MODULE

[FLOW BUILDER]

Flow Builder- Agenda

1. Learn About Flow Resources and Variables
2. Create a Variable
3. Add Screens to Your Flow
4. Add Logic to Your Flow
5. Add Actions to Your Flow

Learn About Flow Resources and Variables

Learning Objectives

After completing this unit, you'll be able to:

* List the resources available in Flow Builder.
* Describe what a flow variable is.

Before You Start

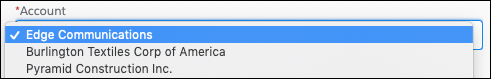
Before you can complete this module, make sure that you complete the Build a Simple Flow project. The concepts here depend on the flow you build in that project.

Flow Resources

In flows, resources are placeholders similar to merge fields in an email template or a formula. Let's say you start an email with Hi, {!$User.FirstName}. {!$User.FirstName} is a placeholder, so when the email is sent, it displays the actual first name of the user. In each step of the flow (the elements added to the canvas), you can reference flow resources instead of manually entering values.

Let's go over the basic kinds of flow resources available in Flow Builder.

* Constant represents a fixed value, such as a tax rate.
* Choice, Picklist Choice Set, or Record Choice Set represents an option in a screen component. With the choice resource, you explicitly set each option's label and value. Choice sets, on the other hand, generate choices for you by using a filtered list of records or the values of a picklist (or multi-select picklist) field in your org.



Example: In the New Contact flow, the screen prompts the user to select the associated account.

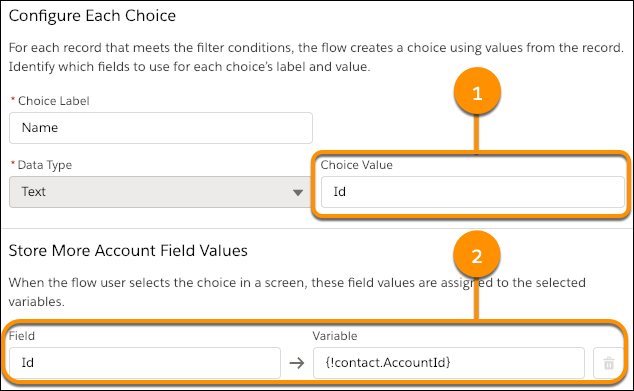
A screen component labeled “Account” with Acme selected from the picklist

Rather than building individual options for each account in your org, a record choice set generates the options.

To generate an option for each account record in your org, the record choice set uses the default filters. Every choice consists of two components: a label to display in the screen component, and a value to use when the choice is referenced elsewhere in the flow. The way this record choice set is configured, the screen component displays the account name for each option.

When the flow user selects an account:

* The value of the screen component is set to the choice value (1): the ID of the selected account.
* The selected account ID is stored in the {!contact.AccountId} record variable (2).



* Formula represents a calculated value, similar to a formula field. For example, create a formula that calculates 30 days from today, then reference that formula to set an opportunity close date.
* Text Template represents some formatted text. For example, format the body of an email or Chatter post in a text template, then reference the text template in the appropriate action.
* Variable represents a value that can change throughout the flow. We're going to spend the rest of this unit diving deep on variables.

Introducing Variables

The word variables can make you think of algebra classes or writing code in some scary language like Apex. However, they're an important thing to understand when you're building flows. Don't worry, no programming (or math) experience required.

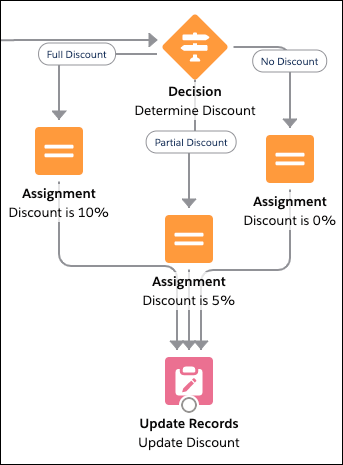
Simply put, a variable is a placeholder for a value you don't know yet. Every flow resource is a placeholder, but variables are the only resource that can change during the flow, hence the name “variable.” In fact, Flow Builder includes the Assignment element just for updating the values of variables. Watch a quick video for more information on what a variable is.

When Do I Create a Variable?

Use a variable when a value can differ based on certain conditions. Perhaps the value depends on which record the flow operates on, or perhaps it depends on the result of some logic in the flow.

You can't reference a field from a Salesforce record directly, so the field value must be stored in the flow using a variable. Flow elements that can pull data into the flow, such as a Get Records element or a Post to Chatter core action, always prompt you to store that data in a variable.

Example: In the Build a Discount Calculator project, the flow updates an opportunity's Discount field. But not every opportunity gets the same discount; it's determined by the associated account's revenue. A variable acts as a placeholder for the discount percentage and is set to a different percentage based on the flow logic. The flow then uses the variable to update the opportunity's discount.



Now that you're more familiar with the resources available in Flow Builder, let's put that into practice by creating a variable.

**Create a Variable**

**Learning Objectives**

After completing this unit, you'll be able to:

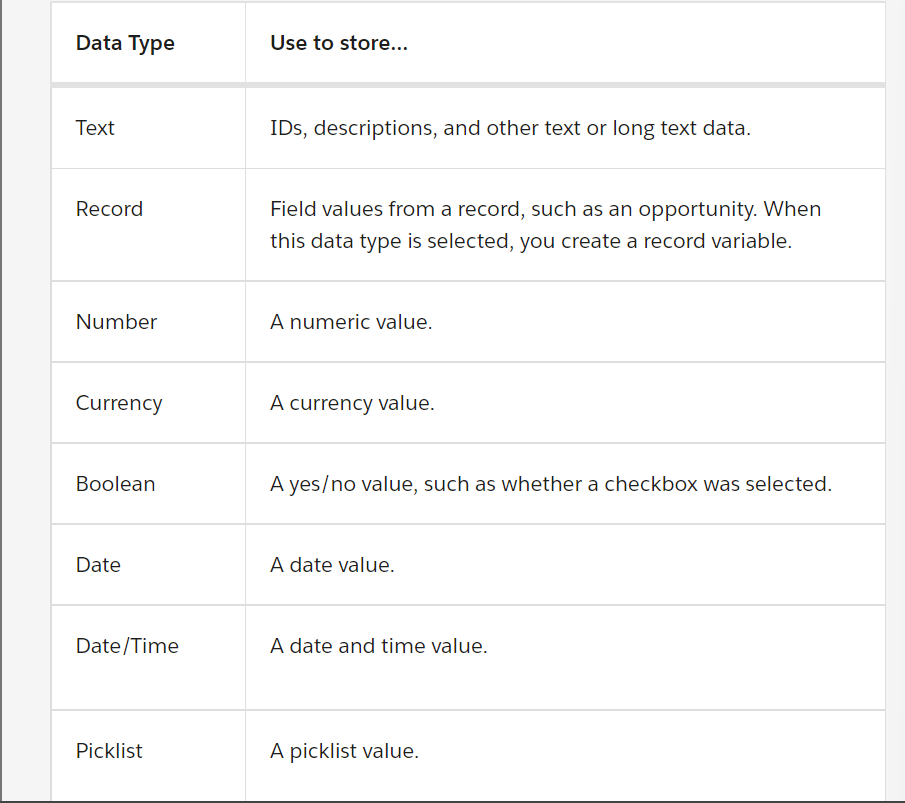
* Create a flow variable.
* Define input and output variables.

**Prepare to Create a Variable**

Before you create your variable, figure out what kind of variable you need.

**What type of data should the variable store?**

First consider what type of data the variable will store, much like you do when you create a custom field.



**How many values should the variable store?**

By default, variables store one value that's compatible with the selected data type. If the data type is Number, the variable stores one numeric value. If the data type is Record, the variable stores field values for one record.

But what if you need to store multiple values in one variable, such as multiple email addresses? When you create a variable, you can enable it to do just that with the Allow multiple values checkbox. When that option is selected, you create a collection variable.

**Should the value be available from outside the flow?**

Each variable in your flow gets its value from somewhere.

Some variables get their values from inside the flow, such as when the user enters something in a screen component or the flow looks up field values from a record.

Other variables get their values from outside of the flow—namely, from whatever started the flow. For example, if you distribute a flow on an Account record page, you can pass the account's ID into the flow by using the flow's input variables. An input variable is a variable that has the Available for input checkbox selected.

If whatever starts the flow should be able to set the value of a variable, make sure the variable is available for input.

When you configure your variable, you see another checkbox: Available for output. When selected, you've created an output variable, which means the value is accessible to something outside the flow. Output variables are too advanced for this module, so we won't go into much detail about them here. When in doubt, don't make your variable available for output.

**Where do you plan to use the variable?**

If you already know where you plan to use the variable, review documentation to see whether that field requires a certain kind of variable.

Tip: Pay special attention to which data types the field supports and whether it supports single values or collection values.

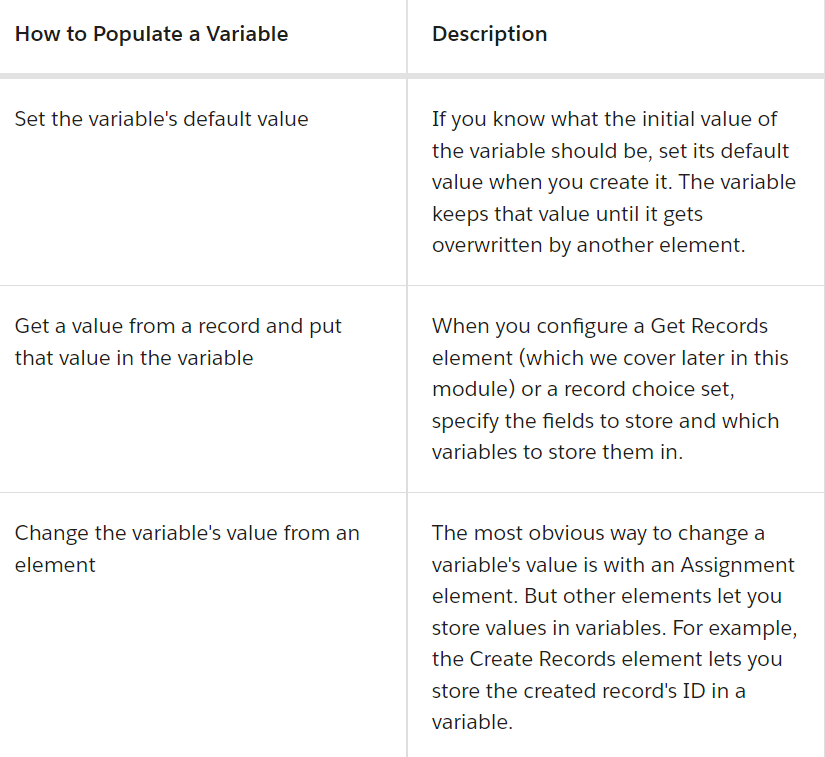
For example, you need to automatically submit an opportunity for approval from a flow. Rather than using the default approval settings, you need the approval request to be sent to a specific user, so you need to set the Next Approver Ids field. When you look at the reference documentation for the Submit for Approval core action, you see that the Next Approver Ids field only accepts Text collection variables. So you need to store the ID of that user in a Text collection variable.

**Create a Variable**

1. From Setup enter Flows in the Quick Find box and click Flows.
2. Click New Flow.
3. Select Screen Flow and click Create.
4. From the toolbox, click Manager.
5. Click New Resource.
6. For Resource Type, select Variable.
7. Enter an API name and description for your variable.
8. Select the appropriate data type.
9. If you want to store multiple values (a collection), select Allow multiple values.
10. For record variables, select the object whose record values you plan to store.
11. Identify the variable's availability outside the flow.

**Store Information in a Flow Variable**

Once you've created a variable, you've got a placeholder just waiting for a value to… well, hold. A variable is only as good as the values stored in it, and Flow Builder offers a few ways to populate that variable. Let's review them.



Now that we've got variables under our belt, let's dig in to the element categories available in Flow Builder. First up: screens.

**Add Screens to Your Flow**

**Learning Objectives**

After completing this unit, you'll be able to:

* List the types of components you can add to screens.
* Add a confirmation screen to a flow**.**

**Introducing Screen Components**

Just as you configure the user experience of your record pages in Lightning App Builder, you use screens to do the same for your flow users.

Each screen is made up of one or more screen components. A screen component is a configurable, reusable element added to a screen.

Screen components are available in three categories.

* Input includes standard components that request information from the user.
* Display includes standard components that display information to the user.
* Custom includes components that you or someone else have created. Install them from AppExchange or a third-party library, or work with a developer to build your own.

**Choices in Screen Components**

Most standard input components request a value like a number or a paragraph of text. For Radio Buttons, Picklist, Checkbox Group, and Multi-Select Picklist components, the user instead chooses from a set of options. To identify the available options, select at least one choice or choice set resource.

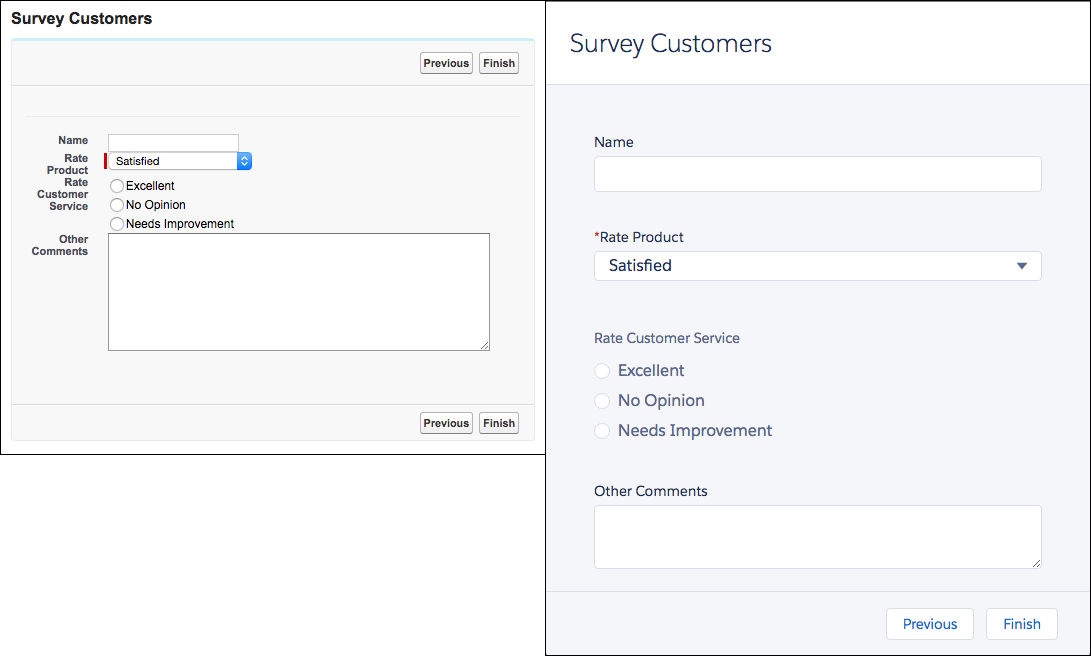
* A record choice set generates multiple options by using a filtered list of records.
* A picklist choice set generates multiple options by using the values of a picklist or multi-select picklist field.
* A choice represents a single option with the label and value set manually.

**Tip**: We recommend using a choice resource only when you can't use either of the other two. Record choice sets and picklist choice sets are easier to configure and don't require as much maintenance.

**The Runtime Experiences**

There's one more important consideration for screen components: which flow runtime experience they're supported in.

Flows have two different runtime experiences: Lightning runtime and Classic runtime. Like its name suggests, Lightning runtime looks and feels like Lightning Experience, while Classic runtime looks and feels like Visualforce.



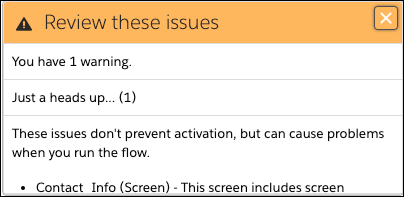
That said, the two runtime experiences aren't tied to either desktop experience. You can use Lightning runtime in Salesforce Classic, and you can use Classic runtime in Lightning Experience.

All screen components are supported in Lightning runtime, but not all screen components are supported in Classic runtime. Here are three indicators that a component requires Lightning runtime.

* The component icon is a lightning bolt.
* No preview is available for the component.The Toggle component on the screen canvas, which indicates that no preview is available for this component

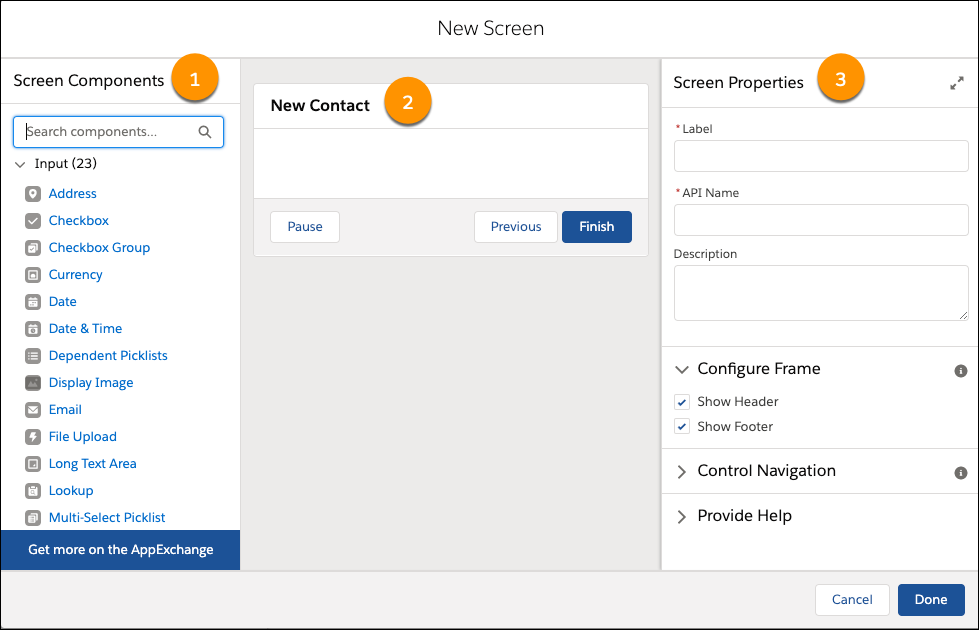


* A warning that a screen component requires Lightning runtime appears when the flow is saved.Warning screen indicating that some screen components require Lightning runtime to work properly.



**The Screen Element**

Let's break down the Screen element.The New Screen element



* Screen Components Pane (1): The left-side pane displays all the screen components available in your org. Click and drag a component to add it to the screen. Tip: Use the search field to easily find the screen component you need.
* Screen Canvas (2): The canvas is where you build your screen. Drag components to arrange them in the right order.
* Properties Pane (3): Depending on the canvas selection, the properties pane shows either the screen's properties or the properties of the selected component. To view or modify the screen properties, click the header or footer in the canvas. The screen properties include whether to display the header, footer, or particular navigation options.

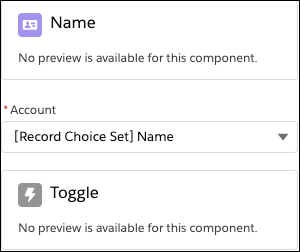
**What You Did in the Project**

Let's look at the screen you built in the Build a Simple Flow project.

In the New Contact flow, open the screen.

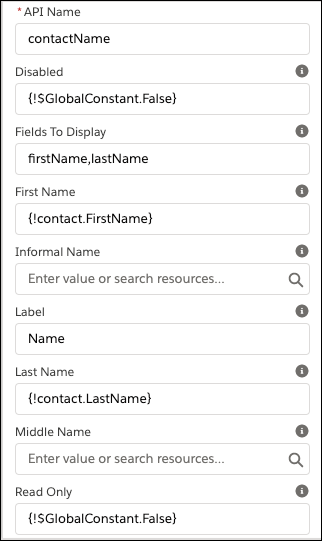
1. From Setup, enter Flows in the Quick Find box, and select Flows.
2. Open the New Contact flow.
3. From the canvas, double-click Contact Info.

The screen contains three components that request information about the contact: the name, the associated account, and a toggle that determines what to do when a contact with that name already exists.



**Name Component**

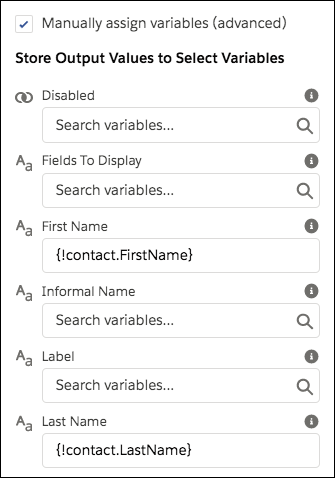
First, let's look at the Name component.



* The component isn't disabled, nor is it read-only.
* The component displays fields for only First Name and Last Name. (The Name component can also display other name fields like Middle Name.) If your project's Name component displayed a field for Salutation, the options would be Mr., Mrs., and Ms.
* The label for the component is Name.

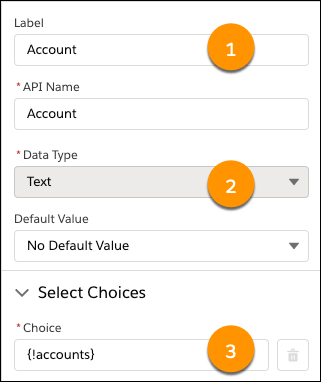
You need to be able to reference the user-entered values for First Name and Last Name in other parts of the flow. Enter the Store Output Values section.

The First Name and Last Name values are stored in fields on the {!contact} record variable.



**Account Component**

Next, let's look at Account, which is a picklist component.

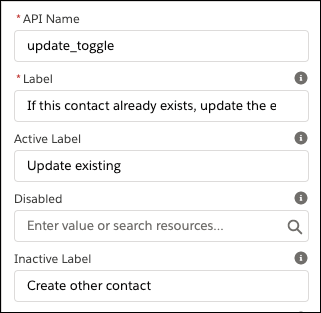


* The component label is Account (1).
* The component supports only Text choices and choice sets (2).
* The choices are generated using the “accounts” record choice set (3).

You don't have to manually store the value of the selected choice in a variable. Instead, you can reference the screen component by its API name (Account). When the user selects an account, the screen component is set to the selected choice's value. Based on how we configured the record choice set, the choice value is the selected account's ID.

**Toggle Component**

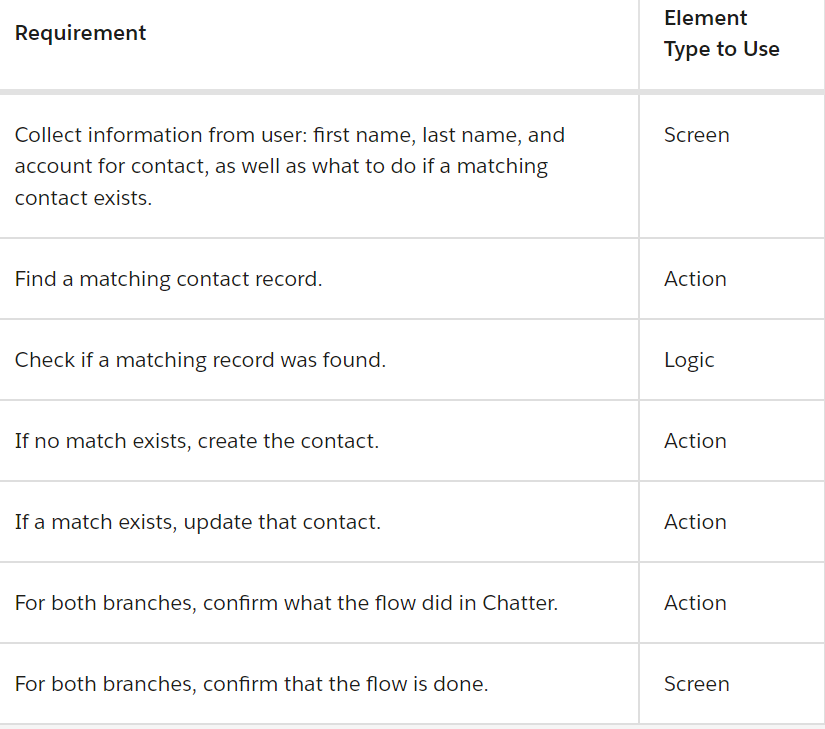
Last up, the Toggle component.



* The text displayed next to the toggle is, “If this contact already exists, update the existing record.”
* When the toggle is activated, its label is Update existing.
* Otherwise, the toggle label is Create other contact.

Update the Flow

Let's take a quick look at the business requirements from the Flow Basics module.



The flow already has a screen to collect information from the user, so we can consider the first requirement as met. But there's one screen missing: one that confirms that the flow is done.

Let's add that confirmation screen.

**Add a Confirmation Screen**

1. If you haven't already, open the "New Contact" flow that you created in the Build a Simple Flow project. Ensure the Freeform option is selected.
2. Drag a Screen element onto the canvas.
3. Give the screen a label and confirm.
4. Scroll or tab to the Configure Footer section, then select the Hide Previous radio button. Leave the other screen properties as is.
5. Add a Display Text component to the screen. From the screen components pane, search for Text and drag Display Text onto the canvas.
6. Give the Display Text component an API name: confirmation\_message. Now let's craft a message that thanks the user and confirms what the flow did. This flow has multiple branches. It either creates a new contact or updates an existing one. Ideally the confirmation messages is either, “Thanks! The contact was created.” or “Thanks! The contact was updated.” To provide custom confirmation messages, you can:

* Create one static confirmation message that works for all possibilities. For example, Thanks! The contact was created or updated. This option is easy, although the user will immediately wonder, Well… which one?
* Create a dynamic confirmation message that changes depending on the outcome of the flow.
* Create a separate confirmation screen for each possibility. (To keep our flow lean, let's leave this as the last resort.)

1. Because only one word changes between the two messages, all it takes to make a dynamic message work is a simple formula.
2. In the text box, enter Thanks! The contact was XYZ. (Don't worry, XYZ is a placeholder for the formula.)

Now, let's figure out what the formula should be.

**Dynamic Confirmation Message with a Formula**

We start with the IF function: IF(logical\_test, value\_if\_true, value\_if\_false)

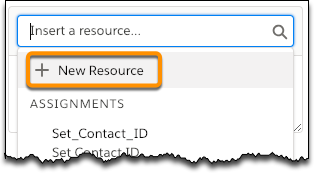
For the logical\_test, check whether the flow created the contact or updated it. To do so, reference the Create Records element. The {!Create\_Contact} merge field resolves to true if the flow executed the Create Contact element. Otherwise (if the flow updated the contact instead), the merge field doesn't resolve to true.

If the logical test is true, the flow created the contact. If the logical test is false, the flow updated the contact. So value\_if\_true is “created” and value\_if\_false is “updated.”

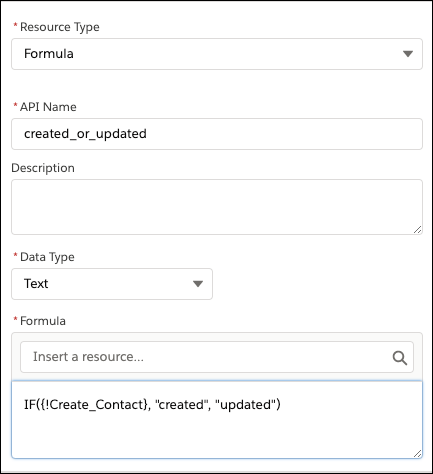
Here's the final formula expression. IF({!Create\_Contact}, "created", "updated")

OK, let's return to the Display Text component and put this expression to work.

1. In the Display Text component, click into the search box above the text box and then click New Resource. The New Resource button highlighted in the dropdown options from the search box.



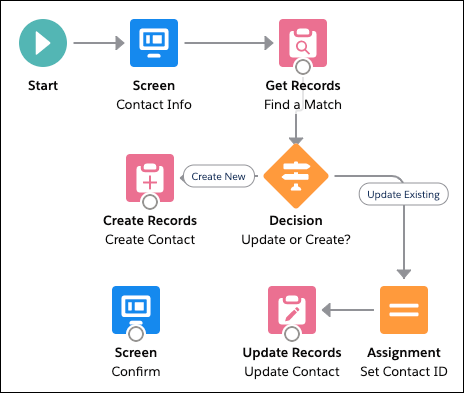
1. Configure the formula with these values, then click Done.



1. In the Display Text component, replace XYZ with a reference to your formula. To insert the formula, click into the search box again, enter created, and select the formula you created.
2. Add a relative link to the created or updated contact, using the ID of the contact record stored in the {!contact} record variable. No formulas required!

In the confirmation message, select the text "The contact", then click the link button Link.

1. In the Link URL window, enter /{!contact.Id}. This may not look like a typical link, but because the link starts with /, your browser knows this means to go to {!contact.Id} on the same website.
2. Click Save to close the Link URL window.
3. On the Screen element, click Done.



1. Save the flow and ignore the warnings. You connect the screen later.

Now the flow has two screens that interact with the user. The first one, which you built in the Build a Simple Flow project, requests information from them. The second one, which you built in this unit, confirms what the flow did with that information. Next up, let's dig in to the basic logic elements in Flow Builder.

**Add Logic to Your Flow**

Learning Objectives

After completing this unit, you'll be able to:

* List the logic elements available in Flow Builder.
* Add branching logic to a flow.
* Change a variable value in a flow.

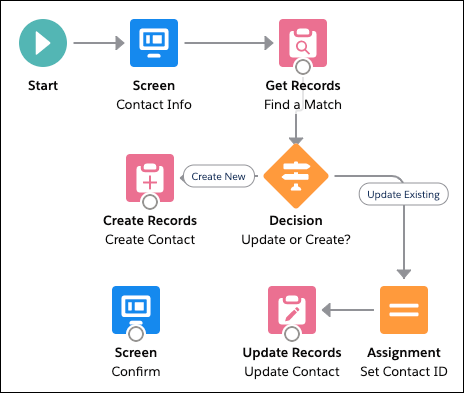
**Create Branches with Decision Elements**

For most business processes, rules determine whether a particular action should be taken or not. Maybe a case should be escalated when the associated account is at risk. The rule in play here is “the associated account is at risk.” In Flow Builder, you evaluate the rules of your business process by using Decision elements.

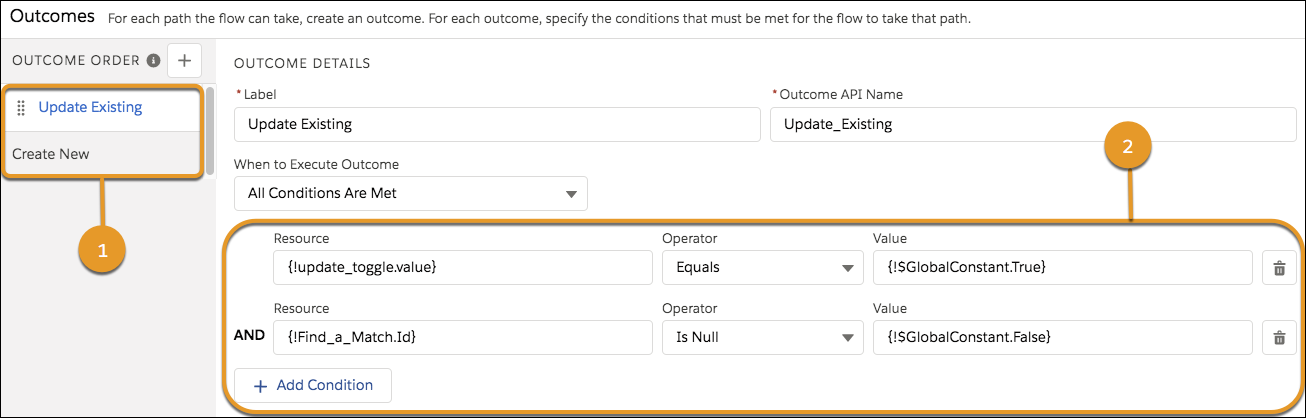
Each decision answers a question, and the outcomes provide the possible answers. The answers can be as simple as Yes and No, but there's no limit to the number of answers. You determine which answer is appropriate by configuring conditions for each outcome. When a flow executes a Decision element, it evaluates each decision outcome in order. If an outcome's conditions are met, the flow takes the associated connector and ignores the other outcomes. If an outcome's conditions are not met, it evaluates the next outcome in the list

Flow Builder provides the last outcome for you, which has no conditions: the default outcome. It acts as a fallback. When the conditions for every other outcome aren't met, the flow takes the connector for the default outcome.

As an example, let's look at the decision you built in the New Contact flow. Notice that the Update or Create? decision has two connectors and, unlike the others in the flow, these connectors are labeled: Create New and Update Existing.



Double-click the Update or Create? decision to open it.



Under Outcome Order (1), you see two outcomes whose labels match what you just saw on the canvas. Each decision outcome has an associated connector on the canvas.

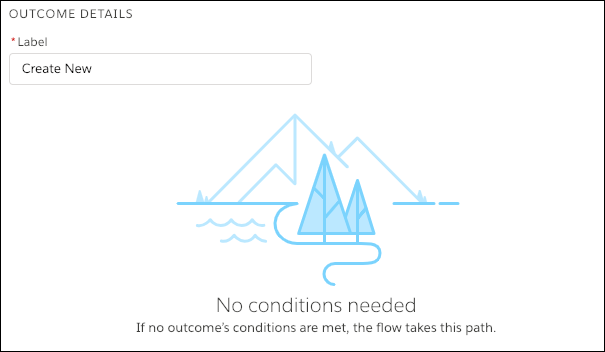
In Outcome Details, you see the conditions (2) for the Update Existing outcome.

{!update\_toggle.value} equals {!GlobalConstant.True} This condition evaluates whether the user opted to update the existing record. The {!update\_toggle.value} represents the value from the Toggle component on the Contact Info screen. If the user activates the toggle, {!update\_toggle.value} is set to true and this condition is met.

Tip: To validate a Boolean value, such as for a toggle or checkbox, always use the global constants for true or false. If you manually enter true, that's the equivalent of the word true rather than the Boolean value that you want.

{!Find\_a\_Match.Id} Is null {!GlobalConstant.False} This condition evaluates whether the flow found a matching record. {!Find\_a\_Match.Id} is set in when the Find a Match element finds a matching record. If {!Find\_a\_Match.Id} is not null, this condition is met. Because When to Execute Outcome is set to All Conditions Are Met, both of these conditions must be true for the flow to take the Update Existing path. Otherwise, the flow evaluates the next outcome in the list.

Under Outcome Order, click the Create New. Notice it has no conditions. That's because it's the default outcome, relabeled to better answer the decision's question (Update or Create?) on the canvas. If the conditions aren't met for the flow to update the existing record, the flow should always create a new one.



Tip: We suggest using the default outcome as the last answer to your decision's question. For example, if the possible outcomes are Yes and No, configure conditions for the Yes outcome, and relabel the default outcome to No. If the answer isn't Yes, it's No.

In summary, when the flow executes the Update or Create? element, it evaluates whether the requirements are met for the flow to update the record (the user opted to update the existing record and there's a matching record to update). If those requirements are met, the flow takes the Update Existing path. Otherwise, it takes the Create New path. Later in the module, you go over the actions the flow executes on these paths.

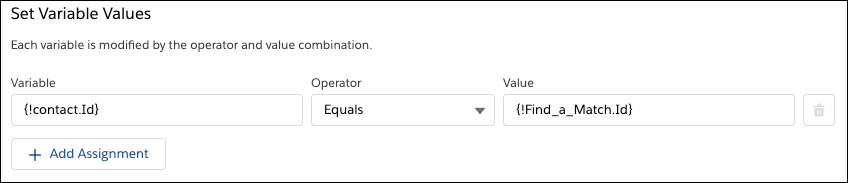
**Update Variable Values with Assignment Elements**

Earlier in this module, you learned about the importance of flow variables and how they can change values during the flow. The easiest way to change a variable's value is with an Assignment element. Here's a quick video with information on loops and the assignment element.

Use each row to modify a variable. How the variable value changes depends on the operator selected and the value set. For example, the Add operator works differently for different data types. For Date variables, Value is added in days to the selected Variable's value. For Text variables, the text entered or selected for Value is added to the end of Variable.

Let's look at the assignment you built in the New Contact flow.

From the canvas, double-click the Set Contact ID node. The element updates one value: the Id field on the {!contact} record variable.



Why do you need this assignment? Because a flow can't update a record if it doesn't know which record to update.

**Update the Flow**

The way the flow is currently built, it looks for a matching contact record before determining whether to update the existing one or create a new one. That's a wasted data element: Why look up the ID of a matching record if the user opts to create a new contact anyway?

To fix that, let's break the existing decision into two decisions. The Update or Create? Decision element checks two conditions: whether the user opted to update the existing record or not, and whether a matching contact record exists.

You move the first condition into its own Decision element, so you only use the Get Records element if the user opted to update a matching record. If the user didn't, you won't bother looking for a matching record.

**Create a New Decision**

1. Drag a Decision element onto the canvas.
2. Give the decision a label: Update If Existing? The API Name is automatically set to Update\_If\_Existing.
3. Give the first outcome the label Yes. Update the API Name to Update\_Yes. This way, the label for the decision connector is easy to read, but you can easily differentiate the name of that outcome from others in the Manager tab.
4. Leave When to Execute Outcome as is, and configure one condition for the outcome.
5. Under Outcome Order, click Default Outcome. Change its label to No. If you leave the outcome label as Default Outcome, it might not be obvious when that outcome is executed.
6. Click Done.

**Update the Original Decision**

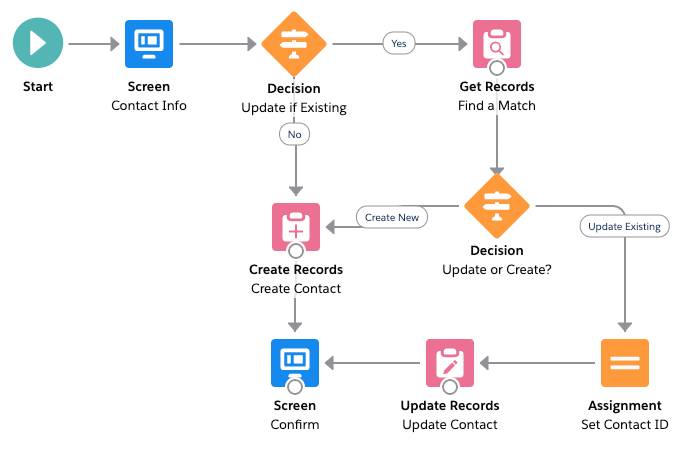
Now that you've added the new decision, the original decision (Update or Create?) has a redundant condition. Let's remove it!

1. Double-click Update or Create?.
2. Find the condition that matches what you added to the Update if Existing? decision. The Resource field should contain {!update\_toggle.value}.
3. For that condition, click the trash icon Delete icon .
4. Click Done.

**Reconnect the Elements**

You're almost done! To finish, you need to make sure the Update If Existing? decision is executed before the Find a Match element.

1. Remove the connector between Contact Info and Find a Match.
2. Click the connector. When it's highlighted, the color changes to blue.
3. Press the Delete key.
4. Connect Contact Info to Update If Existing?.
5. Connect Update If Existing? to Find a Match. When prompted, select the Yes outcome, and click Done.
6. Connect Update If Existing? to Create Contact. Since there's only one outcome left to connect (No), Flow Builder automatically selects it for you.
7. Save the flow, and ignore the FYI warning.



Now the flow has two decision points. First, it evaluates whether to look for a matching record, based on whether the user opted to always create a new contact or not. Second, it evaluates whether the Find a Match element found a matching contact or not.

**Add Actions to Your Flow**

Learning Objectives

After completing this unit, you'll be able to:

* List the actions available in Flow Builder.
* Understand when to use a record variable in a data element.
* Configure a Post to Chatter core action.

**Work with Salesforce Data from a Flow**

The real power of a flow is that it can automatically do things on behalf of the user, including making updates to your data, sending emails, sending information to an external system, and (thanks to integration with Apex and local actions) pretty much anything else you can imagine.

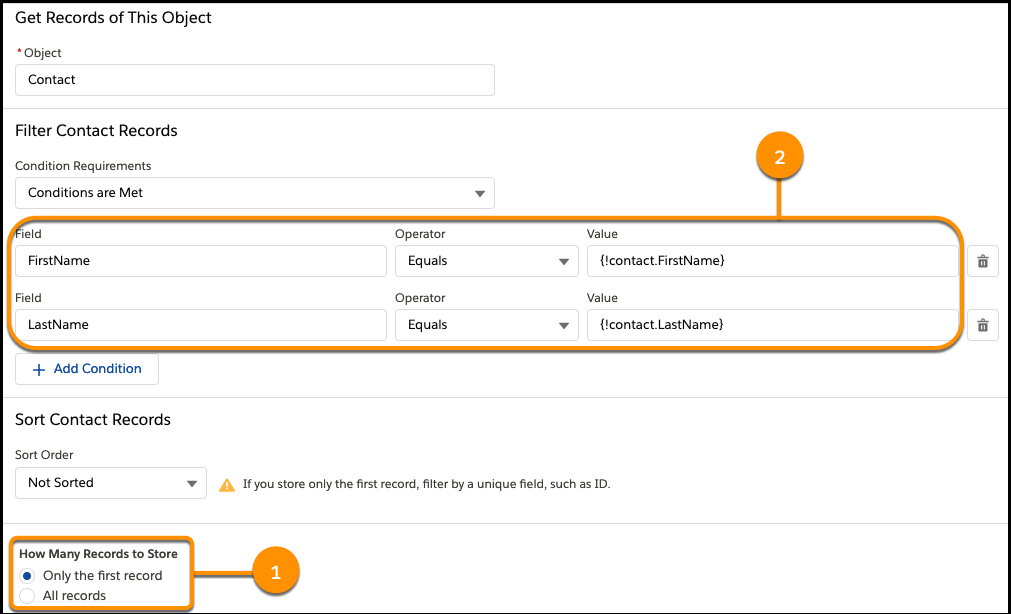
In a flow, you can get values from records, connect to external systems, create records, update records, delete records—the whole shebang!

**Get Records**

In a Get Records element, you identify which object to look up, how to filter the object's records, and how to sort the filtered records. Then you specify how many records to store and whether you want to manually store variables.

If you choose to store all the records that meet the filters, the records' values are stored in a record collection. If you choose to store only the first record, only that record's values are stored.

Let's look at the Find a Match element in your New Contact flow.



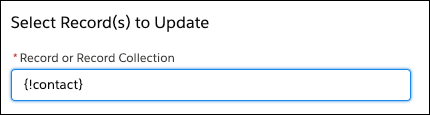
It looks for at least one contact (1) that matches the first and last name the user entered on the screen (2). Back in the Name screen component, you stored the First Name value in {!contact.FirstName} and the Last Name value in {!contact.LastName}.

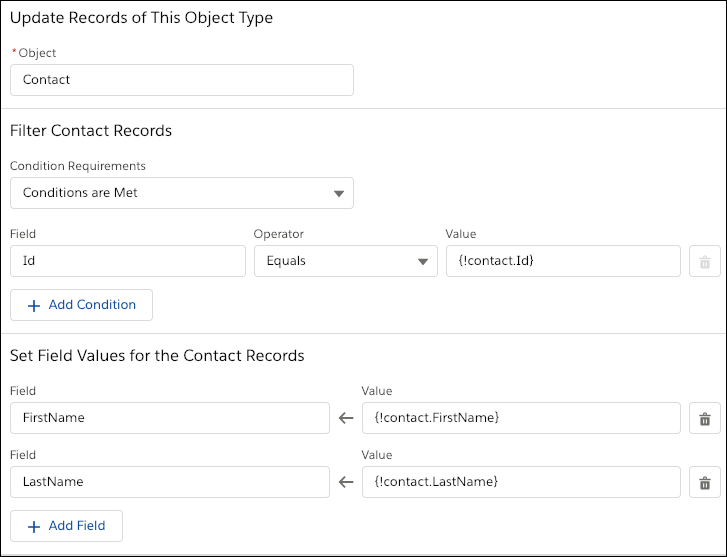
**Update Records Elements**

In an Update Records element, you identify which records to update and how to update those records.

1. If you choose to use a record variable or record collection variable, the flow uses the IDs in that variable to identify the records to update, then updates the records by using the other field values in the variable.
2. Otherwise, you can manually set filter conditions to identify the records to update. With this option, you also manually identify the field values to change.

Here's a comparison of what the Update Contact element in the New Contact flow looks like for each option





The Update Contact element uses a record variable to update the matching contact record, because the flow populates the {!contact} record variable with the right values throughout the flow.

* FirstName and Last Name are populated with in the Contact Info screen
* Id is populated in the Set Contact ID assignment

**Create Records Elements**

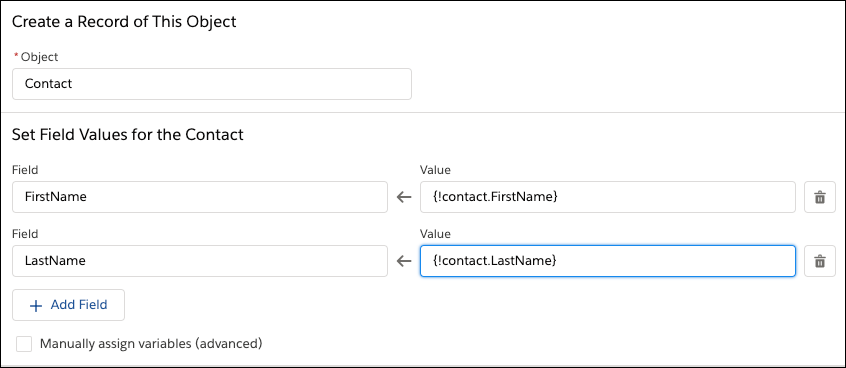
To create multiple records, you must use the values from a record collection variable. Earlier in the flow, populate the record collection variable with the new records' field values. For each record that's created, the flow stores the ID of the created record in the ID fields of the record collection variable.

To create one record, you have two options for setting the record's values.

* If you've populated a record variable with the field values for the new record, choose to set the record fields by using all the values from a record variable. Then select the record variable or record collection variable to use. When the record is created, the flow stores the ID of the created record in the record variable's ID field.
* To manually map values from various resources in the flow, choose to set the record fields by using separate variables, resources, and literal values. The ID of the created record is automatically stored for you. Optionally, you can manually store the ID of the created record in a Text variable, by selecting Manually assign variables (advanced).

Here's a comparison of what the Create Contact element in the New Contact flow looks like for each option.

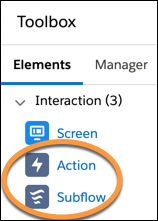




he Create Contact element uses a record variable to create a contact record because the flow populates the {!contact} record variable's FirstName and LastName fields in the “Contact Info” screen.

**Perform Other Actions from a Flow**

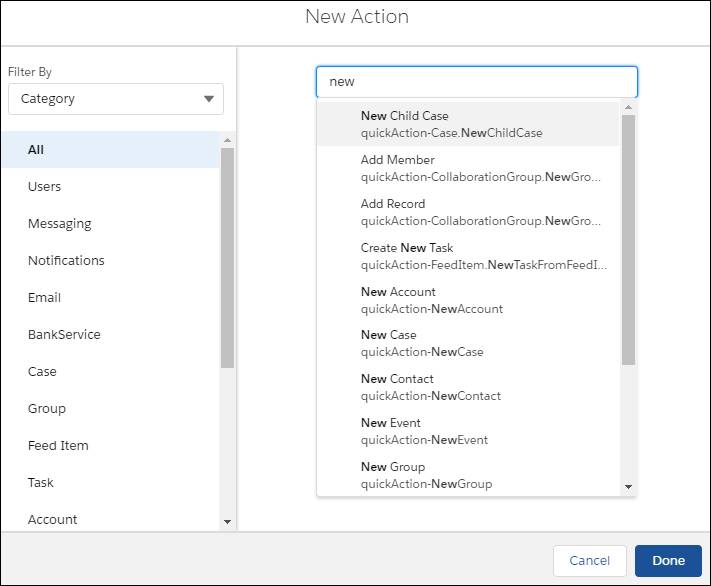
Now you're all set if you need to manipulate Salesforce data from a flow. For anything else, you need the Flow Builder Action or Subflow element.

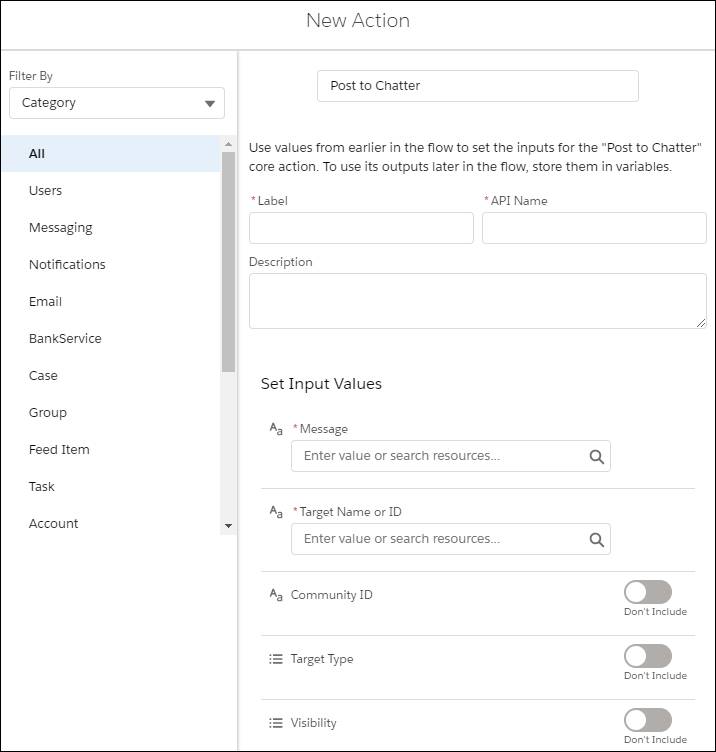


Action elements in the Flow Builder toolbox

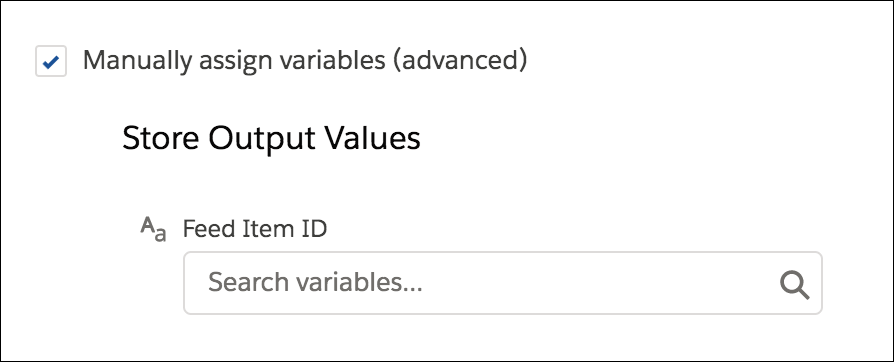
**The Action Element**

Action is one of the Interaction elements in the Toolbox. When you drag an Action element onto the canvas, you can select the specific action to use from the Search All actions... pull-down list and configure it.

Each action requests some values so it knows what to do. These are inputs. For example, Post to Chatter requests five inputs, which you use to identify what to post (see the Message field below) and where to post it.



Some actions respond with information, usually related to the results of the action. These outputs are automatically stored for you and available for use later in the flow. For example, use the feed item ID from a Post to Chatter action to display a link to the created Chatter post.



**Action Types**

Without creating or configuring anything else in Salesforce, you can use these actions.

* Post to Chatter.
* Send Email.
* Submit For Approval.
* Activate or Deactivate a Permission Set.
* Global or object-specific actions provided by Salesforce.

You can also include actions you create with clicks.

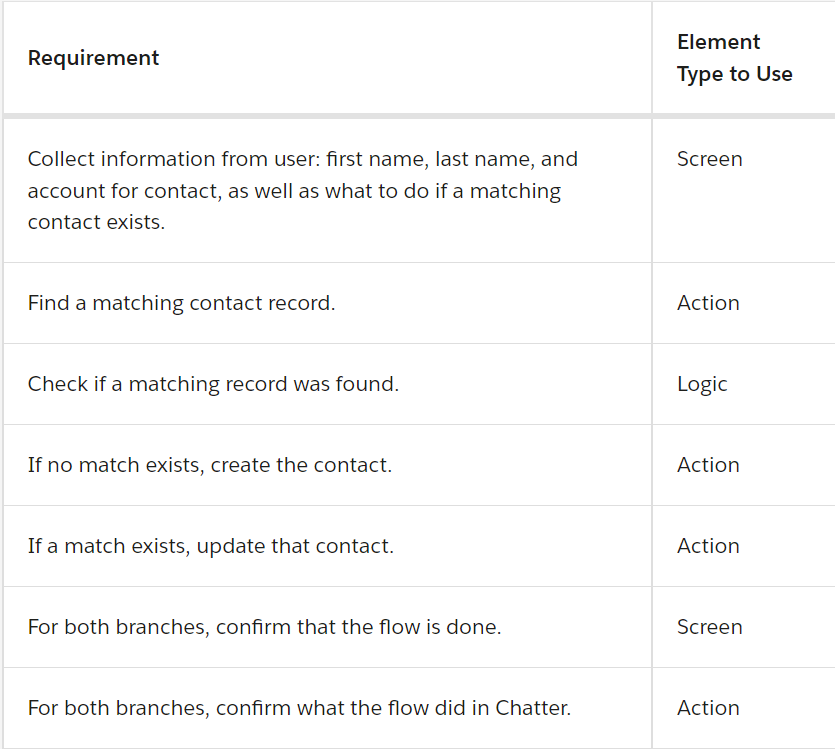
1. Other flows in your org are available in the Subflow element.
2. Custom global or object-specific actions are available in the Action element.

You can also include actions that require code. And you can install custom actions from AppExchange or a third-party library.

1. To do something in the Salesforce database, build or install an Apex action.
2. To do something in the browser, build or install a local action.

**Update the Flow**

According to the business requirements from the Flow Basics module, it's not enough to summarize what the flow did in a confirmation screen. Let's quickly recap the requirements.



So far, the flow meets every requirement except the last action: Confirm what the flow did in Chatter.

**Add a Post to Chatter Action**

1. Drag an Action element onto the canvas.
2. For Action, enter chatter and select Post to Chatter.
3. Give the action a label: Post to Contact's Feed.
4. Set the action's inputs. The input values identify the message to post and which Chatter feed to post it to. For this use case, you only set the required inputs.
5. For Message, click New Resource and select Text Template. Text templates are useful resources for adding merge fields or HTML formatting to some text.
6. Name the text template chatterMessage.
7. For Text Template:
8. Click the down arrow Dropdown arrow and select Plain Text.
9. Enter The contact was .
10. In the resource picker, enter created and select the formula you created earlier: created\_or\_updated.
11. Add a period to finish that sentence.
12. Here's the final text template: The contact was {!created\_or\_updated}.
13. Click Done.
14. Make sure message is set to the text template you created.
15. For Target Name or ID, enter {!contact.Id}, and select the Id field.
16. Click Done.

You've now successfully configured an action to post a message, using a text template, to the Chatter feed of the contact record that the flow created or updated.

**Connect the Elements**

The last step is connecting the action to the rest of the flow so the Chatter post is created when the flow runs. Let's place the action before the confirmation screen, so the confirmation is the last step in the flow.

1. Connect the Create Contact and Update Contact elements to the Post to Chatter core action.
2. Connect the Post to Chatter core action to the Confirm screen.
3. Save the flow, and ignore the warning. The “New Contact” flow, with the new Post to Chatter core action and its connectors highlighted.

