fields

Lightning Aura Components Developer Guide : Create Flow Local Actions Using Aura Components

Lightning Aura Components Developer Guide : Use Aura Components with Flow

Flow Elements

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.

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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

EDITIONS

Flow Record Create Element

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

Field	Description	Available in: both Salesforce
Create	The object for which you want to create a record	Classic and Lightning
Field values	Identifies the field values for the new record. Each value must be compatible with the selected field's data type.	Available in: Essentials, Professional Enterprise
	Important: 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.	Performance, Unlimited, and Developer Editions
Variable	Assigns the ID of the new record to a variable so you can reference it later in the flow.	
	This field accepts only single-value variables of type Text.	

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.

Field	Description
Delete	Identifies the object whose records you want to delete.
Filter Criteria	Narrows down the scope of records that the flow deletes.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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

👃 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 deploy or retrieve a version, you can specify the version number. For example, sampleFlow-3 specifies version 3 of the flow whose unique name is sampleFlow. If you don't specify a version number, the flow is the latest version.
- In API version 43.0 and earlier, this field included the version number. In API version 44 and later, this field no longer includes the version number.
- In API version 44.0, we recommend upgrading your flows to flow metadata file names without version numbers and discontinue using the FlowDefinition object to activate or deactivate a flow. Then use the Flow object to activate or deactivate a flow.
- If you deploy with flow definitions, the active version numbers in the flow definitions override the status fields in the flows. For example, the active version number in the flow definition is version 3, and the latest version of the flow is version 4 with the status field as Active. After you deploy your flow, the active version is version 3.

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.

EDITIONS

🗹 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.

Field	Description	Available in: both Salesford Classic and Lightning	
Look up	Identifies the object whose records you want to look up.	Experience	
Filter criteria	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.	Available in: Essentials , Professional , Enterprise , Performance , Unlimited , and Developer Editions	
	Tip: Make sure that your filter criteria sufficiently narrows the search. The Record Lookup element returns only the first record from the filtered results.		
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.		

EDITIONS

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 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 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.

Field	Description	Available in: both Salesforce Classic and Lightning	
Update	Identifies the object whose records you want to update.	Experience	
Filter criteria	Narrows down the scope of records that the flow updates.	Available in: Essentials, Professional, Enterprise,	
	Important: Configure at least one filter, or the flow updates all the records for the object.	Performance , Unlimited , and Developer Editions	
	Value must be compatible with the selected field's data type.		
Update record fields	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.		

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.

Field	Description	Available in:
Related Record ID	Only for object-specific actions. The ID of the record from which the action executes.	Professional, Performance
	For example, the action creates a case that's associated with a given account. Assign the ID for that account to Related Record ID.	ana Develop
	This parameter accepts single-value resources of any type . That value is treated as text.	
Input Parameter	Varies for each action. The action layout determines which parameters are required. Required	
	parameters appear by default and can't be removed. If a required field has a	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Essentials, Enterprise, , Unlimited, er Editions

EDITIONS

Experience

Available in: both Salesforce Classic and Lightning

Fiel	Description	
	default or predefined value, that field is optional in object-specific and global actions in the flow. If you later remove the field's default or predefined value and you didn't set a value in the flow, the interview fails at run time.	
	The value must be compatible with the parameter.	
0	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 when the interview finishes, executes a Screen element, or executes a Wait element.	
SEE /	LSO:	
C	reate Salesforce Records from a Flow	
ι	pdate Salesforce Records from a Flow	
F	ow Elements	
C	ustomize What Happens When a Flow Fails	
C	efine the Path That a Flow Takes	
C	oss-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

Input Parameter	Description	Available in: Essentials, Professional, Enterprise,
Community	ID of a community to post to.	Performance , Unlimited , and Developer Editions
ID	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.	
Message	The text that you want to post.	
	 To mention a user or group, enter @[reference], where reference is the ID for the user or group that you want to mention. The reference can be a literal value, a merge field, or a flow resource. For example: @[{!UserId}]. 	
	 To add a topic, enter #[string], where string is the topic that you want to add. For example: #[Action Required]. 	

Input Parameter	Description	
	This parameter accepts single-value resources of any type . That value is treated as text and is limited to 10,000 characters.	
Target Name or ID	 Reference to the user, Chatter group, or record whose feed you want to post to. To post to a user's feed, enter the user's ID or Username. For example: jsmith@salesforce.com To post to a Chatter group, enter the group's Name or ID. For example: Entire Organization To post to a record, enter the record's ID. For example: 001D00000JWBDx This parameter accepts single-value resources of any type. That value is treated as text. 	
Target Type	 Required only if Target Name or ID is set to a username or a Chatter group name. The type of feed that you want to post to. Valid values are: User—If Target Name or ID is set to a user's Username, enter this value. Group—If Target Name or ID is set to a Chatter group's Name, enter this value. 	
Visibility	Specifies whether this feed item is available to community users. To display this feed item only to internal us set to <i>internalUsers</i> . Valid only if Salesforce Communities is enabled. Valid values are: <i>allUsers</i> <i>internalUsers</i> 	

Outputs

Output Parameter	Description
Feed Item ID	Assigns the created post's ID to a resource in the flow.
	This parameter accepts any single-value variables of type Text, Picklist, or Picklist (Multi-Select).

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:

General Info

Identifies whether a header or footer for a screen is available to the flow user. Help text is available from the header, and navigation buttons appear in the footer.

User Input Screen Fields

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

Choice Screen 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.

Display Text Screen Fields

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

Lightning Component Screen Fields

Displays a Lightning component, such as to let the user upload a file. If your flow screen needs more than a simple input field or display text, build a custom Lightning component to fill the gap.

SEE ALSO:

Flow Elements Define the Path That a Flow Takes

General Info

Identifies whether a header or footer for a screen is available to the flow user. Help text is available from the header, and navigation buttons appear in the footer.

Header and Footer

Field	Description
Show Header	Exposes the flow title and the help text for the screen. This option is supported for only Lightning runtime.
	If you hide the header but want to expose the help text, you can do so with a Lightning component.
Show Footer	Exposes the screen's navigation buttons. This option is supported for only Lightning runtime.
	If you hide the footer, use Lightning components to let the user navigate between screens.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Navigation Actions

The navigation actions appear as buttons if the footer is exposed. If you hide the footer but want to let the user navigate between screens, expose the actions with Lightning components.

Field	Description
Allow Finish	Exposes the Finish action if the screen is the last element in the flow. If Show Footer is enabled for the screen, this action appears as a button.
Allow Previous	Exposes the Previous action if there is a screen before this one. If Show Footer is enabled for the screen, this action appears as a 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.
Allow Pause	Exposes the Pause action if Let users pause flows is enabled in your org's Process Automation settings. If Show Footer is enabled for the screen, this action appears as a button.
Paused Confirmation Message	The message that's displayed to flow users when they pause a flow.

Help Text

Field	Description
Text box	Information for users to see when they access the screen's help text.
	This field accepts single-value resources of any type. That value is treated as text.
	If you hide the header but want to expose help text, use a Lightning component.

Usage

If you allow users to pause interviews of this flow:

- Customize the interview label.
- Enable Let users pause flows in your org's Process Automation Settings.
- Add the Paused Flow Interviews component to the Home page layout for relevant users.

SEE ALSO:

Design Home Page Layouts in Salesforce Classic

User Input Screen Fields

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

Field	Description
Label	User-friendly text that displays to the left of the field.
	To format the label, click $\overline{\mathbf{m}}$.
Unique Name	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.
Input Type	Automatically populated based on the type of input field you selected.
Default Value	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.
Scale	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.
Required	Forces users to enter a value before they can move on to the next screen.
Validate	Enables input validation on this field.
Formula Expression	Boolean formula expression that evaluates whether the user entered an acceptable value.
Error	Displays underneath the field if the user didn't enter an acceptable value.
Message	This field accepts single-value resources of any type . That value is treated as text.
Help Text	Adds i 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

EDITIONS

Choice Screen 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.

Field	Description	Available in: both Salesforce Classic and Lightning
Label	Displays to the left of the field. To format the label, click ${\scriptstyle III}$.	Experience
Unique Name	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.	Available in: Essentials , Professional, Enterprise , Performance , Unlimited , and Developer Editions
Value Data	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.	•
Туре	You can't change the value data type of multi-select choice fields; only text is supported.	
Scale	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.	
Required	Forces users to identify a choice before they progress to the next screen.	
Default Value	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.	
Choice	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.	
Help	Information that users see when they click i.	
'I'ext box	This field accepts single-value resources of any type . That value is treated as text.	

SEE ALSO:

Limitations for Multi-Select Choice Fields

Options for Choice Fields in Flow Screen Elements

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.

If you want the user to select	Use this resource
From a set of filtered records	Dynamic Record Choice
From a set of values that correspond to an existing picklist or multi-select picklist field	Picklist Choice
Something that can't be generated from a record, picklist field, or multi-select picklist field	Choice

SEE ALSO:

Flow Dynamic Record Choice Resource Flow Picklist Choice Resource Flow Choice Resource

Display Text Screen Fields

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

Field	Description
Unique Name	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.
Text box	The text to display to the flow user. Click \mathbf{m} 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.

EDITIONS

EDITIONS

Experience

Available in: both Salesforce

Classic and Lightning

Available in: Essentials,

Professional, Enterprise, Performance, Unlimited, and **Developer** Editions

Available in: both Salesforce Classic and Lightning Experience

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

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

Lightning Component Screen Fields

Displays a Lightning component, such as to let the user upload a file. If your flow screen needs more than a simple input field or display text, build a custom Lightning component to fill the gap.

Field	Description
Unique Name	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.
Lightning Component	Select the Lightning component to render in your screen. The component must implement the lightning:availableForFlowScreens interface.
Inputs	Set the component's attributes using values from the flow.
Outputs	Pass values from the component's attributes into flow variables.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Note: For attributes to appear in the Inputs or Outputs tab, they must be defined in the component's design resource.

Example:

A custom Lightning component that displays a radio button field has two attributes: Label and Selected Option. To customize the radio button field's label, use the Inputs tab to set Label to *Do you want to participate*? To determine which element to execute after this screen, use the Outputs tab to store the value for Selected Option in a flow variable. Then you use a Decision element to determine which screen to show next.

SEE ALSO:

Build Rich Screens with Screen Components Provided Flow Screen Components

Provided Flow Screen Components

Salesforce provides several screen components that extend the types of input fields available in screens.

Tip: If you need more functionality, install a custom screen component from an external library, such as LightningFlow.net, or have a developer build one for you.

IN THIS SECTION:

Flow Screen Component: Dependent Picklists

Display picklists in a flow screen in which the options for one picklist depends on the selected value of another picklist by adding the flowruntime:dependentPicklists Lightning component. This component uses the field dependencies configured in your org to restrict the options for each picklist field.

Flow Screen Component: Email

Let users enter email address values by adding the flowruntime:email Lightning component to a flow screen.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Cloud Flow Designer Guide

Flow Elements

Flow Screen Component: File Upload

Let users upload files from a flow screen by adding the forceContent:fileUpload Lightning component.

Flow Screen Component: Phone

Let users enter phone values from a flow screen by adding the flowruntime:phone Lightning component.

Flow Screen Component: Name

Let users enter multiple name values with one screen component by adding the flowruntime:name Lightning component to a flow screen. Without this component, you can use Text input fields to capture name information, but it takes a lot more configuration.

Flow Screen Component: Toggle

Let users flip a toggle in a flow screen by adding the flowruntime:toggle Lightning component.

Flow Screen Component: Slider

Let users visually specify number values in a flow screen by adding the flowruntime:slider Lightning component.

Flow Screen Component: URL

Let users enter URL values in a flow screen by adding the flowruntime:url Lightning component.

SEE ALSO:

Build Rich Screens with Screen Components

Flow Screen Component: Dependent Picklists

Display picklists in a flow screen in which the options for one picklist depends on the selected value of another picklist by adding the flowruntime:dependentPicklists Lightning component. This component uses the field dependencies configured in your org to restrict the options for each picklist field.

For information about adding screen components, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:dependentPicklists**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

Configure the Dependent Picklists Component

Use the Inputs tab to set the Dependent Picklists component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Attribute	Description
Object API Name	The API name of the object that the picklist fields belongs to. This attribute accepts single-value resources. The value is treated as text.
Picklist 1 API Name	The API name of the first picklist field. It must be the controlling field in a field dependency between Picklist 1 and Picklist 2
	This attribute accepts single-value resources . The value is treated as text.
Picklist 1 Label	The label for the first picklist field. By default, the label is "Select:". This attribute accepts single-value resources . The value is treated as text.

Attribute	Description
Picklist 1 Value	The default selection for the first picklist field. Setting this attribute from the Inputs tab pre-selects an option for the field.
	This attribute accepts single-value resources . The value is treated as text.
Picklist 2 API Name	The API name of the second picklist field. It must be the dependent field in a field dependency between Picklist 1 and Picklist 2. If you display a third picklist field, Picklist 2 must be the controlling field in a field dependency between Picklist 2 and Picklist 3.
	This attribute accepts single-value resources . The value is treated as text.
Picklist 2 Label	The label for the second picklist field. By default, the label is "Select:".
	This attribute accepts single-value resources . That value is treated as text.
Picklist 2 Value	The default selection for the second picklist field. Setting this attribute from the Inputs tab pre-selects an option for the field.
	This attribute accepts single-value resources . The value is treated as text.
Picklist 3 API Name	The API name of the third picklist field. It must be the dependent field in a field dependency between Picklist 2 and Picklist 3.
	This attribute accepts single-value resources . That value is treated as text.
Picklist 3 Label	The label for the third picklist field. By default, the label is "Select:".
	This attribute accepts single-value resources . The value is treated as text.
Picklist 3 Value	The default selection for the third picklist field. Setting this attribute from the Inputs tab pre-selects an option for the field.
	This attribute accepts single-value resources . The value is treated as text.
Read Only	If set to \$GlobalConstant.True, the user can't modify the value, but the user can copy it.
	This attribute accepts single-value Boolean resources .
Disabled	If set to \$GlobalConstant.True, the user can't modify or copy the value.
	This attribute accepts single-value Boolean resources.

Store the Dependent Picklists Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. The values are assigned when the user navigates to the next screen.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

Attribute	Description
Picklist 1 Value	What the user selected for the first picklist field.
	You can store this value in a single-value Text variable or a Text field on an sObject variable.
Picklist 2 Value	What the user selected for the second picklist field.
	You can store this value in a single-value Text variable or a Text field on an sObject variable.
Picklist 2 Value	What the user selected for the third picklist field.
	You can store this value in a single-value Text variable or a Text field on an sObject variable.

Example: In a Dinner Order flow, users select a specific dessert. Each dessert comes in different flavors, and the flavor options change based on the dessert that the user selects.

- 1. On the Order custom object, define two picklist fields: Dessert and Flavor.
- 2. Define a field dependency between Dessert and Flavor, where Dessert is the controlling picklist. Identify which Flavor options apply to each Dessert option.

Dessert:	Cake slice	Fruit crisp	Ice cream	Pie	Trifle
Flavor:	Strawberry	Strawberry	Strawberry	Strawberry	Strawberry
	Chocolate	Chocolate	Chocolate	Chocolate	Chocolate
	Vanilla	Vanilla	Vanilla	Vanilla	Vanilla
	Neopolitan	Neopolitan	Neopolitan	Neopolitan	Neopolitan
	Mint Chocolate Chip				
	Carrot	Carrot	Carrot	Carrot	Carrot
	Mixed Berry				
	Banana Cream				
	Apple	Apple	Apple	Apple	Apple
	Blackberry	Blackberry	Blackberry	Blackberry	Blackberry
	Peach	Peach	Peach	Peach	Peach
	German Chocolate	German Chocolate	German Chocolate	German Chocolate	German Chocolate
	Stone Fruit				
	Pumpkin	Pumpkin	Pumpkin	Pumpkin	Pumpkin
	Confetti	Confetti	Confetti	Confetti	Confetti

3. In your flow screen, add a Dependent Picklists screen component. Use the Inputs tab to configure the component.

Attribute	Value
Object API Name	Orderc
Picklist 1 API Name	Dessertc
Picklist 1 Label	Dessert
Picklist 2 Value	Flavorc
Picklist 2 Label	Flavor

When a user runs the flow, the options for Flavor change based on what's selected for Dessert.

Dessert	[Dessert	
Ice cream	\$	Fruit crisp	\$
Flavor	F	lavor	
✓ None		None)
Strawberry		Strawberry	P
Chocolate		Mixed Berry	
Vanilla		Apple	
Neopolitan		Blackberry	
Mint Chocolate Chip		Peach	J

SEE ALSO:

Provided Flow Screen Components Other Input Fields Define Field Dependencies

Flow Screen Component: Email

Let users enter email address values by adding the flowruntime:email Lightning component to a flow screen.

Email		
you@example.com		

For details about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:email**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Configure the Email Component

Use the Inputs tab to set the Email component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
Label	The label that appears above the email field. This attribute accepts single-value resources . The value is treated as text.
Placeholder Text	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field. This attribute accepts single-value resources . The value is treated as text.

Attribute	Description
Value	The value of the email field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	The Email component validates that the entered email address follows this pattern: <pre>local-part@domain</pre> , where domain includes at least one period (.). For example, <pre>user@example.com</pre> is valid, but <pre>user@example</pre> isn't valid. This attribute accepts single-value resources. The value is treated as text.
Required	If set to \$GlobalConstant.True, the user must enter a value.
	nis attibute accepts single-value boolean resources.
Read Only	If set to \$GlobalConstant.True, the user can't modify the value, but the user can copy it. This attribute accepts single-value Boolean resources .
Disabled	If set to \$GlobalConstant.True, the user can't modify or copy the value. This attribute accepts single-value Boolean resources .

Store the Email Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. All attributes are available in the Outputs tab, but Value is the most likely attribute you need to store. The value is assigned when the user navigates to the next screen.

To store the email address that the user entered, map the Value attribute to a flow variable.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

SEE ALSO:

Provided Flow Screen Components Other Input Fields

Flow Screen Component: File Upload

Let users upload files from a flow screen by adding the forceContent:fileUpload Lightning component.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

	Jpload Files
Airline.jpg 53 KB	Ø
hotel-receipt.jpg 17 KB	Ø
2 of 2 files uploaded	Done

For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **forceContent:fileUpload**.

Mote: Lightning component fields are supported only in Lightning runtime.

Configure the File Upload Component

Use the Inputs tab to set the File Upload component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
File Upload Label	Required. Label that appears above the upload button.
	This attribute accepts single-value resources . The value is treated as text.
Related Record ID	Required. ID of the record to associate the files with. If no value is passed, the component is disabled.
	This attribute accepts single-value resources . The value is treated as text.
Accepted Formats	Using the format . ext, enter a comma-separated list of the file extensions that the user can upload.
	This attribute accepts single-value resources . The value is treated as text.
Allow Multiple Files	If set to \$GlobalConstant.True, the user can upload multiple files.
	This attribute accepts single-value Boolean resources.
Disabled	If set to \$GlobalConstant.True, the component is disabled.
	This attribute accepts single-value Boolean resources.
Hover Text	Tooltip that appears when the user hovers over the component.
	This attribute accepts single-value resources . The value is treated as text.

Store the File Upload Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. The value is assigned when the user navigates to the next screen.

Attribute	Description
Content Document IDs	The IDs of the uploaded files.
	You can store this value in a Text collection variable .
Upload File Names	The names of the uploaded files.
	You can store this value in a Text collection variable .

File Upload Limits

By default, you can upload up to 10 files simultaneously, unless Salesforce has changed that limit. The org limit for the number of files simultaneously uploaded is 25 files with a minimum of one file. The maximum file size you can upload is 2 GB. In Communities, the file size limits and types allowed follow the settings determined by community file moderation. Guest users can't upload files.

SEE ALSO:

Provided Flow Screen Components Other Input Fields Upload Files Directly from a Flow

Flow Screen Component: Phone

Let users enter phone values from a flow screen by adding the flowruntime:phone Lightning component.

Cell Phone		
(555) 555-5555		

For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:phone**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Configure the Phone Component

Use the Inputs tab to set the phone component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
Label	The label that appears above the phone field. This attribute accepts single-value resources . The value is treated as text.
Placeholder Text	Text that appears in the field when it's empty. Use placeholder text to give users a hint about what to enter in the field.
	רווא מנווטעניפ מכנפענא אוווקופ-ימועפ רפאטערכפא . רוופ ימועפ וא נופמנפט מא נפאנ.

Attribute	Description
Value	The value of the phone field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Required	If set to \$GlobalConstant.True, the user must enter a value.
	This attribute accepts single-value Boolean resources .
Pattern	Determines whether the value is valid. By default, there is no pattern.
	To require the user to enter a value in a specific format, use a regular expression. Make sure that your regular expression checks for a valid phone number.
	This example expression validates that the phone number is in the 555-555-5555 format.
	$[0-9] \{3\} - [0-9] \{3\} - [0-9] \{4\}$
	This attribute accepts single-value resources . The value is treated as text.
Read Only	If set to \$GlobalConstant.True, the user can't modify the value, but the user can copy it.
	This attribute accepts single-value Boolean resources.

Store the Phone Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. All attributes are available in the Outputs tab, but Value is the most likely attribute you need to store. The value is assigned when the user navigates to the next screen.

To store the phone number that the user entered, map the Value attribute to a flow variable.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

SEE ALSO: Provided Flow Screen Components Other Input Fields

Flow Screen Component: Name

Let users enter multiple name values with one screen component by adding the flowruntime:name Lightning component to a flow screen. Without this component, you can use Text input fields to capture name information, but it takes a lot more configuration.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Name	
Salutation	
None	•
First Name	
First Name	
Middle Name	
Middle Name	
Last Name	
Last Name	
Suffix	
Suffix	
Informal Name	
Informal Name	

For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:name**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

Configure the Name Component

Use the Inputs tab to set the Name component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description		
Label	The label that appears above the name fields.		
	This attribute accepts single-value resources . The value is treated as text.		
Fields to Display	By default, the component displays only the First Name and Last Name fields, but other fields are available. To customize which fields to display at runtime, set this attribute to a comma-separated list of the field names.		
	For First Name, use firstName		
	For Last Name, use lastName		
	For Middle Name, use middleName		
	For Informal Name, use informalName		
	For Salutation, use salutation		
	For Suffix, use suffix		
	Note: This attribute doesn't control the order that the fields display in.		
	For example, to display all the fields, set this attribute to firstName, lastName, middleName, informalName, salutation, suffix.		
	This attribute accepts single-value resources . The value is treated as text.		
Salutation Options	By default, the options for Salutation are Mr., Mrs., and Ms. To override these options, set this attribute to a comma-separated list of values.		
	This attribute accepts single-value resources . The value is treated as text.		

Attribute	Description
First Name	The value of the First Name field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Middle Name	The value of the Middle Name field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Last Name	The value of the Last Name field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Informal Name	The value of the Informal Name field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Salutation	The value of the Salutation field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Suffix	The value of the Suffix field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Read Only	If set to \$GlobalConstant.True, the user can't modify the value, but the user can copy it.
	This attribute accepts single-value Boolean resources.
Disabled	If set to \$GlobalConstant.True, the user can't modify or copy the value.
	This attribute accepts single-value Boolean resources.

Store the Name Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. The values are assigned when the user navigates to the next screen.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

Attribute	Description
First Name	What the user entered in the First Name field.
	This value can be stored in a single-value Text variable or a Text field on an sObject variable.

Attribute	Description
Last Name	What the user entered in the Last Name field. This value can be stored in a single-value Text variable or a Text field on an sObject variable .
Informal Name	What the user entered in the Informal Name field. This value can be stored in a single-value Text variable or a Text field on an sObject variable .
Middle Name	What the user entered in the Middle Name field. This value can be stored in a single-value Text variable or a Text field on an sObject variable .
Salutation	What the user entered in the Salutation field. This value can be stored in a single-value Text variable or a Text field on an sObject variable .
Suffix	What the user entered in the Suffix field. This value can be stored in a single-value Text variable or a Text field on an sObject variable .

SEE ALSO:

Provided Flow Screen Components Other Input Fields

Flow Screen Component: Toggle

Let users flip a toggle in a flow screen by adding the flowruntime:toggle Lightning component.



For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:toggle**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Configure the Toggle Component

Use the Inputs tab to set the Toggle component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
Label	This label appears next to the toggle and describes what the user is enabling.
	This attribute accepts single-value resources . The value is treated as text.

Attribute	Description
Active Label	When the toggle is active, this label appears underneath the toggle. Use it to clarify what active means. The default label is "Active."
	This attribute accepts single-value resources . The value is treated as text.
Inactive Label	When the toggle is inactive, this label appears underneath the toggle. Use it to clarify what inactive means. The default label is "Inactive."
	This attribute accepts single-value resources . The value is treated as text.
Value	Whether the toggle is active (<i>\$GlobalConstant.True</i>) or inactive (<i>\$GlobalConstant.False</i>). Setting this attribute from the Inputs tab controls the default state of the toggle. To store the user's selection in a flow variable, set this attribute from the Outputs tab. This parameter accepts single-value Boolean resources .
Disabled	If set to \$GlobalConstant.True, the user can't modify or copy the value. This attribute accepts single-value Boolean resources .

Store the Toggle Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. All attributes are available in the Outputs tab, but Value is the most likely attribute you need to store. The value is assigned when the user navigates to the next screen.

To store the user's selection, map the Value attribute to a Boolean flow variable or a checkbox field on an sObject variable.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

SEE ALSO: Provided Flow Screen Components

Other Input Fields

Flow Screen Component: Slider

Let users visually specify number values in a flow screen by adding the flowruntime:slider Lightning component.

Rate your experience How satisfied are you with our service? (10 means you love	it)
0-10 7	



Available in: both Salesforce Classic and Lightning Experience

For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:slider**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

Configure the Slider Component

Use the Inputs tab to set the Slider component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
Label	This label appears above the slider.
	This attribute accepts single-value resources . The value is treated as text.
Range Minimum	The minimum value of the slider range. The default is 0.
	This parameter accepts Number resources .
Range Maximum	The maximum value of the slider range. The default is 100.
	This parameter accepts single-value Number resources .
Step Size	Divides the slider into a set of steps. The default is 1.
	For example, for a range of 0–100, set the Step Size to 10 to let the user select every 10th value. Other example step sizes are 0.1 and 5.
	This parameter accepts single-value Number resources .
Slider Size	Controls the size of the slider. The accepted values are x-small, small, medium, or large.
	This parameter accepts single-value resources of any type . That value is treated as text.
Value	The default value represented by the slider position. Setting this attribute from the Inputs tab pre-sets the value.
	This parameter accepts single-value Number resources .

Store the Slider Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. All attributes are available in the Outputs tab, but Value is the most likely attribute you need to store. The value is assigned when the user navigates to the next screen.

To store the value that the user selected, map the Value attribute to a Number flow variable.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to

both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

SEE ALSO:

Provided Flow Screen Components Other Input Fields

Flow Screen Component: URL

Let users enter URL values in a flow screen by adding the flowruntime:url Lightning component.

Vanity URL			
http://			
			_

For information about adding screen components to your flow screen, see Build Rich Screens with Screen Components. For the Lightning Component field, select **flowruntime:url**.

Note: Lightning component fields are supported only in Lightning runtime on page 213.

Configure the URL Component

Use the Inputs tab to set the URL component's attributes. You can select resources from the flow, such as variables or global constants, or you can manually enter a value.

Attribute	Description
Label	The label that appears above the URL field.
	This attribute accepts single-value resources . The value is treated as text.
Value	The value of the URL field. Setting this attribute on the Inputs tab prepopulates the field. To store the user-entered value in a flow variable, set this attribute from the Outputs tab.
	This attribute accepts single-value resources . The value is treated as text.
Required	If set to \$GlobalConstant.True, the user must enter a value.
	This attribute accepts single-value Boolean resources.
Pattern	Determines whether the value is valid. The default pattern verifies that the first character is a letter and that the value includes a colon (:).
	To force the user to enter a value in a specific format, use a regular expression. Make sure that your regular expression checks for a valid protocol in the URL, such as https:// or file:///.
	This example expression checks for a secure HTTP protocol (https://) and a specific domain (acmewireless.com).
	<pre>^https?://(?:www\.)?acmewireless\.com/?.*</pre>
	This attribute accepts single-value resources . The value is treated as text.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Attribute	Description
Read Only	If set to \$GlobalConstant.True, the user can't modify the value, but the user can copy it.
	This attribute accepts single-value Boolean resources.
Disabled	If set to \$GlobalConstant.True, the user can't modify or copy the value.
	This attribute accepts single-value Boolean resources.

Store the URL Component's Values in the Flow

Use the Outputs tab to store values from the component in flow variables. All attributes are available in the Outputs tab, but Value is the most likely attribute you need to store. The value is assigned when the user navigates to the next screen.

To store the URL that the user entered, map the Value attribute to a flow variable.

Tip: By default, screen components have no memory. If a user enters a value, navigates to another screen, and returns to the component's screen, the user-entered value is lost. To enable a flow to remember the value of an attribute, add the attribute to both the Inputs and Outputs tabs in the screen component. For details, see Retain Previously Entered Values in Flow Lightning Component Screen Fields.

SEE ALSO:

Provided Flow Screen Components Other Input Fields StackOverflow: Sample Regular Expressions for Valid URLs MDN: What is a URL?

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.

Field	Description
Body	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.
Subject	Text for the subject of the email.
	This parameter accepts single-value resources of any type . That value is treated as text.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Email Addresses (comma-separated)	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 single-value resources of any type . That value is treated as text.
Email Addresses	Optional. Recipients of the email.
(collection)	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 <i>OrgWideEmailAddress</i> .
	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: <i>CurrentUser</i>—Email address of the user running the flow. (Default) <i>DefaultWorkflowUser</i>—Email address of the default workflow user. <i>OrgWideEmailAddress</i>—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: M

- 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.

Input Parameter	Description
Record ID	The ID of the record that you want to submit for approval.
	This parameter accepts single-value resources of any type . That value is treated as text.
Next Approver IDs	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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Input Parameter	Description
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
	This parameter accepts single-value resources of any type . That value is treated as text.
Skip Entry Criteria	If set to <i>\$GlobalConstant.True</i> , the record isn't evaluated against the entry criteria set on the process that is defined in Approval Process Name or ID.
	This parameter accepts any single-value resource of type Boolean.
Submission Comments	Text that you want to accompany the submission. Don't reference merge fields or formula expressions.
	Submission comments appear in the approval history for the specified record. This text also appears in the initial approval request email if the template uses the { !ApprovalRequest.Comments } merge field.
	This parameter accepts single-value resources of any type . That value is treated as text.
Submitter ID	The ID for the user who submitted the record for approval. The user receives notifications about responses to the approval request.
	The user must be one of the allowed submitters for the process.
	If you don't set this field, the user who launched the flow is the submitter. If a workflow rule triggers a flow that includes this element, the submitter is the user who triggered the workflow rule. Workflow rules can be triggered when a user creates or edits a record. When the record is approved or rejected, the user who launched the flow or triggered the workflow rule is notified.
	This parameter accepts single-value resources of any type . That value is treated as text.

Outputs

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

Optional Output Parameter	Description
Instance ID	The ID of the approval request that was submitted.
	This parameter accepts single-value variables of type Text, Picklist, or Picklist (Multi-Select).

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

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 org. 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. However, for a text, picklist, or multi-select picklist variable that isn't a collection, a value of null is converted to an empty string in the referenced flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Column Header	Description
Target	Referenced flow's variable whose value you want to set.
	By default, this dropdown list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the dropdown list contains the variables of the <i>latest</i> version of the referenced flow.
Source	Master flow's resource or value to assign to the target.
	Options:
	• Select an existing flow resource, such as a variable, constant, or user input.
	Select CREATE NEW to create a flow resource.
	Manually enter a literal value or merge field.
	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.

Column Header	Description
Source	Referenced flow's variable whose value you want to assign to the target.
	By default, this dropdown list contains the variables of the currently active version of the referenced flow. If the referenced flow has no active version, the dropdown list contains the variables of the <i>latest</i> version of the referenced flow. 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.

SEE ALSO:

Flow Elements View Inputs and Outputs of Other Referenced Flow Versions Customize What Happens When a Flow Fails Define the Path That a Flow Takes Cross-Object Field References in Flows

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.

Field	Description
Name	The name appears on the connector that's associated with this event.
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Event Type	 Determines whether the flow is waiting for: An absolute time (such as 3 days from now), or A time relative to a date/time field on a Salesforce record (such as 3 days after an opportunity closes)
Event Conditions	Determines the exact event that the flow is waiting for. Parameters vary, based on the selected event type.
Wait for this event only if additional conditions are met	If selected, the flow waits for this event only when certain conditions are met.
Waiting Conditions	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 56.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Variable Assignments	Assigns the event's outputs to flow variables. Parameters vary, based on the selected event type.
[Default Path]	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.

Usage



- 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 24 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 78.

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."

▼ Waiting	Conditions		
If these If these Instead the Wai	t for this event only if additional cond conditions aren't met when the inte , it waits for the other defined event t element's default path.	litions are met view hits this Wait element, the s. If the conditions aren't met fo	e interview doesn't wait for this event. r all of the defined events, the interview takes
	Resource	Operator	Value
	{!account.Rating}	equals 🔻	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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

• 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 78.

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 dropdown list.

Flow Resource	Description	Creatable from the Resources Tab
Choice	Represents an individual value that can be used in choice screen fields.	~
Collection Variable	Stores multiple updatable values that have the same data type, such as a group of email addresses.	~
Constant	Stores a fixed value.	~
Dynamic Record Choice	Represents a set of choices that's generated from an object's records.	*
Element	Any element that you add to the flow is available as a Resource with the was visited operator in outcome criteria. An element is considered visited if the element has already been executed in the flow interview.	
	Any element that you add to the flow that supports a fault connector is available as a Boolean resource. If the element has already been successfully executed in the flow interview, the resource's value is True. If the element wasn't executed or was executed and resulted in an error, the resource's value is False.	
Formula	Calculates a value by using other resources in your flow and Salesforce functions	~

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Cloud Flow Designer Guide

Flow Resource	Description	Creatable from the Resources Tab
Global Constant	Fixed, system-provided values, such as EmptyString, True, and False, that can be assigned as the values of flow resources.	
Global Variables	System-provided variables that reference information about the organization or running user, such as the user's ID or the API session ID. In Cloud Flow Designer, global variables are available in only flow formulas.	
Outcome	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.	
Picklist Choice	Represents a set of choices that's generated from the values of a picklist or multi-select picklist field.	~
Picklist Values	System-provided values for picklist fields in sObject variables and sObject collection variables. Available for only Assignment and Decision elements.	
Screen Field	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.	
SObject Collection Variable	Stores updatable field values for one or more Salesforce records.	~
SObject Variable	Stores updatable field values for a Salesforce record.	~
Stage	Represents the user's progress through the flow.	✓
System Variable	Provides information about the running interview. Some variables contain system-provided values. You can update the other variables throughout the flow by using Assignments.	
Text Template	Stores formatted text.	✓
Variable	Stores a value that can be updated as the flow executes.	~
Wait Event	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 .	

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 Flows

System-provided variables that reference information about the organization or running user, such as the user's ID or the API session ID. In Cloud Flow Designer, 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.

Flow Stage Resource

Represents the user's progress through the flow. You can reference stages in flow logic or in the UI, such as with a progress indicator. For example, in a payment flow, the stages are payment details, shipping details, billing details, and order confirmation.

System Variables in Flows

System variables provide information about the running interview. Some variables contain system-provided values. You can update the other variables throughout the flow by using Assignments.

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.

Field	Description
Label	A user-friendly label for the choice option.
Unique Name	A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Helps you differentiate this choice option from other resources.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Value Data Type	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.
Scale	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.

Field	Description
Label	A user-friendly label for the text box.
Required	Forces users to enter a value in the text box before they can progress or finish the flow.
Validate	Evaluates whether the user entered an acceptable value.

These fields appear only if you select Validate.

Field	Description
Formula	Boolean formula expression that evaluates whether the user entered an acceptable value.
Error Message	Displays if the user didn't enter an acceptable value.

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 My Label renders as My Label 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 Choice Screen 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.

Field	Description
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Helps you differentiate this collection variable from other resources.
Data Type	Determines the type of values that can be assigned to the collection variable.
Input/Output Type	Determines whether the collection variable can be accessed outside the flow.
	Private—Can be assigned and used only within the flow
	 Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs
	 Output—Can be accessed from Visualforce controllers and other flows
	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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
	from URL parameters, Visualforce controllers, subflows, and processes.
255 M 20	
SEE ALSO:	
Flow Resources	
Flow sObject Collec	ction Variable Resource
Flow Loop Element	
Cross-Object Field	References in Flows

Flow Constant Resource

A flow constant represents a fixed value. Unlike variables, this value can't change throughout a flow.

Field	Description	Available in: bo	
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided	Classic and Lig Experience	
	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.	Available in: Es Professional, E Performance, and Develope	
Description	Helps you differentiate this constant from other resources.		
Data Type	Determines the types of values that the constant can store.		
Value	The constant's value. This value doesn't change throughout the flow.		

EDITIONS

oth Salesforce htning

Flow Resources

sentials, Interprise, Unlimited, **r** Editions

SEE ALSO:

Flow Resources

Flow Dynamic Record Choice Resource

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

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Value Data Type	Data type of the choice's stored value.
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 <i>every</i> 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 T_{ype} 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	Controls the number of options that appear in the screen field that uses this dynamic record choice.
choices to	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	Takes field values from the record that the user chose and stores them in flow variables that you can reference later.
variables	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

EDITIONS

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 Choice Screen 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.

Field	Description	Available in: both Salesforce
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name,	Classic and Lightning Experience
	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.	Available in: Essentials , Professional , Enterprise , Performance , Unlimited , and Developer Editions
Description	Helps you differentiate this formula from other resources.	
Value Data Type	The data type for the value calculated by the formula.	
Scale	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.	
Formula	The formula expression that the flow evaluates at run time. The returned value must be compatible with Value Data Type.	
	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.

Global Constant	Supported Data Types
{!\$GlobalConstant.True}	Boolean
{!\$GlobalConstant.False}	Boolean
{!\$GlobalConstant.EmptyString}	Text

Sconstant.True and 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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 Flows

System-provided variables that reference information about the organization or running user, such as the user's ID or the API session ID. In Cloud Flow Designer, global variables are available only in flow formulas.

Scample: 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.

Global Variable	Description
\$Api	References API URLs or the session ID. The following merge fields are available.
	 Enterprise_Server_URL_xxx—The Enterprise WSDL SOAP endpoint where xxx represents the version of the API.
	 Partner_Server_URL_xxx—The Partner WSDL SOAP endpoint where xxx represents the version of the API.
	• Session_ID

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Global Variable	Description
\$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.
	Тір:
	 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 <i>supposed</i> to see, while \$User.UIThemeDisplayed returns the look and feel the user <i>actually</i> 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.

Global Variable	Description
	Theme1—Obsolete Salesforce theme
	 Theme 2—Salesforce Classic 2005 user interface theme
	 Theme 3—Salesforce Classic 2010 user interface theme
	 Theme4d—Modern "Lightning Experience" Salesforce theme
	 Theme4t—Salesforce mobile app theme
	 Theme4u—Lightning Console theme
	 PortalDefault—Salesforce Customer Portal theme
	Webstore—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.

Field	Description
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Helps you differentiate this resource from other resources.
Value Data Type	Data type of the choice's stored value.
Object	The object whose fields you want to select from.
Field	The picklist or multi-select picklist field to use to generate the list of choices.
Sort Order	Controls the order that the choices appear in. The choices sort based on the translated picklist value for the running user's language.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

S 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).

Unique Name * picklist_AccountIndustry	i
Add Description	
Value Data Type * Picklist	•
Object * Account 2	
Field * Industry 1	
Sort Order: Ascending	

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:

Choice Screen Fields Options for Choice Fields in Flow Screen Elements Flow Resources

EDITIONS

Flow sObject Collection Variable Resource

Stores updatable field values for one or more Salesforce records.

Field	Description	Available in: both Salesforce
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided	Classic and Lightning Experience
	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.	Available in: Essentials , Professional , Enterprise , Performance , Unlimited , and Developer Editions
Description	Helps you differentiate this sObject collection variable from other resources.	
Input/Output Type	Determines whether the sObject collection variable can be accessed outside the flow.	
	Private—Can be assigned and used only within the flow	
	 Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs 	
	 Output—Can be accessed from Visualforce controllers and other flows 	
	The default value is Private.	
	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.	
Object Type	Type of Salesforce records that the sObject collection represents in the flow.	

Usage

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

To add or remove items in a collection, use an Assignment element.

To replace all items in a collection, use a Fast Lookup element.

You can't update existing collection items, but you can work around this by using a Loop element to iterate through the collection. One by one, the loop copies each collection item into an sObject variable that you specify as the *loop variable*. Within the loop, configure elements to update the loop variable and then add the updated content to a second collection. After the loop, update the Salesforce records with the values from the second collection.



SEE ALSO:

Sample Flow That Loops Through a Collection Flow Loop Element Operators in Flow Assignment Elements Flow Resources

Flow sObject Variable Resource

Stores updatable field values for a Salesforce record.

Field	Description	Availab	
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided	Classic Experie	
	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.	Availab Profess Perforn and De	
Description	Helps you differentiate this sObject variable from other resources.		

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Field	Description
Input/Output Type	Determines whether the sObject variable can be accessed outside the flow.Private—Can be assigned and used only within the flow
	 Input—Can be set at the start of the flow using Visualforce controllers, or subflow inputs Output—Can be accessed from Visualforce controllers and other flows
	This field doesn't affect how variables are assigned or used within the same flow, for example, through these types of elements: Assignment, Record or Fast Create, Record or Fast Lookup, and Apex Plug-in.
	The default value of the field is Private.
	Warning: Disabling input or output access for an existing variable can break the functionality of applications and pages that call the flow and access the variable. For example, you can access variables from URL parameters, Visualforce controllers, subflows, and processes.
Object Type	Type of Salesforce record that the sObject variable represents in the flow.

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

Flow Stage Resource

Represents the user's progress through the flow. You can reference stages in flow logic or in the UI, such as with a progress indicator. For example, in a payment flow, the stages are payment details, shipping details, billing details, and order confirmation.

Field	Description
Label	A user-friendly label for the stage. Merge fields aren't supported.
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Helps you differentiate this stage from other resources.
Order	Required. Determines how to sort this stage among the other stages in this flow. The order must be unique among all other stages in the flow.
Active by Default	Adds this stage to { !\$Flow.ActiveStages } when an interview starts.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Usage

When ordering your stages, leave gaps between the numbers in case you later want to add a stage between two other stages. For example, if you use 10, 20, and 30 as the order, you can insert a stage at order 15 without having to update the original three stages.

Most of the time, stages resolve to the fully qualified name: namespace.flowName:stageName or flowName:stageName. Stages resolve to the label in:

- Display contexts, such as choice labels and Display Text fields
- Attributes in Lightning component screen fields

SEE ALSO:

Plan the Stages in Your Flow Identify the Relevant Stages in Your Flow Considerations for Flow Stages Flow Resources

System Variables in Flows

System variables provide information about the running interview. Some variables contain system-provided values. You can update the other variables throughout the flow by using Assignments.

System Variable	Supported Resource Types	Description	Value Set By
{!\$Flow.ActiveStages}	Stage	A collection of stages that are relevant to the current path of the flow.	Assignment
		For example, each item in a progress indicator corresponds to a stage in \$Flow.ActiveStages.	
{!\$Flow.CurrentDate}	Text, Date, and DateTime	Date when the flow interview executes the element that references the system variable.	System
{!\$Plow.CurrentRecord}	Text	ID of a related record. The value must be a single ID for a valid object. All custom objects and most standard objects are valid.	Assignment
		When a user pauses the flow interview or the interview executes a Wait element, the interview is associated with this record by creating a FlowRecordRelation record. If the ID isn't valid, the interview doesn't pause.	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

System Variable	Supported Resource Types	Description	Value Set By
{!\$Flow.CurrentStage}	Stage	The stage that's currently selected. For example, the selected item in a progress indicator corresponds to \$Flow.CurrentStage.	Assignment
{!\$Flow.CurrentDateTime	Text, Date,and DateTime	Date and time when the flow interview executes the element that references the system variable.	System
{!\$Flow.FaultMessage}	Text	System fault message that can help flow administrators troubleshoot runtime issues.	System
{!\$Flow.InterviewGuid}	Text	Unique identifier for the interview.	System

Example: A flow is used internally by call center personnel. For each flow element that interacts with the Salesforce database, a fault connector leads to a screen. A Display Text 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}
```

Example: If a customer asks to be forgotten, make sure to delete all references to information that could personally identify the customer, including data in paused flow interviews. When an interview executes a Wait element or is paused by a user, all the interview data is serialized and saved to the database as a Paused Flow Interview record. When the interview is resumed, the Paused Flow Interview record is deleted.

To identify which paused interviews include personal data for a contact, lead, or user, build a custom object to track the interview's GUID and the affected contact, lead, or user. When an interview references personal data, such as a lead's email or credit card number, create a record of the custom object using the lead's ID and { !\$Flow.InterviewGuid}. Before the final screen, delete all records of the custom object referencing the interview's GUID. That way, the custom object tracks only interviews that are saved to the database.

When a customer asks to be forgotten, create a report that lists all the custom object records where LeadId matches the customer's record. Then for each custom object record, delete the flow interview that corresponds to the provided GUID.

SEE ALSO:

Customize What Happens When a Flow Fails Flow Resources

Flow Text Template Resource

Stores HTML-formatted text.

Field Description				
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are	Classic a Experien		
	used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.			
Description Helps you differentiate this text template from other resources.		and Dev		
Text Template	The text for the template. Use HTML to format the text and merge fields to reference information from other resources.			

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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

SEE ALSO:

Flow Resources Cross-Object Field References in Flows

Flow Variable Resource

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

Field	Description
Unique Name	The requirement for uniqueness applies only to elements within the current flow. Two elements can have the same unique name, provided they are used in different flows. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.
Description	Helps you differentiate this variable from other resources.
Data Type	Determines the types of values that can be assigned to the variable.
Scale Controls the number of digits to the right of the decimal point. Controls the number of digits to the right of the decimal point. Controls the value of the decimal point of the decimal point of the decimal point. Appears only when the Data Type is set to Number or Cur	
Input/Output Type Determines whether the variable can be accessed outside the f Private—Can be assigned and used only within the flow	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 doesn't affect how variables are assigned or used within the same flow, for example, through these types of elements: Assignment, Record or Fast Create, Record or Fast Lookup, and Apex Plug-in.
The default value of the field depends on the release or API version in which the variable is created:
• Private for a variable created in Summer '12 and later or in API version 25.0 and later.
• Input and Output for a variable created in Spring '12 and earlier or in API version 24.0.
Worning: 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.
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

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

{!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_Contact.Account.Id} references Id of the account that's related to the contact 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

{!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.

EDITIONS



1. Use a Fast Lookup element to store the contract's fields, including AccountId, in an sObject variable called varContract.

- 2. Use a Decision element to verify that the value of AccountId was set in varContract.
- 3. Use a Fast Lookup to store the fields for the contract's account, including OwnerId, in another sObject variable called varAccount.
- 4. Use a Decision element to confirm that the value of OwnerId was set in varAccount.
- 5. Use an Assignment element to specify {!varContract.Account.OwnerId} as the value for {!varContract.OwnerId}.
- 6. Use a Fast Update element to write the values in varContract, including the updated OwnerId value, to the contract in Salesforce.

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.

Г		Т	\cap	N IC
Е	ט	ш	U	172

Label	Example	Description
Unlabeled		Identifies which element to execute next.
Decision outcome name	X>Y▶	Identifies which element to execute when the criteria of a Decision outcome are met.
Wait event name	──1 day after X →	Identifies which element to execute when an event that's defined in a Wait element occurs.
FAULT	FAULT>	Identifies which element to execute when the previous element results in an error.
Next element	Next element ->>	Identifies the first element to execute for each iteration of a Loop element.
End of loop	End of loop —>	Identifies which element to execute after a Loop element finishes iterating through a collection.

Available in: both Salesforce Classic and Lightning Experience

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

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
- Stage
- sObject
- Text

Boolean

Replace a Boolean resource with a new value.

Operator	Description	Supported Data Types	Example
equals	What you enter or select for Value	Boolean	Before Assignment: { ! varBoolean } is false
	replaces the value of Variable.		Assignment: { !varBoolean } equals { !\$GlobalConstant.True }

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Operator	Description	Supported Data Types	Example
			After Assignment: { !varBoolean} is true

Collection

Update or replace the value of a collection variable or sObject collection variable.

Operator	Description	Supported Data Types	Example
equals	Value replaces the value of Variable.	Collection of the same data type or object type Text, Picklist, and Multi-Select Picklist data types are compatible with each other.	<pre>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</pre>
add	Value is added as a new item at the end of the collection in Variable.	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. Stages (including \$Flow.CurrentStage) can be added to text collections. Via Metadata API only, you can add collections to collections of the same data type or object type. From Cloud Flow Designer, you can't save an Assignment element that contains a collection variable in the Value column for the "add" operator.	<pre>Before the Assignment: {!collText} is Yellow, Green, Blue {!varPicklist} is Red Assignment: {!collText} add {!varPicklist} After the Assignment: {!collText} is Yellow, Green, Blue, Red</pre>
remove after first	The first instance of Value is found within the collection in Variable. All items after the first instance are removed from the collection in Variable.	 Variable of the same data type or sObject variable of the same object type For text collections only: Multi-Select Picklist Picklist 	<pre>Before the Assignment: • {!collText} is Red, Orange, Yellow, Green, Blue • {!varText} is Yellow Assignment: {!collText} remove after first {!varText}</pre>

Operator	Description	Supported Data Types	Example	
		• \$Flow.CurrentRecord	After the Assignment: { ! collText } is Red, Orange, Yellow	
add at start	Value is added as a new item at the beginning of the collection in Variable.	Collection of the same data type or object type Variable of the same data type or sObject variable of the same object type For text collections only: Multi-Select Picklist Picklist \$Flow.CurrentRecord	<pre>Before the Assignment: {!collText} is Yellow, Green, Blue {!varPicklist} is Red Assignment: {!collText} add at start {!varPicklist} After the Assignment: {!collText} is Red, Yellow, Green, Blue</pre>	
remove all	All instances of Value are removed from the collection in Variable.	Collection of the same data type or object type Variable of the same data type or sObject variable of the same object type For text collections only: • Multi-Select Picklist • Picklist • \$Flow.CurrentRecord	<pre>Before the Assignment: {!collText} is Red, Orange, Red, Yellow {!varText} is Red Assignment: {!collText} remove all {!varText} After the Assignment: {!collText} is Orange, Yellow</pre>	
remove first	The first instance of Value is removed from the collection in Variable.	Collection of the same data type or object type For text collections only: • Multi-Select Picklist • Picklist • \$Flow.CurrentRecord	<pre>Before the Assignment: • {!collText} is Red, Orange, Red, Yellow • {!varText} is Red Assignment: {!collText} remove first {!varText} After the Assignment: {!collText} is Orange, Red, Yellow</pre>	
remove before first	The first instance of Value is found within the collection in Variable. All items before the first instance are removed from the collection in Variable.	 Variable of the same data type or sObject variable of the same object type For text collections only: Multi-Select Picklist Picklist \$Flow.CurrentRecord 	<pre>Before the Assignment: • {!collText} is Red, Orange, Yellow, Green, Blue • {!varText} is Yellow Assignment: {!collText} remove before first {!varText} After the Assignment: {!collText} is Yellow, Green, Blue</pre>	
remove position	Value specifies a position in the collection. The item at the position is removed	Number	Before the Assignment:{!collText} is Red, Orange, Yellow	

Operator	Description	Supported Data Types	Example
	from the collection in Variable.	 {!varNum} is 2 Assignment: {!collText} remove position {!varNum} 	
	Make sure that Value at run time is a positive integer that is within the range of the number of items in the collection in Variable.		After the Assignment: { ! collText } is Red, Yellow
remove uncommon	The items in the Value collection are found within the Variable collection. The found items are kept, and all other items are removed from the collection in Variable.	Collection of the same data type or object type	Before the Assignment:
			 {!collText1} is Red, Orange, Yellow, Green
			• {!collText2} is Orange, Green, Blue
			<pre>Assignment: { !collText1 } remove uncommon { !collText2 }</pre>
			After the Assignment: { !collText1} is Orange, Green

Currency and Number

Replace (equals), add to (add), or subtract from (subtract) the value of a currency or number resource. Count (equals count) the number of active stages or the number of items in a collection.

Operator	Description	Supported Data Types	Example
equals	The number that you enter or select for Value replaces the value of Variable.	CurrencyNumber	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.	CurrencyNumber	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.	CurrencyNumber	Before the Assignment: { !varCurrency} is 10 Assignment: { !varCurrency} subtract 7 After the Assignment: { !varCurrency} is 3
equals count	The number of stages or collection items in what you enter for Value replaces the value of Variable.	Collection\$Flow.ActiveStages	<pre>Before the Assignment: {!varNumber} is 0 {!collText} is Yellow, Green, Blue Assignment: {!varNumber} equals count {!collText}</pre>

Operator Description

Supported Data Types Example

After the Assignment: { !collText } is 3

Date

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

Operator	Description	Supported Data Types	Example
equals	The date that you enter or select for Value replaces the value of Variable.	DateDate/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.	CurrencyNumber	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.	CurrencyNumber	Before the Assignment: { !varDate} is 1/16/2016 Assignment: { !varDate} subtract 7 After the Assignment: { !varDate} is 1/9/2016

Date/Time

Replace a date/time resource with a new value (equals).

Operator	Description	Supported Data Types	Example
equals	The date that you enter or select for Value replaces the value of Variable.	DateDate/Time	Before the Assignment: { !varDateTime } is 1/16/2016 01:00
			<pre>Assignment: {!varDateTime} equals 1/16/2016 08:00</pre>
			After the Assignment: { !varDateTime} is 1/16/2016 08:00

Picklist

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

Note: Before values are assigned or added to a picklist resource, they're converted into string values.

Operator	Description	Supported Data Types	Example
equals	What you enter or select for Value replaces the value of the selected picklist.	 Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text 	Before the Assignment: {!varPicklist} is Blue Assignment: {!varPicklist} equals Yellow After the Assignment: {!varPicklist} is Yellow
add	What you enter or select for Value is added to the end of the selected picklist.	 Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text 	Before the Assignment: { !varPicklist } is Blue Assignment: { !varPicklist } add -green After the Assignment: { !varPicklist } is Blue-green

Multi-Select Picklist

Replace a multi-select picklist resource with a new value (equals), concatenate a value onto the original value (add), or add a selection to the resource (add item).

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

Operator	Description	Supported Data Types	Example
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

Operator	Description	Supported Data Types	Example
add	What you enter or select for Value is added to the last item selected in the multi-select picklist. It doesn't create a selection. Easily add items to a multi-select picklist resource by using the "add item" operator. To add semi-colon-delimited items to a multi-select picklist variable with the "add" operator, always add 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	 Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text 	Before the Assignment: {!varMSP} is Blue; Green. This value includes two separate selections Assignment: {!varMSP} add Yellow After the Assignment: {!varMSP} is Blue; GreenYellow. This value includes two separate selections
add item	What you enter or select for Value is added as a new selection to the end of the multi-select picklist. The Assignment automatically adds ";" before the new item. That way, Salesforce reads it as a separate item selected by the multi-select picklist.	 Boolean Currency Date Date/Time Multi-Select Picklist Number Picklist Text 	Before the Assignment: {!varMSP} is Blue; Green Assignment: {!varMSP} add item Yellow After the Assignment: {!varMSP} is Blue; Green; Yellow. This value includes three separate selections

Stage

You can't update the value of a stage, but you can update the values of the stage system variables: <code>\$Flow.CurrentStage</code> and <code>\$Flow.ActiveStages</code>.

Note: Assignments use the stage's fully qualified name: namespace.flowName:stageNameor flowName:stageName.

\$Flow.CurrentStage

Replace the stage selected in \$Flow.CurrentStage.

Operator	Description	Supported Data Types	Example
equals	The stage that you select for Value replaces the value of \$Flow.CurrentStage.	Stage	Before the Assignment: \$Flow.CurrentStage is stage1
			<pre>Assignment: { ! \$Flow.CurrentStage } equals { !stage2 }</pre>
			After the Assignment: \$Flow.CurrentStage is stage2

\$Flow.ActiveStages

Add or remove active stages in the \$Flow.ActiveStages system variable.

Operator	Description	Supported Data Types	Example
add	Value is added to the end of \$Flow.ActiveStages.	Stage\$Flow.ActiveStages\$Flow.CurrentStage	<pre>Before the Assignment: \$Flow.ActiveStages is stage1, stage2 Assignment: {!\$Flow.ActiveStages} add {!stage3} After the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3</pre>
remove after first	The first instance of the stage in Value is found within \$Flow.ActiveStages. All stages after the first instance are removed from \$Flow.ActiveStages.	StageText\$Flow.CurrentStage	<pre>Before the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3, stage4 Assignment: {!\$Flow.ActiveStages} remove after first {!stage2} After the Assignment: \$Flow.ActiveStages is stage1, stage2</pre>
add at start	Value is added to the beginning of \$Flow.ActiveStages.	Stage\$Flow.ActiveStages\$Flow.CurrentStage	<pre>Before the Assignment: \$Flow.ActiveStages is stage1, stage2 Assignment: {!\$Flow.ActiveStages} add at start {!stage0} After the Assignment: \$Flow.ActiveStages is stage0, stage1, stage2</pre>
remove all	All instances of Value are removed from \$Flow.ActiveStages.	Stage\$Flow.ActiveStages\$Flow.CurrentStage	Before the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3 Assignment: {!\$Flow.ActiveStages} remove all {!\$Flow.ActiveStages} After the Assignment: \$Flow.ActiveStages is empty

Operator	Description	Supported Data Types	Example
remove first	The first instance of the stage in Value is removed from \$Flow.ActiveStages.	Stage\$Flow.CurrentStage	<pre>Before the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3, stage1 Assignment: {!\$Flow.ActiveStages} remove first stage1 After the Assignment: \$Flow.ActiveStages is stage2, stage3, stage1</pre>
remove before first	The first instance of the stage in Value is found within \$Flow.ActiveStages. All stages before that first instance are removed from \$Flow.ActiveStages.	Stage\$Flow.CurrentStage	<pre>Before the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3, stage4 Assignment: {!\$Flow.ActiveStages} remove before first {!stage3} After the Assignment: \$Flow.ActiveStages is stage3, stage4</pre>
remove position	Value specifies a position in \$Flow.ActiveStages. The stage at that position is removed from \$Flow.ActiveStages. Make sure that Value at run time is a positive integer that is within the range of the number of stages in \$Flow.ActiveStages.	Number	 Before the Assignment: \$Flow.ActiveStages is stage1, stage2, stage3 {!varNum} is 2 Assignment: {!\$Flow.ActiveStages} remove position {!varNum} After the Assignment: \$Flow.ActiveStages is stage1, stage3

sObject

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

Operator	Description	Supported Data Types	Example
equals The sObject that y replaces the value	The sObject that you select for Value	sObject with the same object type	Before the Assignment:
	replaces the value of Variable.		 {!account1} contains field values for the Acme Wireless account
			 {!account2} contains field values for the Global Media account
		Assignment: { !account1 } equals { !account2 }	
		After the Assignment: both { !account1 } and { !account2 } contain the field values for the Global Media account	

Text

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

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

Operator	Description	Supported Data Types	Example
equals	The text that you enter or select for Value replaces the value of Variable.	 Boolean Currency Date Date/Time Number Multi-select picklist Picklist Stage, including \$FlowCurrentStage and \$FlowActiveStages 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 Stage, including \$FlowCurrentStage and \$FlowActiveStages 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
- Stage
- Text

Boolean

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

Operator	True if	Supported Data Types
does not equal	The value of the selected Resource doesn't match what you enter or select for Value.	Boolean
equals	The value of the selected Resource matches what you enter or select for Value. 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.	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
was visited	The selected Resource is an element in the flow, and it has been visited during the flow interview.	Boolean

Choice

Every choice resource has a data type and obeys the operator rules for that data type. However, choice resources support one extra operator that other resources don't, no matter what their data type is.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Operator	True if	Supported Data Types
was selected	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.	Boolean
	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.	

Collection

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

Operator	True if	Supported Data Types
contains	An item in the collection that's selected for Resource contains the exact same value as Value	Resource of the same data type. For sObject collection variables, only sObject resources with the same object type are supported.
does not equal	The collection that's selected for Resource doesn't match the collection that's selected for Value Two sObject collection variables are unequal if they include different fields or if the fields have different values.	Collection of the same data type. For sObject collection variables, only sObject collection variables with the same object type are supported.
equals	The collection that's selected for Resource matches the collection that's selected for Value Two sObject collection variables are equal if they include the same fields and those fields have the same values.	Collection of the same data type. For sObject collection variables, only sObject collection variables with the same object type are supported.
is null	The collection that's selected for resource isn't populated with any values	Boolean

Currency and Number

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

Operator	True if	Supported Data Types	
does not equal	The value for Resource doesn't match what's entered or selected for Value	CurrencyNumber	
equals	The value for Resource matches what's entered or selected for Value	CurrencyNumber	
greater than	The value of the Resource is larger than what's entered or selected for Value	Currency	

Operator	True if	Supported Data Types
		• Number
greater than or equal	The value of the Resource is larger than what's entered or selected for Value or is the same	CurrencyNumber
less than	The value of the Resource is smaller than what's entered or selected for Value	CurrencyNumber
less than or equal	The value of the Resource is smaller than what's entered or selected for Value or is the same	CurrencyNumber
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

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	DateDate/Time	
equals	The value for Resource matches what's entered or selected for Value	DateDate/Time	
greater than	The value of the Resource is a later date or time than what's entered or selected for Value	DateDate/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	DateDate/Time	
less than	The value of the Resource is an earlier date or time than what's entered or selected for Value	DateDate/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	DateDate/Time	
is null	Resource isn't populated with a value	Boolean	

Operator	True if	Supported Data Types
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.

C	Note:	hese operators treat the resource's value as a text value.
---	-------	--

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

Operator	True if	Supported Data Types
		• Text
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

Multi-Select Picklist

Check whether a multi-select picklist resource's value matches or contains another value or resource.

Note: These operators treat the resource's value as a text value. If the resource's value includes multiple items, the operators treat the value as one string that happens to include semi-colons. It doesn't treat each selection as a different value. For example, the operators treat red; blue; green as a single value rather than three separate values.

Operator	True if	Supported Data Types
contains	 The value for Resource contains what's entered or selected for Value Tip: When you use this operator for a multi-select picklist resource, be aware of the values that a user can enter. If you want to check that a specific value is included and that value is also included as part of another value, create a flow formula resource that uses the INCLUDES function. For example, your organization has a Color multi-select picklist value. Among the possible values are "green" and "yellow-green". If both "green" and "yellow-green" are acceptable values, use the contains operator in a flow condition. If only "green" is an acceptable value, create a formula that uses the INCLUDES() function. 	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Text
does not equal	 The value for Resource doesn't match what's entered or selected for Value Note: Order matters. If you aren't sure which order the values that you're checking for will appear in, use the INCLUDES() function in a flow formula. For example, if you compare "red; blue; green" to "blue; green; red" using the does not equal operator, that condition resolves to true. 	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Text
equals	The value for Resource exactly matches what's entered or selected for Value Note: Order matters. If you aren't sure which order the values that you're checking for will appear in, use the INCLUDES() function in a flow formula. For example, if you compare "red; blue; green" to "blue; green; red" using the equals operator, that condition will resolve to false.	 Boolean Currency Date Date/Time Multi-select Picklist

Operator	True if	Supported Data Types	
		•	Number
		•	Picklist
		•	Text
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	Bo	olean

Stage

Note: Stages resolve to the fully qualified stage name: namespace.flowName:stageName or flowName:stageName.

Check whether a Stage resource or the <code>\$Flow.CurrentStage</code> system variable matches, ends with, or starts with another value or resource.

Operator	True if	Supported Data Types
does not equal	The value for Resource doesn't match what's entered or selected for Value	StageText
equals	The value for Resource matches what's entered or selected for Value	StageText
ends with	The end of the value for Resource matches what's entered or selected for Value	StageText
is null	Resource isn't populated with a value	Boolean
starts with	The beginning of the value for Resource matches what's entered or selected for Value	StageText

Check whether \$Flow.ActiveStages contains a particular stage, matches the value of a Text collection, or is null.

Operator	True if	Supported Data Types
contains	\$Flow.ActiveStages contains an item that matches the resource that's selected for Value.	StageText
does not equal	The collection that's selected for Resource doesn't match \$Flow.ActiveStages.	Text collection
equals	The collection that's selected for Resource doesn't match <code>\$Flow.ActiveStages</code> .	Text collection

Operator	True if	Supported Data Types
is null	\$Flow.ActiveStages isn't populated with any stages.	Boolean

sObject

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

Operator	True if	Supported Data Types
does not equal	The value for Resource doesn't match what's entered or selected for Value	sObject with the same object type
equals	The value for Resource matches what's entered or selected for Value	sObject with the same object type
is null	Resource isn't populated with a value	Boolean

Text

Check whether a Text resource's value matches, contains, ends with, or starts with another value or resource.

Note:

- Before values are compared to a text resource, they're converted into string values.
- Stages resolve to the fully qualified stage name: namespace.flowName:stageName or flowName:stageName.

Operator	True if	Supported Data Types
contains	The value for Resource contains what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
does not equal	The value for Resource doesn't match what's entered or selected for Value	BooleanCurrencyDateDate/Time

Operator	True if	Supported Data Types
		 Multi-select Picklist Number Picklist Stage Text
equals	The value for Resource matches what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
ends with	The end of the value for Resource matches what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
is null starts with	Resource isn't populated with a value The beginning of the value for Resource matches what's entered or selected for Value	Boolean Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage

Operator	True if	Supported Data Types
		• Text
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

Operators in Flow Record Filters

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

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

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

Checkbox Fields

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Operator	Filters to records where the selected field's value	Supported Data Types
does not equal	Doesn't match what you enter or select for Value	Boolean
equals	Matches what you enter or select for Value	Boolean
is null	Hasn't been populated with a value yet (if you select True for Value)	Boolean

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.

Operator	Filters to records where the selected field's value	Supported Data Types
does not equal	Doesn't match what's entered or selected for Value	CurrencyNumber
equals	Matches what's entered or selected for Value	CurrencyNumber
greater than	Is larger than what's entered or selected for Value	CurrencyNumber
greater than or equal	Is larger than what's entered or selected for Value or is the same	CurrencyNumber
is null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
less than	Is smaller than what's entered or selected for Value	CurrencyNumber
less than or equal	Is smaller than what's entered or selected for Value or is the same.	CurrencyNumber

Date and Date/Time

When you select a date or date/time field under Field, these operators are available.

Operator	Filters to records where the selected field's value	Supported Data Types
does not equal	The value for Resource doesn't match what's entered or selected for Value	DateDate/Time
equals	The value for Resource matches what's entered or selected for Value	DateDate/Time
greater than	The value of the Resource is a later date or time than what's entered or selected for Value	DateDate/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	DateDate/Time
is null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
less than	The value of the Resource is an earlier date or time than what's entered or selected for Value	DateDate/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	DateDate/Time

Picklist and Text Fields

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

Operator	Filters to records where the selected field's value	Supported Data Types
contains	Contains what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
does not equal	Doesn't match what's entered or selected for Value	BooleanCurrency

Operator	Filters to records where the selected field's value	Supported Data Types
		 Date Date/Time Multi-select Picklist Number Picklist Stage Text
equals	Matches what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
ends with	Ends with what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
is null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
starts with	Begins with what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage

Operator	Filters to records where the selected field's value	Supported Data Types
		• Text

Multi-Select Picklist Fields

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

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

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

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

Operator	Filters to records where the selected field's value	Supported Data Types
does not equal	Doesn't match what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
equals	Matches what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text
ends with	Ends with what's entered or selected for Value	BooleanCurrencyDateDate/Time

Operator	Filters to records where the selected field's value	Supported Data Types
		 Multi-select Picklist Number Picklist Stage Text
is null	Hasn't been populated with a value yet (if you select True for Value)	Boolean
starts with	Begins with what's entered or selected for Value	 Boolean Currency Date Date/Time Multi-select Picklist Number Picklist Stage Text

Flow 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.

EDITIONS

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

Alarm events are available in: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

Platform events are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

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.

Parameter	Description	Example
Event	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.	{!vendorResponse}

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.



Available in both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Parameter	Description	Example
Base Time	A date/time value. If you enter values for Offset Number and Offset Unit, this field value is the base for the offset.	{!\$Flow.CurrentDate}
	You can manually enter a date/time value or reference a merge field or flow resource.	
Offset Number	Optional. The number of days or hours to offset 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.	-3
	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.	
Offset Unit	Optional. The unit to offset Base Time. Required if you set a value for Offset Number.	Days
	Manually enter <i>Days</i> or <i>Hours</i> . You can't use a merge field or flow resource for this value.	

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.

Parameter	Description	Example
Base Time	The actual time at which the event occurred and the flow interview resumed.	11/26/2014 10:12 AM
Event Delivery Status	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
	• 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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Parameter	Description	Example
Object Type	The API name of the object whose field you want to base the alarm on. See Supported Objects. on page 194	Contract
	You must manually enter a string. You can't use a merge field or flow resource for this value.	
Base Date/Time Field	The API name for a date or date/time field on the specified object. If you enter values for Offset Number and Offset Unit, this field value is the base for the offset. Manually enter a string.	EndDate
Record ID	ID of the record that the alarm is based on. The record's object type must match Object Type. You can enter a string, merge field, or flow resource.	{!ContractId}
Offset Number	Optional. The number of days or hours to offset Base Date/Time Field. Required if you set a value for Offset Unit.	-3
	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 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 if you set a value for Offset Number.	Days
	Manually enter <i>Days</i> or <i>Hours</i> . 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.

Parameter	Description	Example
Base Time	The actual time at which the event occurred and the flow interview resumed.	11/26/2014 10:12 AM
Event Delivery Status	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
	• Delivered: The event was successfully delivered.	
	• Invalid: An error occurred during delivery, but the flow successfully resumed.	

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
- SOSSession
- SOSSessionActivity
- ServiceContract
- SignupRequest
- Site
- SocialPersona
- SocialPost
- Solution

- SsoUserMapping
- StreamingChannel
- Task
- UsageEntitlementPeriod
- User
- UserLicense
- UserProvisioningRequest
- WorkBadge
- WorkBadgeDefinition
- WorkCoaching
- WorkFeedback
- WorkFeedbackQuestion
- WorkFeedbackQuestionSet
- WorkFeedbackRequest
- WorkFeedbackTemplate
- WorkGoal
- WorkPerformanceCycle
- WorkReward
- WorkRewardFund
- WorkRewardFundType
- WorkThanks
- WorkUpgradeAction
- WorkUpgradeCustomer
- WorkUpgradeUser
- articleType_kav

SEE ALSO:

Flow Event Types

Flow Types

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

Standard Flow Types

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Туре	Description	Available Distribution Methods	Supported in Translation Workbench
Screen Flow	Requires user interaction, because it includes screens, local actions, steps, choices, or dynamic choices. Screen flows don't support Wait elements.	 Flow actions Lightning pages Lightning community pages Custom Lightning components Custom buttons or custom links Web tabs Direct flow URLs Visualforce pages Lightning Out 	✓
Autolaunched Flow	Doesn't require user interaction. This flow type doesn't support screens, local actions, steps, choices, or dynamic choices.	 Processes Custom Apex classes REST API Web tabs Custom buttons or custom links Visualforce pages Einstein Bots 	~
User Provisioning Flow	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.	A user provisioning flow can only be implemented by associating it with a connected app when running the User Provisioning Wizard.	
Field Service Mobile Flow	Requires user interaction because it has one or more screens.	Field Service Lightning mobile app	~
Field Service Embedded Flow	Requires user interaction because it has one or more screens.	Embedded Appointment Booking	*
Contact Request Flow	Requires user interaction because it has one or more screens.	Lightning community pages Use one of the following Community Builder components to add this flow. • Contact Request Button & Flow—launch the flow in a popup window	✓

Туре	Description	Available Distribution Methods	Supported in Translation Workbench
		• Flow—embed the flow directly on the page	

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, a record change process runs and schedules some actions for next week. You can monitor the scheduled actions in the Paused and Waiting Interviews list by looking for the type "Record Change Process".

Туре	Description
Invocable Process	A process, created in Process Builder, that starts when it's called from another process.
Platform Event Process	A process, created in Process Builder, that starts when a particular plaform event occurs.
Record Change Process	A process, created in Process Builder, that starts when a record is created or edited for a particular object.
Transaction Security Flow	A flow used in the Transaction Security App.

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Property	Description
Name	The name for the flow or flow version. The name appears in the flow management page and flow detail page. It also appears in the run time user interface.
	You can edit the name for inactive flows and flow versions.
Unique Name	The unique name for the flow. The unique name is used to refer to this flow from other parts of Salesforce, such as in a URL or Visualforce page. A unique name is limited to underscores and alphanumeric characters. It must begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores. The unique name appears on the flow detail page. You can't edit the unique name after the flow has been saved.
Description	The description for the flow or flow version. The description appears in the flow management page and flow detail page. You can edit the description for inactive flows and flow versions.
	· · · · · · · · · · · · · · · · · · ·
Туре	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 196.
	If the type is Login Flow, you can't update the type after the flow has been saved.
Interview Label	The label for the flow's interviews. An <i>interview</i> 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 the Salesforce app
	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

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

Before you activate a flow, thoroughly test it to make sure that it works 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.

Prepare Your Org for Paused Flow Interviews

A *flow interview* is a running instance of a flow. Not every flow interview can be completed in one go. Add the Pause button to your flows, so that users can pause flow interviews for later. Update the sharing model for flow interviews, so that other users can resume a paused interviews. And make it easy for users to resume interviews by adding a component to their Home page.

SEE ALSO: Cloud Flow Designer Flow Limits Considerations for Managing Flows

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Flow and Flow Version Fields

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

Property	Description
Active Version	Identifies which version is active.
Description	The description for the flow or flow version
Flow Name	The name for the flow. It appears in the run time user interface.
Name	The name for the flow version. It becomes the Flow Name when this version is active.
Namespace Prefix	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.
Туре	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 196.
Status	Identifies whether the flow version is active or not.
Unique Name	Lets you refer to the flow from other parts of Salesforce, such as in Visualforce page.
URL	The relative URL that you can use to run the flow, such as from a custom button or Web tab.
Version	The number of the flow version.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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

Test a Flow

Before you activate a flow, thoroughly test it to make sure that it works as expected.

We recommend that you test all possible paths through the flow, so that you can find errors before they affect your users. For example, if users or inputs don't provide all the data that your flow requires, the flow can fail. Add fault connectors to provide paths for users or the flow logic to correct the data so that the flow can successfully finish.

- Warning: Debugging or testing a flow actually runs the flow and performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.
- 1. From Setup, enter *Flows* in the Quick Find box, then select **Flows**.
- 2. Click the name of the flow, and open the relevant version.
- 3. Save any changes that you make to the flow.

Unsaved changes aren't executed when you test the flow.

- 4. Choose one of these options.
 - (Recommended) To set input variables or see debug details as the flow runs, click **Debug**.

If you opt to view debug details, the flow is rendered in Lightning runtime even if Lightning runtime isn't enabled for the org.

- To use the runtime experience determined by your process automation settings, click one of these buttons. However, you can't set input variables or see debug details.
 - **Run**—This option runs the active version of each flow called by Subflow elements. If a referenced flow has no active version, this option runs the latest version of the referenced flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To run or debug a flow in Cloud Flow Designer:

To run a flow from the flow detail page:

Manage Flow

Run with Latest—This option runs the latest version of each flow called by Subflow elements. This button appears only
when the flow contains a Subflow element.

Once you're confident that your flow is working as expected, activate the version that you tested and distribute the flow.

SEE ALSO:

Debug a Flow in Cloud Flow Designer Flow Runtime Experiences Activate or Deactivate a Flow Version Customize What Happens When a Flow Fails Considerations for Running Flows Manage Your Flows Flows in Transactions

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To activate or deactivate a flow:

Manage Flow

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To delete a flow:Manage Flow

Prepare Your Org for Paused Flow Interviews

A *flow interview* is a running instance of a flow. Not every flow interview can be completed in one go. Add the Pause button to your flows, so that users can pause flow interviews for later. Update the sharing model for flow interviews, so that other users can resume a paused interviews. And make it easy for users to resume interviews by adding a component to their Home page.

IN THIS SECTION:

1. Let Users Pause Flow Interviews

When users can't finish a flow interview, give them the option to pause it for later by customizing your org's process automation settings. For example, a customer service representative can pause an interview when the customer doesn't have all the necessary information.

2. Add Record Context to Your Flows

EDITIONS

EDITIONS

Experience

Available in: both Salesforce Classic and Lightning

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

Available in: both Salesforce Classic and Lightning Experience

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

All it takes to associate your org's paused interviews with a record is setting the \$Flow.CurrentRecord system variable in your flow. That way, you can find all the paused flow interviews related to that record. For example, in the Change Address flow, set \$Flow.CurrentRecord to {!recordId} so that all Change Address interviews are associated with the relevant contact.

3. Make It Easy for Users to Find Their Paused Flow Interviews

Give your users an instant view of their paused flow interviews by customizing the Home page or Salesforce app navigation menu.

4. Make It Easy for Users to Find Paused Flow Interviews for a Record

From a record page, display a list of all paused flow interviews that are associated with that record with this custom Lightning component.

5. Customize Who Has Access to Paused Flow Interviews

By default, users can resume paused flow interviews as long as they have edit access. To control who has edit access, build a sharing model for the Flow Interview object. Configure the org-wide default access level, and build sharing rules to override that default for specific users or groups.

6. Restrict Who Can Resume Shared Flow Interviews

Users can resume all paused flow interviews that they have edit access to. To restrict who can resume interviews in your org, disable Let users resume shared flow interviews.

Let Users Pause Flow Interviews

When users can't finish a flow interview, give them the option to pause it for later by customizing your org's process automation settings. For example, a customer service representative can pause an interview when the customer doesn't have all the necessary information.

User Permissions Needed

To edit process automation settings:

Customize Application

- 1. From Setup, enter *Automation* in the Quick Find box, then select **Process Automation** Settings.
- 2. Select Let users pause flows.
- 3. Click Save.

Screens 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 **Allow Pause** when you configure that screen.

SEE ALSO:

General Info

Add Record Context to Your Flows

All it takes to associate your org's paused interviews with a record is setting the \$Flow.CurrentRecord system variable in your flow. That way, you can find all the paused flow interviews related to that record. For example, in the Change Address flow, set \$Flow.CurrentRecord to

{ ! recordId } so that all Change Address interviews are associated with the relevant contact.

When a user pauses an interview or an interview executes a Wait element, the interview is associated with the record through the FlowRecordRelation object.

- 1. At the beginning of your flow, add an Assignment element.
- 2. For Variable, select **\$Flow.CurrentRecord**.
- 3. For Operator, leave equals selected.
- For Value, select a variable that contains the appropriate ID. Make sure that the variable contains only one ID.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Variable		Operator		Value
{!\$Flow.CurrentRecord}	-	equals	-	{!recordId}

Make It Easy for Users to Find Their Paused Flow Interviews

Give your users an instant view of their paused flow interviews by customizing the Home page or Salesforce app navigation menu.

Lightning Experience

Add the Paused Flow Interviews component to the appropriate Home pages. This component is available only for Home pages in the Lightning App Builder. It displays paused interviews that the user has read access to.

Lightning Communities

Add the Paused Flows component to a Community page. This component is available for most pages in Community Builder, except ones like login pages and error pages. The component displays paused interviews that the user has read access to.

Salesforce mobile app

Add the Paused Flow Interviews item to the Salesforce app navigation menu. This navigation item displays paused interviews that the user has read access to.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Salesforce Classic

Add the Paused Flow Interviews related list to the appropriate home page layouts. This component displays only interviews that the user paused.

SEE ALSO: Set Up the Lightning Experience Home Page Customize Salesforce Classic Home Tab Page Layouts

Make It Easy for Users to Find Paused Flow Interviews for a Record

From a record page, display a list of all paused flow interviews that are associated with that record with this custom Lightning component.

Example: This example uses the Apex controller to get a list of interviews that are associated with the record. The component then displays the interviews in a table. For each interview, the component displays an action menu from which the user can resume or delete the interview.

When the user clicks **Resume**, the helper fires the navigateFlow action to resume the interview. When the user clicks **Delete**, the Apex controller deletes the interview.

c:interviewsByRecord Component

```
<aura:component controller="interviewsByRecordController"
implements="flexipage:availableForRecordHome,force:hasRecordId" access="global" >
   <aura:attribute name="columns" type="List" default=""/>
   <aura:attribute name="Interviews" type="Object" />
   <aura:attribute name="recordId" type="Id" />
   <aura:attribute name="ContextRecord" type="Object" />
   <aura:attribute name="overlay" type="Aura.Component"/>
   <aura:handler name="init" value="{!this}" action="{!c.init}" />
   <aura:handler event="force:refreshView" action="{!c.init}" />
   <force:recordData aura:id="contextRecord" recordId="{!v.recordId}"
      targetFields="{!v.ContextRecord}" layoutType="FULL"/>
   <lightning:overlayLibrary aura:id="overlayLib" />
   <lightning:card iconName="standard:flow" class="slds-card boundary">
       <aura:set attribute="title">
          <span class="slds-card header-link">Paused Flow Interviews</span>
       </aura:set>
       <aura:set attribute="actions">
          <lightning:buttonIcon iconName="utility:refresh" onclick="{!c.init}"</pre>
                              alternativeText="Refresh the list of interviews" />
       </aura:set>
    <table class="slds-table slds-table--bordered slds-table--cell-buffer
       slds-table_fixed-layout">
      <thead>
        <div class="slds-truncate">Interview Label</div>
           <div class="slds-truncate">Pause Reason</div>
           <div class="slds-truncate">Paused Date</div>
           <div class="slds-truncate">Current Element</div>
           <div class="slds-truncate">Owner</div>
```

```
<div class="slds-truncate"/>
         <div class="slds-th action">
           <span class="slds-assistive-text">Actions</span>
         </div>
      </t.h>
  </thead>
<!-- Use the Apex controller to fetch interviews associated
    with this record -->
  <aura:iteration items="{!v.Interviews}" var="interview">
     <t.r>
         <div class="slds-truncate" title="{!interview.InterviewLabel}">
               {!interview.InterviewLabel}
            </div>
         <div class="slds-truncate" title="{!interview.PauseLabel}">
                {!interview.PauseLabel}
            </div>
         <div class="slds-truncate" title="{!interview.PausedDate}">
                <ui:outputDateTime value="{!interview.PausedDate}"</pre>
                  format="M/d/y h:m a"/>
            </div>
         <div class="slds-truncate" title="{!interview.CurrentElement}">
                {!interview.CurrentElement}
            </div>
         <div class="slds-truncate" title="{!interview.PausedBy}">
                {!interview.PausedBy}
            </div>
         <!-- Display Resume and Delete actions in a menu at the
               end of each row -->
            <div class="slds-shrink-none">
                dightning:buttonMenu iconSize="x-small"
                  class="paused-interview-card-row-menu"
                   alternativeText="Actions for this interview"
                   onselect="{! c.handleMenuSelect }">
                   <lightning:menuItem aura:id="{!interview.Id + 'resume'}"</pre>
                      label="Resume" value="{!interview.Id + '.resume'}" />
                   <lightning:menuItem aura:id="{!interview.Id + 'delete'}"</pre>
                      label="Delete" value="{!interview.Id + '.delete'}"/>
                </lightning:buttonMenu>
```

```
</div>
</div>

</div>

</div>

</aura:iteration>

</lightning:card>
</aura:component>
```

Apex Controller

```
public class interviewsByRecordController {
```

```
@AuraEnabled
public static List<FlowRecordRelation> getInterviews(Id recordId) {
    return [ SELECT
        ParentId, Parent.InterviewLabel, Parent.PauseLabel,
        Parent.CurrentElement, Parent.CreatedDate, Parent.Owner.Name
        FROM FlowRecordRelation
        WHERE RelatedRecordId = :recordId ];
}
@AuraEnabled
public static FlowInterview deleteInterview(Id interviewId) {
    FlowInterview interview = [Select Id from FlowInterview Where Id = :interviewId];
    delete interview;
    return interview;
}
```

c:interviewsByRecord JavaScript Controller

```
( {
  init : function(component, event, helper) {
     helper.populateTable(component, event, helper);
  },
  handleMenuSelect: function(component, event, helper) {
     // Figure out which action was selected
     var interviewAction = event.getParam("value").split(".");
     if(interviewAction.includes("resume")) {
        helper.handleShowModal(component, interviewAction[0]);
      } else if(interviewAction.includes("delete")) {
        helper.handleDelete(component, event, helper, interviewAction[0]);
      }
  },
  statusChange: function(component, event) {
      // When the interview finishes, close the overlay
     if(event.getParam("status").includes("FINISHED")) {
        component.get("v.overlay").close();
      }
  }
})
```

c:interviewsByRecord Helper

```
( {
   populateTable : function(component, event, helper) {
        var action = component.get("c.getInterviews");
        action.setParams({
            recordId: component.get("v.recordId")
        });
        action.setCallback(this, $A.getCallback(function (response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                // Push interviews fetched by the Apex controller to the component
                var recordRelations = response.getReturnValue();
                var interviews = [];
                for (var i = 0; i < recordRelations.length; i++) {</pre>
                    interviews.push(
                        {
                            Id: recordRelations[i].ParentId,
                            InterviewLabel: recordRelations[i].Parent.InterviewLabel,
                            PauseLabel: recordRelations[i].Parent.PauseLabel,
                            CurrentElement: recordRelations[i].Parent.CurrentElement,
                            PausedDate: recordRelations[i].Parent.CreatedDate,
                            PausedBy: recordRelations[i].Parent.Owner.Name
                        });
                }
                component.set('v.Interviews', interviews);
            } else if (state === "ERROR") {
                var errors = response.getError();
                console.error(errors);
            }
        }));
        $A.enqueueAction(action);
   },
   handleShowModal: function (component, id) {
        // On resume, render the interview in a modal
        $A.createComponent("lightning:flow", {"onstatuschange":
component.get("c.statusChange") },
            function (content, status) {
                if (status === "SUCCESS") {
                    component.find('overlayLib').showCustomModal({
                        body: content,
                        showCloseButton: true,
                        closeCallback: function () {
                            $A.get('e.force:refreshView').fire();
                        }
                    }).then(function(overlay) {
                        //\ Use to close the modal later
                        component.set("v.overlay", overlay);
                    });
                    content.resumeFlow(id);
                }
            });
```

```
},
   handleDelete: function (component, event, helper, id) {
       // On delete, pass the interview ID to the Apex controller
       var action = component.get("c.deleteInterview");
        action.setParams({
            interviewId: id
       });
        action.setCallback(this, $A.getCallback(function (response) {
            var state = response.getState();
            if (state === "SUCCESS") {
                // Automatically refresh the table
                helper.populateTable(component, event, helper);
            } else if (state === "ERROR") {
                var errors = response.getError();
                console.error(errors);
            }
        }));
        $A.enqueueAction(action);
   }
})
```

Customize Who Has Access to Paused Flow Interviews

By default, users can resume paused flow interviews as long as they have edit access. To control who has edit access, build a sharing model for the Flow Interview object. Configure the org-wide default access level, and build sharing rules to override that default for specific users or groups.

🕜 Note:

- The default sharing model for interviews is Private, which means that users inherit edit access from users lower in the role hierarchy. If your org uses a role hierarchy, users can resume all interviews that users lower in the hierarchy own or have edit access to.
- Users with the CEO role have read/write access to all flow interviews in the org, even if the interview owner isn't part of the hierarchy.

Example: To let all agents in your org resume any interview:

- 1. Add all agents to the Agents public group.
- 2. For Flow Interview, leave the organization-wide default set to Private.
- **3.** In a flow interview sharing rule, give read/write access (1) for interviews owned by internal users (2) to the Agents public group (3).

EDITIONS

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

Step 1: Rule Name						
Label	Share All with Agents					
Rule Name						
Description						
Step 2: Select your rule type						
Rule Type	 Based on record owner Based on criteria 					
Step 3: Select which records to be shared						
Flow Interview: owned by members of	Public Groups ᅌ All Internal Users ᅌ 2					
Step 4: Select the users to share with						
Share with	Public Groups ᅌ Agents ᅌ 3					
Step 5: Select the level of access for the users						
Access Level	Read/Write ᅌ 📘					

SEE ALSO:

Restrict Who Can Resume Shared Flow Interviews Sharing Considerations

Restrict Who Can Resume Shared Flow Interviews

Users can resume all paused flow interviews that they have edit access to. To restrict who can resume interviews in your org, disable Let users resume shared flow interviews.

User Permissions Needed		
To edit process automation settings:	Customize Application	Experie
When this setting is disabled a paused interview	w can be resumed only by the interview owner or	Availat Profes

When this setting is disabled, a paused interview can be resumed only by the interview owner or by a flow admin who has view access to the interview.

- 1. From Setup, enter *Automation* in the Quick Findbox, then select **Process Automation** Settings.
- 2. Deselect Let users resume shared flow interviews.
- 3. Click Save.

SEE ALSO:

Customize Who Has Access to Paused Flow Interviews

EDITIONS

Available in: both Salesforce Classic and Lightning Experience
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 Aura 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 Aura component or Visualforce page. Flows in custom Aura 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 Lightning Bolt Solutions, change sets, and packages. The recipient organization of the solution, change set, or package must have flows 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).

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

Survey Customers	Survey Customers
Name Rate Product Rate Customer Service Nedes Improvement	Name
Other Comments	*Rate Product Satisfied
Previous Finish	Rate Customer Service Excellent No Opinion Needs Improvement
	Other Comments
	Previous Finish

Which Runtime Experience Do My Users See?

Flows that run from a Visualforce component always use Classic runtime. Flows that run from a Lightning page, flow action, or custom Lightning component always use Lightning runtime. All other methods depend on whether Lightning runtime has been enabled in your org's Process Automation settings.

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

Flow Distribution Method	When Lightning Runtime for Flows is		
	Not selected	Selected	
Visualforce component	Classic runtime	Classic runtime	
Custom button	Classic runtime	Lightning runtime	
Custom link	Classic runtime	Lightning runtime	
Web tab	Classic runtime	Lightning runtime	
Direct link	Classic runtime	Lightning runtime	
Flow action	Lightning runtime	Lightning runtime	
Lightning page	Lightning runtime	Lightning runtime	
Custom Lightning component	Lightning runtime	Lightning runtime	

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 Aura component.

IN THIS SECTION:

Embed a Flow in a Lightning Page

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

Add a Flow to the Actions & Recommendations Component

Want to use your flow to guide users through complex business processes in Lightning console or standard navigation apps? Associate the flow with records by using a process to create a

RecordAction record. Then add the Actions & Recommendations component to your Lightning pages using the Lightning App Builder.

Create a Flow Action

To easily distribute a flow to Lightning Experience or Salesforce app users, create a flow action and add it to the appropriate page layout. Flows aren't supported for global actions.

Add a Flow to the 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 Aura Component

To customize how your flow gets and receives data, add it to a custom Aura component. Then distribute that component through a custom action, Lightning tab, or Lightning page.

Embed a Flow in a Lightning Page

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

Available in: Lightning Experience

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

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

1. Open a Lightning page in the Lightning App Builder.

USER PERMISSIONS

To create and save Lightning pages in the Lightning App Builder:

Customize Application

To view Lightning pages in the Lightning App Builder:

 View Setup and Configuration

Available in: both Salesforce Classic and Lightning Experience

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

- 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 screen flows 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 Lightning Flow 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
 - Record variables
 - Record collection variables
- The component supports only manually entered values for input variables.
- Text input variables accept a maximum length of 4,000 characters.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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:

Flow Limits and Considerations Lightning Pages Lightning App Builder Considerations Considerations for Two-Column Flows

Add a Flow to the Actions & Recommendations Component

Want to use your flow to guide users through complex business processes in Lightning console or standard navigation apps? Associate the flow with records by using a process to create a RecordAction record. Then add the Actions & Recommendations component to your Lightning pages using the Lightning App Builder.

When users open a record that's associated with flows, the flows are added to the Actions & Recommendations component. The first flow launches as a subtab in your console app, or in a popup window in a standard navigation app. The Actions & Recommendations component is great for call scripts or chat interactions.

SEE ALSO:

Lightning Flow for Service Developer Guide (English only)

Create a Flow Action

To easily distribute a flow to Lightning Experience or Salesforce app users, create a flow action and add it to the appropriate page layout. Flows aren't supported for global actions.

Available in: Lightning Experience

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

USER PERMISSIONS

To create actions:

Customize Application

1. From the management settings for the object for which you want to create an action, go to Buttons, Links, and Actions.

- 2. Click New Action.
- 3. For Action Type, select Flow.
- 4. Select the flow to use in your action.

The flow must be active and of type "Screen Flow" or "Field Service Mobile Flow".

Note: This release contains a beta version of Field Service Mobile flow actions that is production quality but has known limitations. To access this feature, contact Salesforce. To provide feedback and suggestions, go to IdeaExchange.

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.

- Optionally, change the action icon to a static resource in your org.
 Custom images used for action icons must be less than 1 MB in size.
- **9.** Save the action.

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

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

Variable Setting	Value
Unique Name	recordId (case sensitive)
Data Type	Text
Input/Output Type	Input OnlyInput and Output

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

Keep these considerations in mind before using flow actions.

Flow Action Considerations

Keep these considerations in mind before using flow actions.

Flow Type

Flow actions support only flows that include screens.

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 let you pass the value of the record's ID field into the flow, but that's it. If your flow has a Text input variable called recordId, the action passes the record's ID into that variable at runtime. If not, it doesn't and the flow tries to run anyway.

EDITIONS

Available in: both the Salesforce app and Lightning Experience

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

Help Text

A flow action's screen-level help text isn't available for feed-based page layouts.

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: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

- 1. From Setup, enter App 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.
- 3. Under App Settings, click Utility Items.
- 4. Click Add Utility Item, and select Flow.
- 5. Configure the utility item properties and the component properties.

Flow

Only active screen flows are available. Flows that were built in the Desktop Flow Designer aren't supported.

Layout

By default, flows display in one column.

6. 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 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:

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To view flow detail pages:

Manage Flow

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.

U	ser Permissions Needed	
To edit process automation setting	s: Customize Application	

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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:

Cloud Flow Designer Guide

- 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 FirstNameand 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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

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.
- 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.

Classic and Lightning Experience

Available in: both Salesforce

EDITIONS

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

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 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

Variable Type	Acceptable Values
Boolean	 Merge field of type Checkbox True values: true or 1 False values: false or 0
Currency	Merge field of type Number or a numeric value
Date	Merge field of type Date or YYYY-MM-DD
DateTime	Merge field of type Date/Time or YYYY-MM-DDThh:mm:ssZ

Variable Type	Acceptable Values
Multi-Select Picklist	Merge field of any type or a string in this format: value1; value2
Number	Merge field of type Number or a numeric value
Picklist	Merge field of any type or a string
Text	Merge field of any type or a string

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**:

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

/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
 (lightning/page/home), even if the user has Salesforce Classic enabled. Users who don't have permission to access Lightning
 Experience see an error message.
- If your URL redirects users to a web tab, Salesforce renders the web tab in Salesforce Classic.
- Web tabs in Lightning Experience can redirect only to Visualforce pages.

Redirect Destination	Relative URL	Example
Chatter	_ui/core/chatter/ui/ChatterPage	_ui/core/chatter/ui/ChatterPage
Home page	home/home.jsp	home/home.jsp
List view	objectCode?fcf=listViewId	006?fcf=00BD0000005lwec
Object home page, such as Accounts home	objectCode/0	001/o
Specific record, such as a contact, report, dashboard, user, profile, or Chatter post	recordId	0D5B000000SKZ7V
Visualforce page	apex/pageName	apex/myVisualforcePage
Web tab	servlet/servlet.Integration?lid=webTabId	servlet/servlet.Integration?lid=01rD000000A88h

Limitations

- You can't use a flow variable as the value for the retURL parameter. If you want to use a flow variable to redirect a user, such as to a specific record, distribute the flow by using Visualforce.
- retURL can cause nested top and side navigation bars to render on the destination page.
- retURL is case-sensitive. If you use retUr1, the URL doesn't redirect the user.

Examples

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

/flow/myFlow?retURL=001/o

When Lightning Experience users finish the flow interview, Salesforce redirects them to

http://yourInstance.lightning.force.com/lightning/o/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.

For instance, after viewing the Visualforce page, users navigate to the home page. For Lightning Experience users, Salesforce renders the Lightning Experience home page (http://yourInstance.lightning.force.com/lightning). For Salesforce Classic users, Salesforce renders the Salesforce Classic home page (http://yourInstance.salesforce.com/home/home.jsp).

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

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

SEE ALSO:

Distribute 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.
- Add the <flow:interview> component somewhere between the <apex:page> tags.
- 5. Set the name attribute to the unique name of the flow.

For example:

```
<apex:page>
<flow:interview name="flowuniquename"/>
</apex:page>
```

If the flow is from a managed package, the name attribute must be in this format: namespace.flowuniquename.

- 6. Click Save.
- 7. Restrict which users can access the Visualforce page.
 - a. Click 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 Lightning Platform app by using a custom button, link, or Visualforce tab.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To create, edit, and set version settings for Visualforce pages:

Customize Application

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, record variables, collection variables, and record 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 record variables. To display values for a collection variable or a record 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, record variables, collection variables, and record 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. 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, record variable, and record collection variable values using Visualforce.

Method	Variables	Record Variables	Collection Variables	Record Collection Variables
Without a controller	~			
With a standard controller	~			
With a standard List controller				~
With a custom Apex controller	V	<		~
With an Interview Map	~	×	~	~

Setting Variable Values without a Controller

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

```
<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.

Setting a Record Collection Variable Value with a Standard List Controller

Because record 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.

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 ld when the interview starts.

```
public 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:page controller="MyCustomController">
        <flow:interview name="flowname">
        </apex:param name="myVariable" value="{!apexVar}"/>
        </flow:interview>
<//apex:page>
```

This example uses Apex to set a record collection variable myAccount to the Id and Name field values for every record with a Name of *Acme*.

```
public class MyCustomController {
   public Account[] myAccount {
      get {
         return [
            SELECT Id, Name FROM account
            WHERE Name = 'Acme'
            ORDER BY Id
   ];
```

```
}
set {
    myAccount = value;
}
public MyCustomController () {
}

</apex:page id="p" controller="MyCustomController">
    <flow:interview id="i" name="flowname">
        <apex:param name="accountColl" value="{!myAccount}"/>
        </flow:interview>
<//apex:page>
```

Setting Variable Values with an Interview Map

This example uses an Interview map to set the value for accVar to a specific account's ld when the interview starts.

```
<apex:page controller="MyCustomController">
<flow:interview name="flowname" interview="{!myflow}"/>
</apex:page>
```

Here's a similar example that sets the value for accvar to a new account when the interview starts.

```
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);
   }
}
```

```
<apex:page controller="MyCustomController">
<flow:interview name="flowname" interview="{!myflow}"/>
</apex:page>
```

This example uses a map to add two values to a string collection variable (stringCollVar) and two values to a number collection variable (numberCollVar).

```
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 record variables. To display values for a collection variable or a record 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. 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 a record variable value from a flow and then displays it in a Visualforce page.

```
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.

```
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;
        }
   }
}
```

```
}
```

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 record collection variable and display the values for each item in the collection.

```
public class MyCustomController {
  public Flow.Interview.flowname myflow { get; set; }
}
<apex:page controller="MyCustomController" tabStyle="Account">
   <flow:interview name="flowname" interview="{!myflow}" reRender="nameSection" />
    <!-- The data table iterates over the variable set in the "value" attribute and
         sets that variable to the value for the "var" attribute, so that instead of
         referencing {!myflow.collectionVariable} in each column, you can simply refer
         to "account".-->
    <apex:dataTable value="{!myflow.collectionVariable}" var="account"</pre>
        rowClasses="odd, even" border="1" cellpadding="4" id="nameSection">
        <!-- 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>
</apex:page>
```

Depending on the contents of the record collection variable in your flow, here's what that data table looks like.

Name	Rating	Billing City	Employees
<u>Global Media</u>	Hot	Toronto	14668
ABC Labs	Warm	San Jose	120
<u>Canson</u>	Hot	Ohta-ku, Tokyo	125
Acme Inc.	Hot	Atlanta	680
Ecotech - Switzerland	Cold	Geneva	3500
Informatica Global	Warm	Buenos Aires	300
Lutron Technologies	Hot	Murray Hill	200
Sapient-UK	Cold	London	80
<u>Targas</u>	Warm	Anaheim	1200

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:page>
    <flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/home/home.jsp')}"/>
</apex:page>
```

This example routes users to a detail page with an ID of 001D000000IpE9X.

```
<apex:page>
<flow:interview name="MyUniqueFlow" finishLocation="{!URLFOR('/001D000000IpE9X')}"/>
</apex:page>
```

For details about URLFOR, see Functions in the Visualforce Developer's Guide.

Set finishLocation with the \$Page Variable

To route users to another Visualforce page without using URLFOR, set finishLocation to the name of the destination page with the format {!\$Page.pageName}.

```
<apex:page>
        <flow:interview name="MyUniqueFlow" finishLocation="{!$Page.MyUniquePage}"/>
        </apex:page>
```

For details about \$Page, see Global Variables in the Visualforce Developer's Guide.

Set finishLocation with a Controller

You can set finishLocation in a few ways with a custom controller.

This sample controller configures a flow's finish behavior in three different ways.

- 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.

```
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.

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 Aura Component

To customize how your flow gets and receives data, add it to a custom Aura component. Then distribute that component through a custom action, Lightning tab, or Lightning page.

To embed a flow in your Aura component, add the lightning:flow component to it. In the JavaScript controller, identify which flow to start.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

```
<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 API Name.
        flow.startFlow("myFlow");
      },
})
```

Note: As of Spring '19 (API version 45.0), you can build Lightning components using two programming models: the Lightning Web Components model, and the original Aura Components model. Lightning web components are custom HTML elements built using HTML and modern JavaScript. Lightning web components and Aura components can coexist and interoperate on a page. This developer guide covers Aura components only.

IN THIS SECTION:

Reference Flow Output Variable Values in a Wrapper Aura Component

When you embed a flow in an Aura component, you can display or reference the flow's variable values. Use the onstatuschange action to get values from the flow's output variables. Output variables are returned as an array.

Set Flow Input Variable Values from a Wrapper Aura Component

When you embed a flow in a custom Aura component, give the flow more context by initializing its 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 by Wrapping the Flow in a Custom Aura Component

By default, when a flow user clicks **Finish**, a new interview starts and the user sees the first screen of the flow again. By embedding a flow in a custom Aura component, 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.

Reference Flow Output Variable Values in a Wrapper Aura Component

When you embed a flow in an Aura component, you can display or reference the flow's variable values. 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. 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.

```
<aura:component>
```

```
<aura:attribute name="accountName" type="String" />
<aura:attribute name="numberOfEmployees" type="Decimal" />
<lightning:formattedText value="{!v.accountName}" />
<lightning:formattedNumber style="decimal" value="{!v.numberOfEmployees}" />
```

```
<aura:handler name="init" value="{!this}" action="{!c.init}"/>
```

```
<lightning:flow aura:id="flowData" onstatuschange="{!c.handleStatusChange}" />
</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 API 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++) {</pre>
            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 Input Variable Values from a Wrapper Aura Component

When you embed a flow in a custom Aura component, give the flow more context by initializing its 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. 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 API name for the flow data type. For example, for a record variable use SObject, and for a text variable use String.

```
{
    name : "varName",
    type : "flowDataType",
    value : valueToSet
},
{
    name : "varName",
    type : "flowDataType",
    value : [ value1, value2]
}, ...
```

}

Example: This JavaScript controller sets values for a number variable, a date collection variable, and a couple of record variables. The Record data type in Flow Builder corresponds to SObject here.

```
( {
  init : function (component) {
     // 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 record (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 record (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 record variable through the JavaScript controller.

```
<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 record (sObject) variable to the value of the component's
            // account attribute.
            var inputVariables = [
               {
                  name : "account",
                  type : "SObject",
                  value: component.get("v.account")
               }
            1;
            // In the component whose aura:id is "flowData, start your flow
            // and initialize the account record (sObject) variable. Reference the
flow's
            // API name.
            flow.startFlow("myFlow", inputVariables);
         }
         else {
            console.log("Failed to get account date.");
         }
      });
      // Send action to be executed
      $A.engueueAction(action);
   }
})
```

Control a Flow's Finish Behavior by Wrapping the Flow in a Custom Aura Component

By default, when a flow user clicks **Finish**, a new interview starts and the user sees the first screen of the flow again. By embedding a flow in a custom Aura component, 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.

() Tip: To control a flow's finish behavior at design time, make your custom Aura component available as a flow action by using the lightning:availableForFlowActions interface. To control what happens when an autolaunched flow finishes, check for the FINISHED_SCREEN status.

```
// init function here
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. handleStatusChange : function (component, event) { if(event.getParam("status") === "FINISHED") { var outputVariables = event.getParam("outputVariables"); var outputVar; for(var i = 0; i < outputVariables.length; i++) {</pre> 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 Aura component or Visualforce page. Flows in custom Aura components use Lightning runtime, and flows in Visualforce pages use Classic runtime.

For example, set up a self-service tool for your 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 Aura Component for External Users

To control how a flow behaves in external contexts and use Lightning runtime, create a custom Aura component. Then distribute the component externally, such as through a Lightning community or a Lightning Out-enabled app.

Embed a Flow in a Visualforce Page for External Users

Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

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:

- 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 screen flows 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 Aura Component for External Users

To control how a flow behaves in external contexts and use Lightning runtime, create a custom Aura component. Then distribute the component externally, such as through a Lightning community or a Lightning Out-enabled app.

Note: As of Spring '19 (API version 45.0), you can build Lightning components using two programming models: the Lightning Web Components model, and the original Aura Components model. Lightning web components are custom HTML elements built using HTML and modern JavaScript. Lightning web components and Aura components can coexist and interoperate on a page. This developer guide covers Aura components only.

Create a custom Aura 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.

```
<aura:component access="GLOBAL">
<aura:handler name="init" value="{!this}" action="{!c.init}" />
```

EDITIONS

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

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

USER PERMISSIONS

To create, customize, or publish a community:

 Create and Set Up Communities AND View Setup and Configuration

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

```
<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 API Name.
        flow.startFlow("myFlow");
    },
})
```

Lightning Communities

To expose your custom Aura 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 Aura component in an app outside of Salesforce servers, use Lightning Out. Create a Lightning dependency app to describe the component dependencies.

```
<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.

```
...
<script src="https://myDomain.my.salesforce.com/lightning/lightning.out.js"></script>
<script>$Lightning.use("c:lightningOutApp", function() {
    $Lightning.createComponent("lightning:flow", {},
    "flowContainer",
    function (component) {
      component.startFlow("Survey_customers", inputVariables);
    })
  ;;
});
</script>
```

SEE ALSO:

Lightning Aura Components Developer Guide : Configure Components for Communities Lightning Aura Components Developer Guide : Add Aura Components to Any App with Lightning Out (Beta)

Embed a Flow in a Visualforce Page for External Users

Let external users run your flow by adding the flow to a Visualforce page and distributing that page externally. For example, through a community.

For example, you can set up a self-service tool for your public Salesforce site to help visitors generate custom sales quotes. Because the flow is embedded in a Visualforce page, you can customize the appearance of the flow so that it uses your company's branding and style.



Note: When you make a flow available to site or portal users, point them to the Visualforce page that contains the embedded flow, not the flow itself. Site and portal users aren't allowed to run flows directly.

To add a flow to a Visualforce page, embed it by using the <flow:interview> component.

- 1. Find the flow's 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:

```
<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 Salesforce 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.

USER PERMISSIONS

To create, edit, and set version settings for Visualforce pages:

Customize Application

Most of these methods can be used only with an autolaunched flow. A *flow* can be launched without user interaction, such as from a process or the Apex interview.start method. Autolaunched flows run in bulk and without user interaction. They can't contain steps, screens, choices, or dynamic choices in the active or latest flow version. When a flow user invokes an autolaunched flow, the active flow version is run. If there's no active version, the latest version is run. When a flow admin invokes a 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 Lightning Platform REST API

Use the Custom Invocable Actions endpoint to invoke an autolaunched flow from the Lightning Platform 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 "Flows" action to the process.
 - **a.** For Flow, search for and select the flow that you created.
 - b. Optionally, click Add Row to set values for the flow's variables.
- 4. Activate the process.

USER PERMISSIONS

To create, edit, or view processes:

Manage Flow
 AND
 View All Data

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, admins run the latest flow version while other users run the active flow version.
- Flows that are launched from workflow rules are run in system context, which means that user permissions, field-level security, and sharing rules aren't 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 in the flow, with the following limitations.
 - Flow triggers can't use multi-select picklist fields to set flow variables.
 - When a flow trigger uses a currency field to set a flow variable, only the amount is passed into the flow. Any currency ISO code or locale information is ignored. If your organization uses multiple currencies, the flow trigger uses the amount in the currency of the record that contains the specified currency field.
 - Flow triggers can't pass values into record collection variables in flows.

EDITIONS

Available in: Salesforce Classic

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

USER PERMISSIONS

To view workflow rules and actions:

 View Setup and Configuration

To create or change workflow rules and actions:

Customize Application

EDITIONS

Available in: Salesforce Classic

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

- 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: Create Records, Update Records, or Delete Records.

For example, suppose that you use Data Loader or the Bulk API to update 50 records, and those updates meet the criteria of a workflow rule with a flow trigger action. In response, the system executes 50 instances of the flow within the same transaction. Each instance of a running flow is called an interview. The system attempts to execute each DML operation across all the interviews in the transaction at the same time. Suppose that five of those interviews are executing the same branch of the flow, which has an Update Records element called "SetEntitlement." The system waits for all five interviews to reach that element, and then executes all five record updates in bulk.

- Flow triggers aren't available in change sets.
- Flow triggers aren't packageable.

Define a Flow Trigger—Pilot

After you create an autolaunched flow, create a flow trigger to launch that flow as part of a workflow rule.

- Note: The pilot program for flow trigger workflow actions is closed. If you've already enabled the pilot in your org, you can continue to create and edit flow trigger workflow actions. If you didn't enable the pilot in your org, use 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.

Field	Description
Name	Name of the flow trigger.
Unique Name	Enter a unique name to refer to this component in the API. The Unique Name field can contain only underscores and alphanumeric characters. It must be unique within the selected object type, begin with a letter, not include spaces, not end with an underscore, and not contain two consecutive underscores.

EDITIONS

Available in: Salesforce Classic

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

USER PERMISSIONS

To view workflow rules and actions:

 View Setup and Configuration

To create or change workflow rules and actions:

Customize Application

Field	Description
Protected Component	Reserved for future use.
Flow	Unique name of the autolaunched flow that this workflow action launches.
Set Flow Variables	Whether to pass values into the flow's variables.

5. If you select Set Flow Variables, specify their names and values.

Click Set Another Value to set up to 25 variables.

6. To put the flow trigger in test mode, select Administrators run the latest flow version.

When selected and an admin triggers the workflow rule, the flow trigger launches the latest version of the flow. For all other users, the flow trigger always launches the active version of the flow.

The same values are passed into the flow variables whether the flow trigger launches the active or latest flow version.

7. Click Save.

Don't forget to associate the flow trigger to a workflow rule.

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.

- 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.

EDITIONS

Available in: Salesforce Classic

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

USER PERMISSIONS

To select existing actions:

Customize Application

Invoke a Flow from the Lightning Platform REST API

Use the Custom Invocable Actions endpoint to invoke an autolaunched flow from the Lightning Platform 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" : "0D5D000000cfMY"
    } ]
}
```

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:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></apex:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel></appx:outputLabel>
```

```
<apex:form >
    value : <apex:inputText value="{!output}"/>
        <apex:commandButton action="{!start}" value="Start" reRender="text"/>
        </apex:form>
    </apex:page>
```

IN THIS SECTION:

start()

Starts an instance (interview) for an autolaunched 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()

Starts an instance (interview) for an autolaunched or user provisioning flow.

Signature

public Void start()

Return Value

Type: Void

Usage

This method can be used only with flows that have one of these types.

- Autolaunched Flow
- User Provisioning Flow

For details, see "Flow Types" in Salesforce Help.

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 a flow, the latest version is always run.

getVariableValue(variableName)

Returns the value of the specified flow variable. The flow variable can be in the flow embedded in the Visualforce page, or in a separate flow that is called by a subflow element.

Signature

public Object getVariableValue(String variableName)

Parameters

variableName Type: String Specifies the unique name of the flow variable.

Return Value

Type: Object

Usage

The returned variable value comes from whichever flow the interview is 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 Lightning Bolt Solutions, change sets, and packages. The recipient organization of the solution, change set, or package must have flows enabled.

IN THIS SECTION:

Deploy Flows in a Lightning Bolt Solution

To distribute automated business processes or bootstrap a community with a complete solution or new look, create a Lightning Bolt Solution. A Lightning Bolt Solution can include flows, custom Lightning apps, or Lightning Community templates or pages. Before you create a solution, group flows that you want to include in a flow category.

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.

Deploy Processes and Flows as Active

By default in production orgs, active processes and flows are deployed as inactive. After deployment, you manually reactivate the new versions. If your production org uses a continuous integration and continuous delivery model to deploy metadata changes, enable the option to deploy processes and flows as active. This option applies to processes and autolaunched flows that are deployed in production orgs via change sets and Metadata API.

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.

EDITIONS

Available in: Lightning Experience and Salesforce Classic

Lightning Bolt Solutions are available in: **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Change sets are available in: **Professional**, **Enterprise**, **Performance**, **Unlimited**, and **Developer** Editions

Packages are available in: Essentials, Professional, Enterprise, Performance, Unlimited, and Developer Editions

Deploy Flows in a Lightning Bolt Solution

To distribute automated business processes or bootstrap a community with a complete solution or new look, create a Lightning Bolt Solution. A Lightning Bolt Solution can include flows, custom Lightning apps, or Lightning Community templates or pages. Before you create a solution, group flows that you want to include in a flow category.

- 1. Add your flows to a flow category. You can only add active flows.
- **2.** Create your Lightning Bolt Solution. Add one or more flow categories, custom Lightning apps, and Lightning Community templates or pages.
- 3. Package the solution to distribute it to your own orgs, or share or sell it on AppExchange.

SEE ALSO:

Salesforce Help: Add Flows to a Lightning Bolt Solution Salesforce Help: Lightning Bolt for Salesforce: Build Once, Then Distribute and Reuse

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

EDITIONS

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

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

USER PERMISSIONS

To create a flow category:

Customize Application

To create a Lightning Bolt Solution without a community template:

Customize Application
 AND Customize
 Application AND View
 Setup and Configuration

To create a Lightning Bolt Solution with a community template:

 Customize Application AND Create and Set Up Communities AND View Setup and Configuration

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Available in **Enterprise**, **Performance**, **Professional**, **Unlimited**, and **Database.com** Editions
- Submit for Approval

For example, if you deploy a flow that includes a Submit for Approval element, manually add the referenced approval process.

 If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package or change set automatically.

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

Deploy Processes and Flows as Active

By default in production orgs, active processes and flows are deployed as inactive. After deployment, you manually reactivate the new versions. If your production org uses a continuous integration and continuous delivery model to deploy metadata changes, enable the option to deploy processes and flows as active. This option applies to processes and autolaunched flows that are deployed in production orgs via change sets and Metadata API.

User Permissions Needed		
To edit process automation settings:	Customize Application	

EDITIONS

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

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

- From Setup, enter Automation in the Quick Find box, then select Process Automation Settings.
- 2. Select Deploy processes and flows as active.

Note: This setting is available only in production orgs. Scratch, sandbox, and developer orgs don't have the setting because you can always deploy a new active version.

3. Save your changes.

Before you can successfully deploy a process or autolaunched flow as active, your production org must meet flow test coverage requirements. 75% of the active processes and autolaunched flows in your org must be covered by at least one Apex test. The required percentage is the same as the code coverage requirement for Apex code. Flow test coverage requirements don't apply to flows that have screens.

To calculate your org's flow test coverage, determine the number of active autolaunched flows and processes in your org. Here's a sample query.

```
SELECT count_distinct(DefinitionId)
FROM Flow
WHERE Status = 'Active'
AND (ProcessType = 'AutolaunchedFlow'
OR ProcessType = 'Workflow'
OR ProcessType = 'CustomEvent'
OR ProcessType = 'InvocableProcess')
```

Then run all tests, and use the FlowTestCoverage object in the Tooling API to determine the number of active autolaunched flows and processes that have test coverage. Here's a sample query.

```
SELECT count_distinct(FlowVersionId)
FROM FlowTestCoverage
```

Divide the second number (number of covered active autolaunched flows and processes) by the first number (number of active autolaunched flows and processes). For example, if you have 10 active autolaunched flows and processes, and 8 of them have test coverage, your org's flow test coverage is 80%.

```
SELECT FlowVersion.Definition.DeveloperName
FROM FlowTestCoverage
GROUP BY FlowVersion.Definition.DeveloperName
```

Tip: To get the names of all active autolaunched flows and processes that don't have test coverage, use this query.

```
SELECT Definition.DeveloperName
FROM Flow
WHERE Status = 'Active'
AND (ProcessType = 'AutolaunchedFlow'
OR ProcessType = 'Workflow'
OR ProcessType = 'CustomEvent'
OR ProcessType = 'InvocableProcess')
AND Id NOT IN (SELECT FlowVersionId FROM FlowTestCoverage)
```

SEE ALSO:

Tooling API: FlowTestCoverage

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

- Send Email
- Submit for Approval

For example, if you deploy a flow that posts to a particular Chatter group, manually add the referenced Chatter group to the package.

• If a flow references a Lightning component that depends on a CSP Trusted Site, the trusted site isn't included in the package or change set automatically.

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 either the user who installed the flow or the Apex exception email recipients.
- 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 Control Who Receives Flow and Process Error Emails

Why Did My Flow Interview Fail?

To troubleshoot a failed flow interview, use the flow fault email. To test the flow and observe what happens as it runs, use the debug option in Cloud Flow Designer. Or add temporary Screen or Send Email elements to the flow.

IN THIS SECTION:

Emails About Flow Errors

Every time a flow interview fails, Salesforce sends an error email in the default language of the user who ran the flow. The email is sent to either the admin who last modified the associated flow or the Apex exception email recipients. The email includes the error message from the failure and details about each flow element that the interview executed.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Cloud Flow Designer Guide

Debug a Flow in Cloud Flow Designer

Whether you're testing a new flow or troubleshooting a flow that fails, the debug option in Cloud Flow Designer can be your best friend. See real-time details of what your flow does, easily set input variables, and restart the flow anytime to debug a different branch.

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, Salesforce sends an error email in the default language of the user who ran the flow. The email is sent to either the admin who last modified the associated flow or the Apex exception email recipients. The email includes the error message from the failure and details about each flow element that the interview executed.

If the interview failed at multiple elements, the recipients receive 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.



Note: Process and flow error emails include the data that's involved in the process or flow, including user-entered data.

Example:

An error occurred at element Apex_Plug_in_1. List index out of bounds: 0. An error occurred at element 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

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

Customize What Happens When a Flow Fails

- Why Did My Flow Interview Fail?
- What Happens When an Apex Exception Occurs?
- Control Who Receives Flow and Process Error Emails

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Limitations of Emails About Flow Errors

The email about errors in flow interviews has some limitations for Screen, Lookup, Create, and Subflow elements—as well as some general limitations.

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 either the user who installed the flow or the Apex exception email recipients.

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? Control Who Receives Flow and Process Error Emails

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

Debug a Flow in Cloud Flow Designer

Whether you're testing a new flow or troubleshooting a flow that fails, the debug option in Cloud Flow Designer can be your best friend. See real-time details of what your flow does, easily set input variables, and restart the flow anytime to debug a different branch.

Warning: Debugging or testing a flow actually runs the flow and performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.

- **1.** Open the flow in Cloud Flow Designer.
- 2. Click Debug.
- 3. Set the debug options and input variables.

Debug the flow		
Debug options		
 Run the latest version of each flow called by subflow elements 		
Show details of what's executed and render flow in Lightning runtime		
Input variables Enter values for the flow's input variables. For each value left blank, the flow starts with the variable's default value. You can't enter values for collection or sObject variables.		
contact_first_name		
Howard		
contact_last_name Jones		
Run		

4. Click Run.

If you opt to show details, they appear in a panel on the right.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

To debug a flow in Cloud Flow Designer:

Manage Flow

	Run Again
Find Account From Contact	Debug Details
Howard Jones's account: Acme	How the Interview Started Admin User (005R000000UF6i) started the flow interview. Some of this flow's variables were set when the interview started. contact_first_name = Howard contact_last_name = Jones
	RECORD QUERY: Find_Contact_s_Account Find one Contact record where: FirstName Equals {!contact_first_name} (Howard) LastName Equals {!contact_last_name} (Jones) Result Successfully found record. {!account_id} = 001R000002ahRYIAY
	RECORD QUERY: Get_Account_Name Find one Account record where: Id Equals {!account_id} (001R0000002ahRYIAY) Result Successfully found record. {!account_name} = Acme

- 5. (Optional) To restart the flow using the same values for input variables that you entered earlier, click Run Again.
- 6. (Optional) To restart the flow with different values for input variables, finish the flow and click Change Inputs.

Considerations for Debugging Flows in Cloud Flow Designer

Before you use the debug option in Cloud Flow Designer, understand its limitations and special behaviors.

- Debugging or testing a flow actually runs the flow and performs its actions, including any DML operations and Apex code execution. Remember, closing or restarting a running flow doesn't roll back its previously executed actions, callouts, and changes committed to the database.
- You can't pass values into input variables of type collection, sObject, and sObject collection.
- Clicking **Pause** or executing a Wait element closes the flow and ends debugging.
- When you click **Finish** in a flow, the debug details incorrectly state "Selected Navigation Button: NEXT."

SEE ALSO: Flow Limits and Considerations Flows in Transactions

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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 flow from the Build a Discount Calculator project on *Trailhead*.

```
RESOURCE VALUES for "Calculate Discounts on Opportunities"
opptyID: {!opptyID}
AccountID: {!AccountID}
AccountRevenue: {!AccountRevenue}
Full_Discount outcome: {!Full_Discount}
Partial_Discount outcome: {!Partial_Discount}
Discount: {!Discount}
ERROR
{!$Flow.FaultMessage}
```

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.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

USER PERMISSIONS

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

Manage Flow



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?

Flow Terminology

The following terminology applies to flow 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 Flow Builder.

Flow Interview

A flow interview is a running instance of a flow.

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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

EDITIONS

Available in: both Salesforce Classic and Lightning Experience

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.

SEE ALSO:

Cloud Flow Designer Cloud Flow Designer

INDEX

С

Components flow, finish behavior 236

F

Flow delivering to users 213 delivering to users, external 237–238, 240 delivering to users, internal 215, 217, 225, 232, 238 embedding in community pages 238 embedding in Lightning components 232, 238 embedding in Lightning page 215, 217 embedding in Visualforce pages 225, 240 launching from processes 241 process action 241 Flow URL setting flow variable values 222 Flow variable values setting via URL 222

L

Lightning App Builder embedding flows 215, 217 Lightning communities embedding flows 238 Lightning components embedding flows 232, 238

V

Visualforce embedding flows 225, 240 Voice create permission set 219

Flow trigger

associated with workflow rule 244

sharing 213, 215, 217, 225, 232, 238, 240