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.

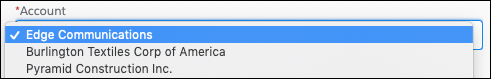
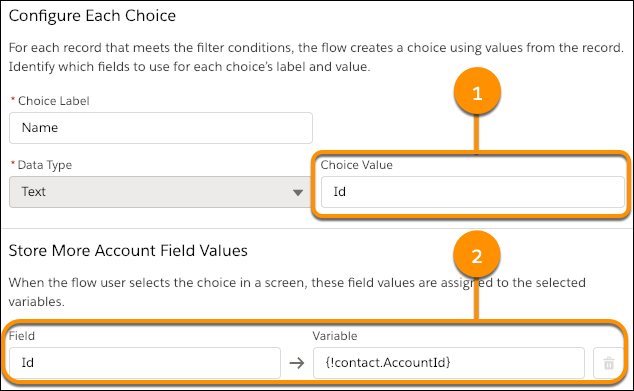
**Before You Start**

Before you can complete this module, make sure that you complete the [Build a Simple Flow](https://trailhead.salesforce.com/content/learn/projects/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.  
    
  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.

Note

**Note**

This isn't a comprehensive list. For details about all available resources, see the Resources section of this unit.

**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.

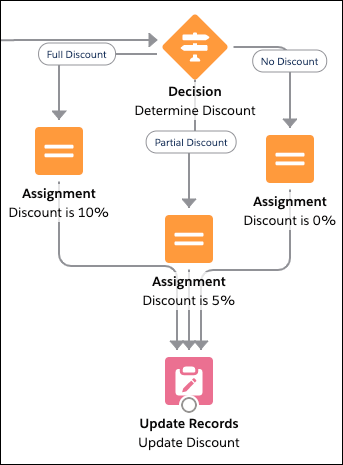
Simply put, a variable is a placeholder for a value you don't know yet. Every flow resource is a placeholder, but variables are the only resource that can change during the flow, hence the name “variable.” In fact, Flow Builder includes the Assignment element just for updating the values of variables. Watch a quick video for more information on what a [variable](https://salesforce.vidyard.com/watch/MppqcKZd36p1YYX8iUruXi) 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](https://www.google.com/url?q=http://trailhead.salesforce.com/projects/flow_calculate&sa=D&ust=1550185536727000), 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.

**Resources**

* [*Salesforce Help*: Flow Resources](https://help.salesforce.com/articleView?id=flow_ref_resources.htm&language=en_US)
* [*Video*: Learn Salesforce Flow](https://salesforce.vidyard.com/watch/PjNcmecuqcYKH2cJsfLEAy)

Top of Form

**Quiz**

**+100 points**

**1A choice set can generate options with:**

**A.**Formulas

**B.**Filtered records

**C.**Picklist field values

**D.**B and C

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.

| **Data Type** | **Use to store...** |
| --- | --- |
| Text | IDs, descriptions, and other text or long text data. |
| Record | Field values from a record, such as an opportunity. When this data type is selected, you create a record variable. |
| Number | A numeric value. |
| Currency | A currency value. |
| Boolean | A yes/no value, such as whether a checkbox was selected. |
| Date | A date value. |
| Date/Time | A date and time value. |
| Picklist | A picklist value. |
| Multi-Select Picklist | Picklist values, separated by semicolons. |
| Apex-Defined | Field values from an Apex class. When this data type is selected, you create an Apex-defined variable. |

**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*.

We don't go into much detail about collection variables in this module, but you can see them in action in the [Salesforce Flow](https://trailhead.salesforce.com/content/learn/modules/business_process_automation/) module.

**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 in doubt, go ahead and make your variable 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.

| **How to Populate a Variable** | **Description** |
| --- | --- |
| Set the variable's default value | If you know what the initial value of the variable should be, set its default value when you create it. The variable keeps that value until it gets overwritten by another element. |
| Get a value from a record and put that value in the variable | When you configure a Get Records element (which we cover later in this module) or a record choice set, specify the fields to store and which variables to store them in. |
| Change the variable's value from an element | The most obvious way to change a variable's value is with an Assignment element. But other elements let you store values in variables. For example, the Create Records element lets you store the created record's ID in a variable. |
| Pass a value in from outside the flow | Depending on how you distribute your flow, you can pass a value into an input variable. For example, if you add the flow to a Contact record page, you can pass the contact's ID into a flow variable. |

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.

Note

#### Note

This badge uses Flow Builder with the Freeform option. Be sure to select **Freeform** instead of Auto-Layout to follow along.

## 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.

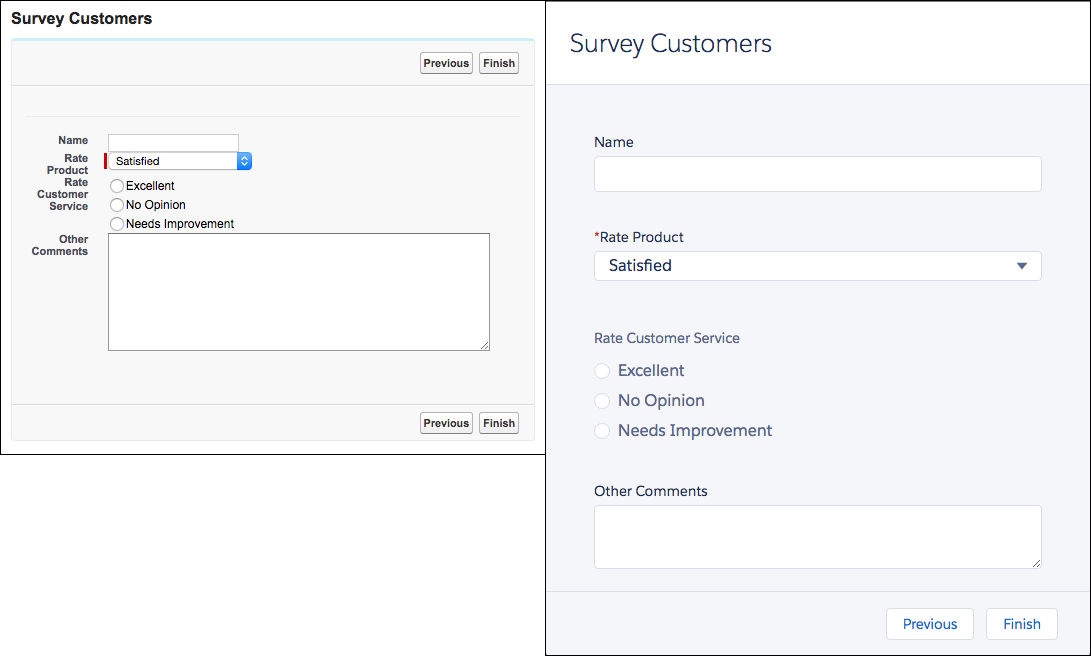
* A record choice set generates multiple options by using a filtered list of records.
* A picklist choice set generates multiple options by using the values of a picklist or multi-select picklist field.
* A choice represents a single option with the label and value set manually.

**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.



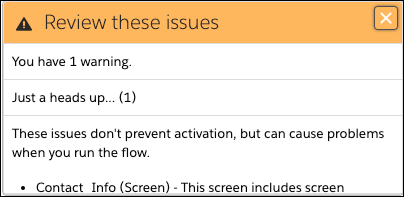
That said, the two runtime experiences aren't tied to either desktop experience. You can use Lightning runtime in Salesforce Classic, and you can use Classic runtime in Lightning Experience.

Note

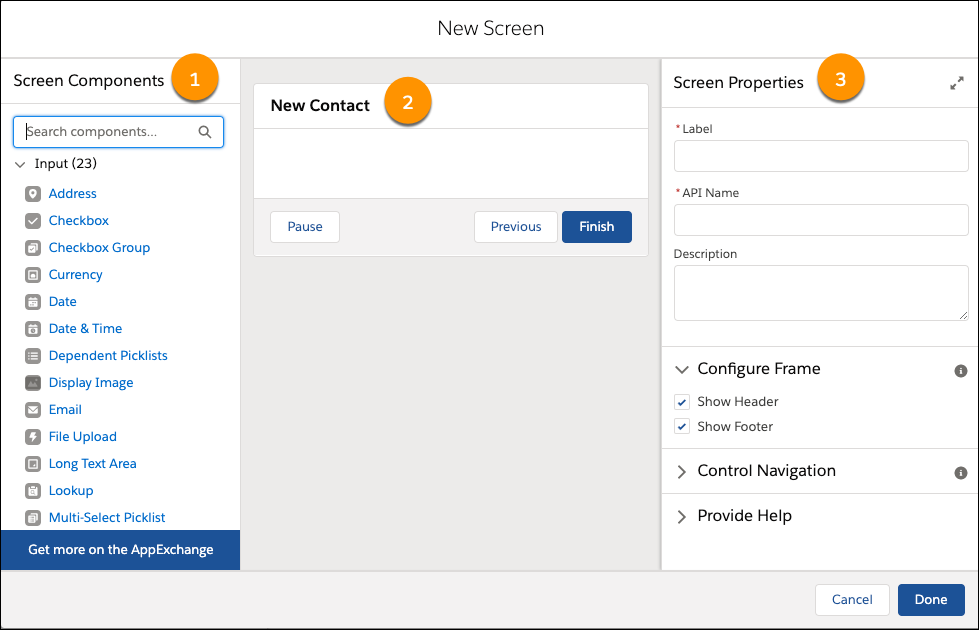
#### Note

We're no longer enhancing Classic runtime, so if you want to take advantage of enhancements to how flow screens work, use Lightning runtime.

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.
* A warning that a screen component requires Lightning runtime appears when the flow is saved.

## The Screen Element

Let's break down the 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.

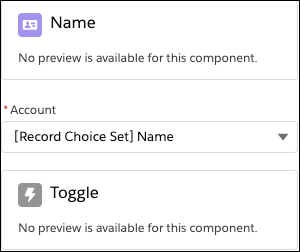
## What You Did in the Project

Let's look at the screen you built in the [Build a Simple Flow](https://trailhead.salesforce.com/content/learn/projects/build-a-simple-flow) project.

In the New Contact flow, open the screen.

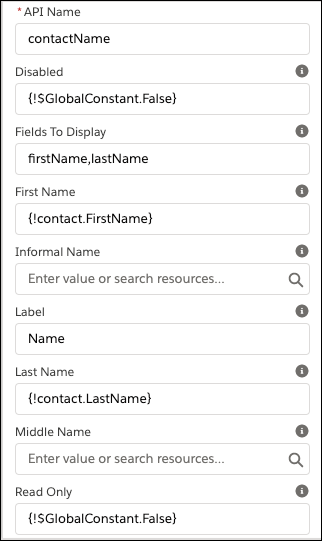
1. From Setup, enter Flows in the Quick Find box, and select **Flows**.
2. Open the **New Contact** flow.
3. From the canvas, double-click **Contact Info**.

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

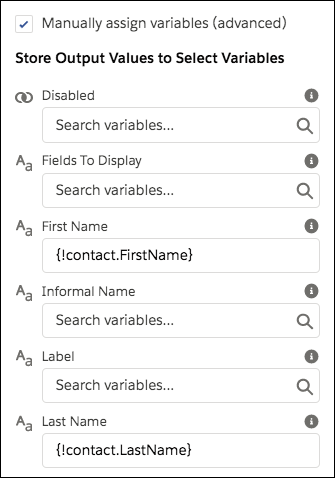
First, let's look at the Name component.



* 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.

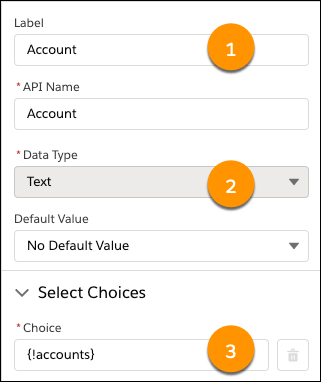
You need to be able to reference the user-entered values for First Name and Last Name in other parts of the flow. With the Name component, the only way to reference the entered values is by storing the values in variables. Enter the Store Output Values section.

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.

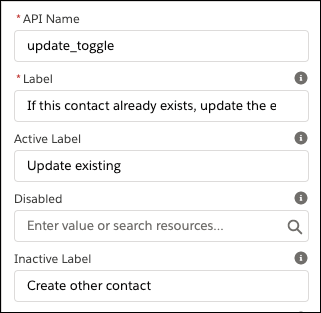


* The component label is Account **(1)**.
* The component supports only Text choices and choice sets **(2)**.
* The choices are generated using the “accounts” record choice set **(3)**.

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](https://trailhead.salesforce.com/content/learn/modules/flow-basics) module.

| **Requirement** | **Element Type to Use** |
| --- | --- |
| Collect information from user: first name, last name, and account for contact, as well as what to do if a matching contact exists. | Screen |
| Find a matching contact record. | Action |
| Check if a matching record was found. | Logic |
| If no match exists, create the contact. | Action |
| If a match exists, update that contact. | Action |
| For both branches, confirm what the flow did in Chatter. | Action |
| For both branches, confirm that the flow is done. | Screen |

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](https://trailhead.salesforce.com/content/learn/projects/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.)
7. Because only one word changes between the two messages, all it takes to make a dynamic message work is a simple formula.
8. 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

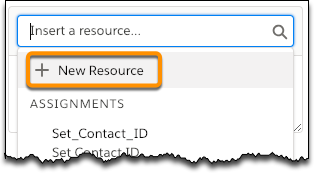
We start with the IF function: IF(logical\_test, value\_if\_true, value\_if\_false)

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.

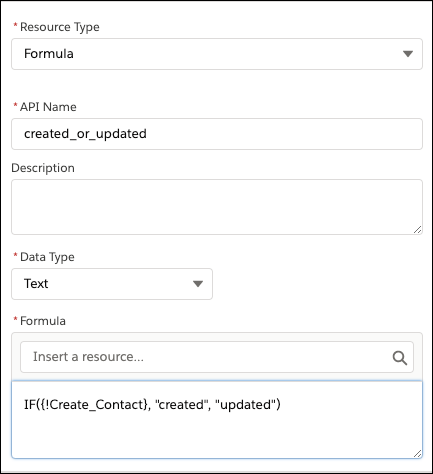
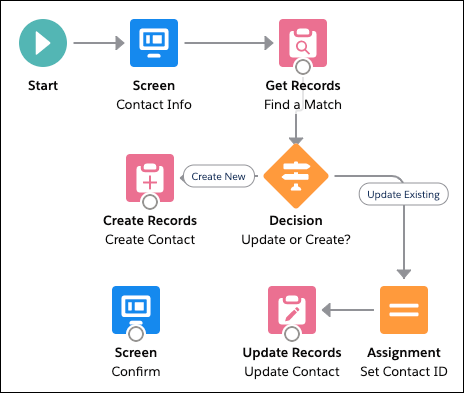
If the logical test is true, the flow created the contact. If the logical test is false, the flow updated the contact. So **value\_if\_true** is “created” and **value\_if\_false** is “updated.”

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**. 
2. Configure the formula with these values, then click **Done**.

| **Field** | **Value** |
| --- | --- |
| Resource Type | Formula |
| API Name | created\_or\_updated |
| Data Type | Text |
| Formula | IF({!Create\_Contact}, "created", "updated") |

1. 
2. 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.
3. Add a link to the created or updated contact. No formulas required! You create a relative link using the ID of the contact record stored in the {!contact} record variable.
   1. In the confirmation message, select the contact, then click the link button Link.
   2. Delete the existing text and enter /{!contact.Id} .
   3. Click **Save**.
   4. On the Screen element, click **Done.**  
      ****
   5. 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](https://trailhead.salesforce.com/content/learn/projects/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.

## Resources

* [Salesforce Help: Screen Element Reference](https://help.salesforce.com/articleView?id=flow_ref_elements_screen.htm&language=en_US)
* [Salesforce Help: Provided Screen Components](https://help.salesforce.com/articleView?id=flow_ref_elements_screencmp.htm&language=en_US)
* [AppExchange: Flow Solutions](https://appexchange.salesforce.com/appxStore?type=Flow)
* [Lightning Aura Components Developer Guide: Customize Flow Screens with Aura Components](https://developer.salesforce.com/docs/atlas.en-us.224.0.lightning.meta/lightning/components_config_for_flow_screens_intro.htm)
* [Trailhead: Formulas & Validations](https://trailhead.salesforce.com/en/content/learn/modules/point_click_business_logic)

Top of Form

## Hands-on Challenge

**+500 points**

### GET READY

You’ll be completing this unit in your own hands-on org. Click **Launch** to get started, or click the name of your org to choose a different one.

If you use Trailhead in a language other than English, make sure that your hands-on org is set to the same language as the challenge instructions. Otherwise you may run into issues passing this unit. Want to find out more about using hands-on orgs on Trailhead? Check out [Trailhead Playground Management](https://trailhead.salesforce.com/en/content/learn/modules/trailhead_playground_management).

### YOUR CHALLENGE

**Build a Flow to Help Users Create an Opportunity**

Build a flow that includes a screen to get required information from the user. You’ll add logic and actions to the flow in later hands-on challenges.

* Create a flow:  
  + Type: **Screen Flow**
* Create a resource:  
  + Resource Type: **Variable**
  + API Name: **accountName**
  + Data Type: **Text**

Note: You can create resources from the Manager tab in the toolbox.

* Create a resource:  
  + Resource Type: **Record Choice Set**
  + API Name: **accounts**
  + Object: **Account**
  + Choice Label: **Name** (Account Name)
  + Data Type: **Text**
  + Choice Value: **Id** (Account ID)
  + Field: **Name** (Account Name)
  + Variable: **{!accountName}**
* Create a screen:  
  + Label: Oppty Screen
  + API Name: Oppty\_Screen
* Add a text component to the screen:  
  + Screen Component: **Text**
  + Label: **Oppty Identifier**
  + API Name: **Oppty\_Identifier**
* Add a date component to the screen:  
  + Screen Component: **Date**
  + Label: **Close Date**
  + API Name: **Close\_Date**
* Add a picklist component to the screen:  
  + Screen Component: **Picklist**
  + Label: **Account**
  + API Name: **Account**
  + Data Type: **Text**
  + Choice: **{!accounts}**
* Connect the Start node to the screen
* Save the flow:
  + Label: **New Oppty**
  + API Name: **New\_Oppty**

Bottom of Form