Before You Begin

Did you sign up for a special OmniStudio Developer Edition org already? You'll need one to do the steps in this guide. Here's how to request one if this is your first time completing an OmniStudio module:

1. Sign up for a free Developer Edition org with OmniStudio.
2. Fill out the form.
    a. For Email, enter an active email address.
    b. For Username, enter a username that looks like an email address and is unique, but it doesn't need to be a valid email account (for example, yourname@omnistudiotrails.com).
    c. After you fill out the form, click **Sign me up**. A confirmation message appears.
3. When you receive the activation email (this might take about 10 minutes), open it and click **Verify Account**.
4. Complete your registration by setting your password and security challenge question.
   Tip: Write down your username, password, and login URL for easy access later.

You are logged in to your Developer Edition and you can begin practicing.

Create a Simple OmniScript

Requirements

"As a service agent, I'd like a simple guided workflow that allows me to update some basic account details from the service console."

Based on the user requirements, create an OmniScript that allows a user to edit account information.

Prerequisites

- None

Tasks

1. Create the Edit Account OmniScript
2. Add an Integration Procedure Action to the Edit Account OmniScript to fetch data
3. Add an Integration Procedure Action to the Edit Account OmniScript to save data
4. Add a Navigate Action to the Edit Account OmniScript

Time

- 40 mins

Task 1: Create the Edit Account OmniScript

1. Go to the OmniScripts tab and find the team/editAccount OmniScript.

    a. Open the **App Launcher**.

    b. Select the **OmniStudio** app.

    c. Click the dropdown arrow to open the menu and select the **OmniScripts** tab.

    Your org has starter OmniScripts for you to use. The type for these OmniScripts is **team**.

    d. In the Search/Find in page box, type the keyword `team`.

    e. From the search results, expand team/editAccount.

    f. Click **Team Starter Edit Account (Version 1)**.

    g. The OmniScript opens in a new tab. Review the information in the Header.

    Omnistudio uses the combination of type, subtype, and language to identify the OmniScript when launching it via an action. In this example:

    - Type = team
    - SubType = editAccount
    - Language = English

    Version 1 of the OmniScript is active – leave it that way and create a new version to edit.

2. Create a new version of the OmniScript for editing.

    a. In the header, click **New Version**. The Version increments from 1 to 2 and the new version opens in a new tab.

    b. Close the tab with version 1 to prevent confusion later.

    c. Confirm you are in the tab for version 2, and in the header, click **Edit**.

   d. In the **Name** field, remove **Starter** from the name. (It then reads **Team Edit Account.**)

   e. Click **Save**.

3. Explore the help tips in the Script Configuration.

   a. Click the **Setup** panel.

   b. Next to the **Reusable** field, move your cursor over the tooltip Info icon to learn about embedding an OmniScript inside another one.

      You can have only one level of nesting. This means if you plan to reuse an OmniScript, it cannot have a reusable OmniScript as part of the structure.

4. As you plan to build a one-step OmniScript, hide the step chart that displays to let users know where they are in a sequence of steps.

   a. Scroll down the Setup panel and expand **STEP CHART OPTIONS**.

   b. Select the **Hide Step Chart** checkbox.

5. Build the basic structure of the OmniScript.

   a. Select the **Build** panel.

   b. Expand the **GROUPS** section.

   c. Drag a **Step** element into the **Canvas**. A step forms a page in an OmniScript.

   d. Select the new element.

   e. Click the **Properties** panel. (It is visible between Build and Setup when you select the element.)

   f. Specify the following values for the element:

Property	Value	Best Practice
Element Name	`StepAccount`	Element names must be unique with no spaces or special characters. It's best

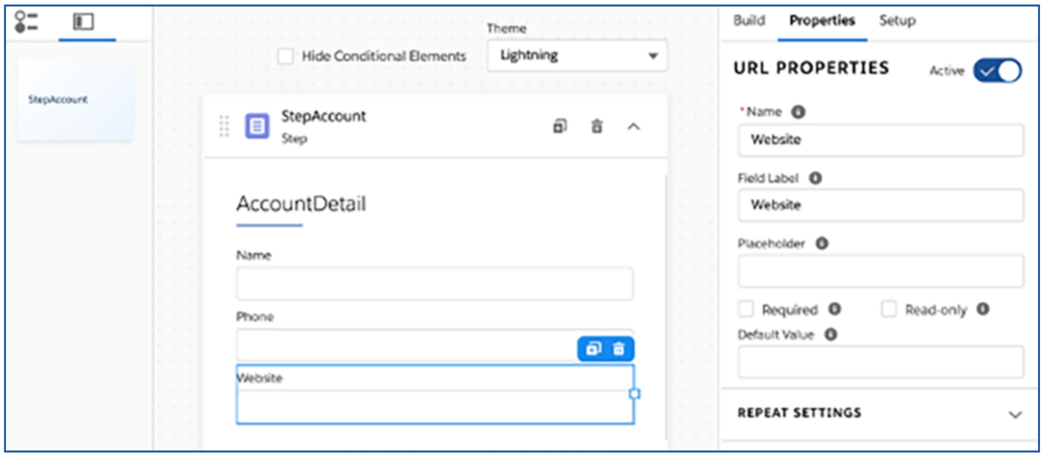
		to use a descriptive name in camel case for element names. They are not visible to users.
Field Label	`Account Details`	Field Labels are what the users see at the top of the page for each step.
Chart Label		If you are displaying the Step Chart this provides a unique label for each step. As you are hiding the Step Chart, there is no need to fill in this field.
Instruction		If you wish to provide scripting or guidance for end-users enter it here. You explore this later in Part 2.

6. Add account details.

    a. In the **Build** panel, expand the **INPUTS** section.

    b. Drag the following elements into the **StepAccount** element, and edit the values as follows:

Element	Element Name	Field Label
Text, for the account name	`Name`	`Name`
Telephone, for the account phone number	`Phone`	`Phone`
URL, for the account website	`Website`	`Website`



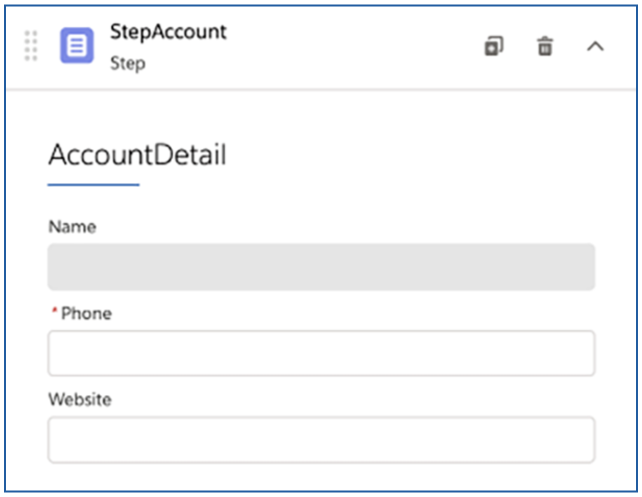
c. Click the **Name** element and select **Read-only**.

   Because it holds data coming from the Salesforce account name (Account.Name) field, the OmniScript user must be unable to edit the name.

d. Click the **Phone** element and select **Required**.

e. In the **Phone** element, remove the **Mask**. The phone number does not display in a Lightning web component with a mask in place.

f. Notice the **Name** field now displays read-only and the **Phone** field now displays required.

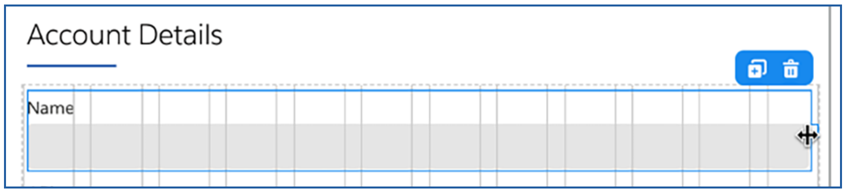


The fields are wider than necessary.

7. Use the **Control Width** element property to narrow the fields. Control Width settings are an HTML web standard and have a responsive grid.

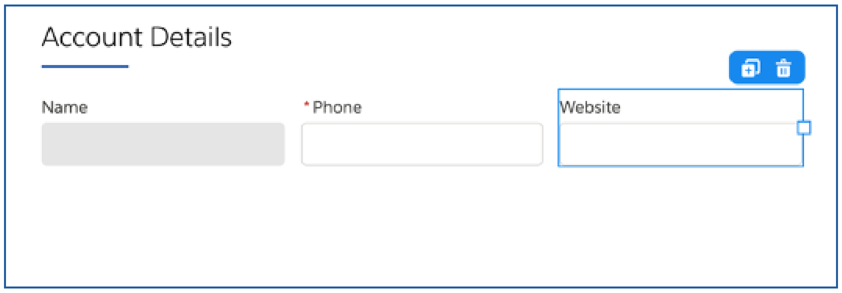
    a. In the **Name** element, click on the right-side border of the element to activate the **Control Width Grid**.

    b. Drag the slider to set the **Control Width** to 4 squares of the grid.



    c. Set the control width to 4 for the **Phone** and **Website** elements.

    The fields are now displayed on a single line.



8. Edit the OmniScript to display the fields on separate lines.

    a. Click the **Build** panel.

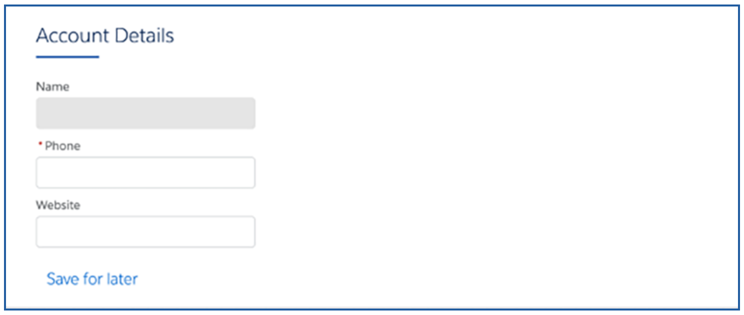
    b. In the **Search For Element** field, enter line to filter the list for **Line Break**. (Alternatively, expand the **Display** section to find it.)

    c. Drag a **Line Break** element into the canvas, between the **Name** and **Phone** elements.

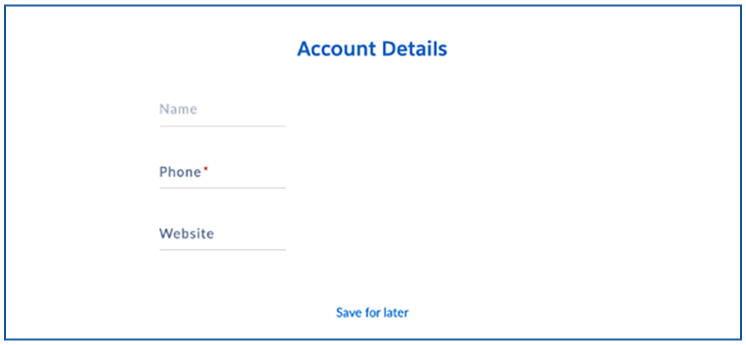
    d. Clone the Line Break and drag it to be between the **Phone** and **Website** elements.

    The fields are now on separate lines.

e. In the Header, click **Preview** to view how the fields appear to end users.



f. Use the **Theme** drop-down to see the difference between **Lightning** and **Newport**.



Review

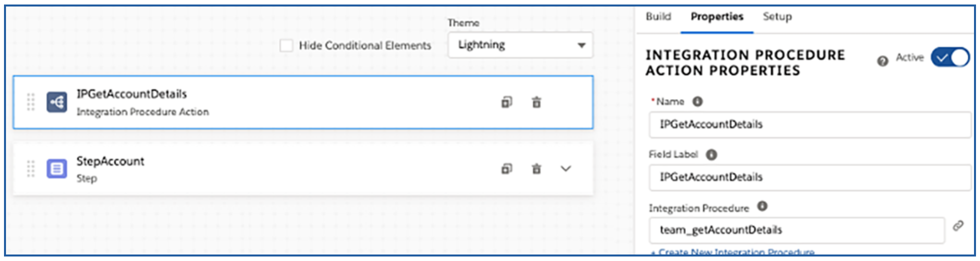
Confirm your understanding by answering these questions.

1. What makes an OmniScript unique?
2. What is the purpose of a Step Element?
3. What are the best practice naming conventions for element names? Why?
4. What is the purpose of the Control Width setting?

Task 2: Add an Integration Procedure Action to Get Data for the Edit Account OmniScript

To get account data from Salesforce, link an Integration Procedure to the Edit Account OmniScript. A prebuilt Integration Procedure is available for you to use.

1. Configure the first part of the data flow in your team/EditAccount OmniScript using a ContextId to get some data.
    a. If still in Preview for Team Edit Account, in the Header, click **Design**.
    b. Click the **Build** panel. Either expand the **ACTIONS** section or use the search to locate **Integration Procedure Action**.
    c. Drag the **Integration Procedure Action** element into the canvas above StepAccount.
    d. Change the **Element Name** and **Field Label** to `IPGetAccountDetails`. The Field Label is not displayed in the Preview. However best practice when working with Lightning web components is to enter the name in the Field Label. This is because the field label **does** display in the Action Debugger. If you configure multiple Integration Procedures for the OmniScript, having descriptive labels for them helps you confirm if they are pulling and pushing the data correctly.
    e. From the **Integration Procedure** dropdown list, choose **team_getAccountDetails**.


Theme
Hide Conditional Elements
Lightning
IPGetAccountDetails
Integration Procedure Action
StepAccount
Step
Build
Properties
Setup
INTEGRATION PROCEDURE ACTION PROPERTIES
Active
+ Create New Integration Procedure

*Name: IPGetAccountDetails
Field Label: IPGetAccountDetails
Integration Procedure: team_getAccountDetails

2. Configure REMOTE PROPERTIES to identify the variable to send to the Integration Procedure.

You need to send a RecordId to identify which Salesforce account record to update. This data is stored in the ContextId.

a. In the **Properties** panel, expand **REMOTE PROPERTIES**. Under **Extra Payload**, set the following values:

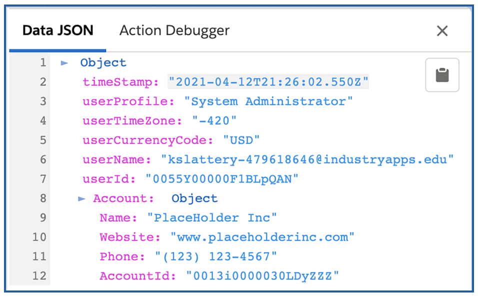
Property	Value	Notes
Key	`AccountId`	The key is the name of the variable to send to the integration Procedure.
Value	`%ContextId%`	Use a merge field to get the value in the ContextId variable. The merge field syntax is a percent sign before and after the value. Enter ContextId exactly as shown.

3. Test the data configuration.

a. In the Header, click **Preview**.

b. Notice that the Name, Phone, and Website fields do not display any data.

c. Look at the **Data JSON** panel and notice the JSON from the Integration Procedure displays, but not in the step. The data is not displaying in the UI because the Integration Procedure uses a DataRaptor Turbo Extract, and DataRaptor Turbo Extracts return objects in a nested array format.



```
► Object
  timeStamp: "2021-04-12T21:26:02.550Z"
  userProfile: "System Administrator"
  userTimeZone: "-420"
  userCurrencyCode: "USD"
  userName: "kslattery-479618646@industryapps.edu"
  userId: "0055Y00000F1BLpQAN"
 ► Account:  Object
    Name: "PlaceHolder Inc"
    Website: "www.placeholderinc.com"
    Phone: "(123) 123-4567"
    AccountId: "0013i0000030LDyZZZ"
```

OmniScripts have a parser that matches incoming data with the elements. To populate the fields in the UI, you need to ensure the element names match, and the Integration Procedure element is configured to recognize the nested array. In addition, to condense the size of the script and make them run faster, LWC OmniScripts only display data in the JSON when there is data to display (information is not null) and only displays data one step at a time. Updating the Integration Procedure element properties to recognize the data is in a nested array, updates the JSON and the data displays in the Step.

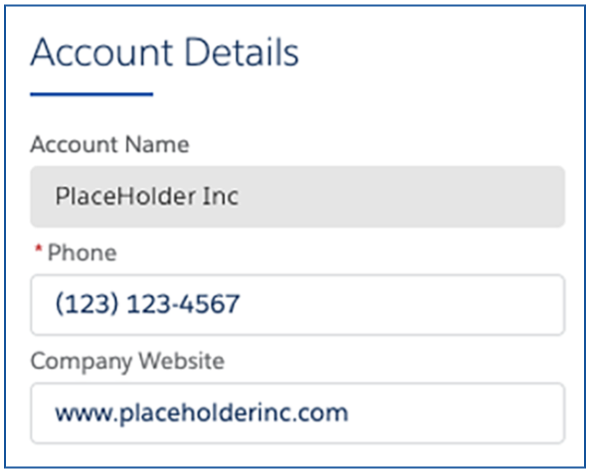
d. In the Header, click **Design**.

e. In the Canvas, select **IPGetAccountDetails**. Then in the **Properties** panel, select **SEND/RESPONSE TRANSFORMATIONS** and set the following values:

Property	Value	Explanation
Response JSON Path	`Account|1`	Account identifies the Object node. Appending a | delimiter and a number accesses the correct instance in the nested array of that node. This enables OmniScripts to prefill elements.
Response JSON Node	`VlocityNoRootNode`	This ensures the object is not nested.

f. In the Header, click **Preview**.

g. Notice the data from the Integration Procedure now populates the fields in the UI.



Account Details

Account Name

PlaceHolder Inc

* Phone

(123) 123-4567

Company Website

www.placeholderinc.com

Review

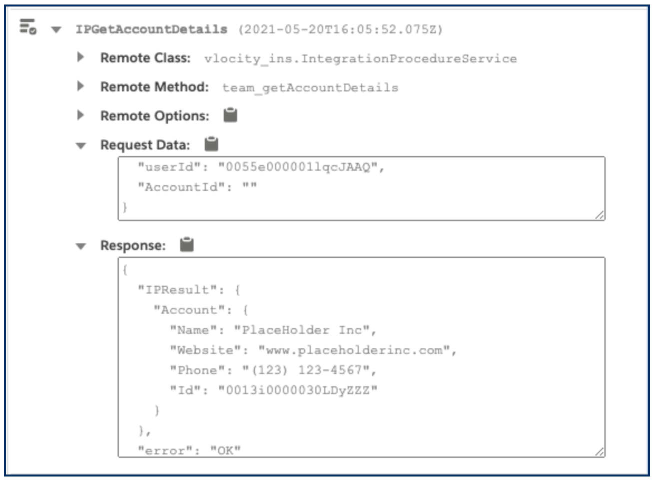
Confirm your understanding by answering these questions.

1. What are the advantages of using OmniStudio Integration Procedures as data sources?
2. Why did you use the same value in the Name and Field Label of the Integration Procedure?
3. What determines the JSON structure of an OmniScript?
4. How does an OmniScript match incoming data with elements?
5. Why did we update the mask?

Task 3: Add an Integration Procedure Action to Save Data Updated in the Edit Account OmniScript

Add a second Integration Procedure Action to link another Integration Procedure to the Edit Account OmniScript. A prebuilt Integration Procedure is available for you to use. This one saves the updated account details back to Salesforce.

1. Add a new Integration Procedure Action.
   a. If still in Preview, in the Header, click **Design**.
   b. Click the **Build** panel. Either expand the **ACTIONS** section or use the search to locate **Integration Procedure Action**.
   c. Drag an **Integration Procedure Action** element into the Canvas and place it under the StepAccount element.
   d. Change the Element Name and Field Label to `IPSaveAccountDetails`.
   e. From the **Integration Procedure** dropdown list, choose **team_saveAccountDetails**.
2. Preview the changes and view the data flowing to and from the OmniScript.
   a. In the Header, click **Preview**.
   b. Click **Action Debugger**.
   c. If there are multiple entries, click **Clear Logs** to clear any existing data in the debug console.
   d. If you cleared the logs, in the Canvas, click the **Reset Data** button.
   e. In the **Action Debugger**, expand the following nodes:
      i. **IPGetAccountDetails**, then
      ii. **Request Data**
      iii. **Response** (resize the response box by grabbing the lower-right corner if you wish to see all of it without scrolling)



IPGetAccountDetails (2021-05-20T16:05:52.075Z)

Remote Class: vlocity_ins.IntegrationProcedureService

Remote Method: team_getAccountDetails

Remote Options:

Request Data:

```
  "userId": "0055e000001lqcJAAQ",
  "AccountId": ""
}
```

Response:

```
{
  "IPResult": {
    "Account": {
      "Name": "PlaceHolder Inc",
      "Website": "www.placeholderinc.com",
      "Phone": "(123) 123-4567",
      "Id": "0013i0000030LDyZZZ"
    }
  },
  "error": "OK"
```

f. Notice the request and response expected JSON output with the account name, website, and phone data. (The AccountId in the Request Data is empty because you are using stub data.)

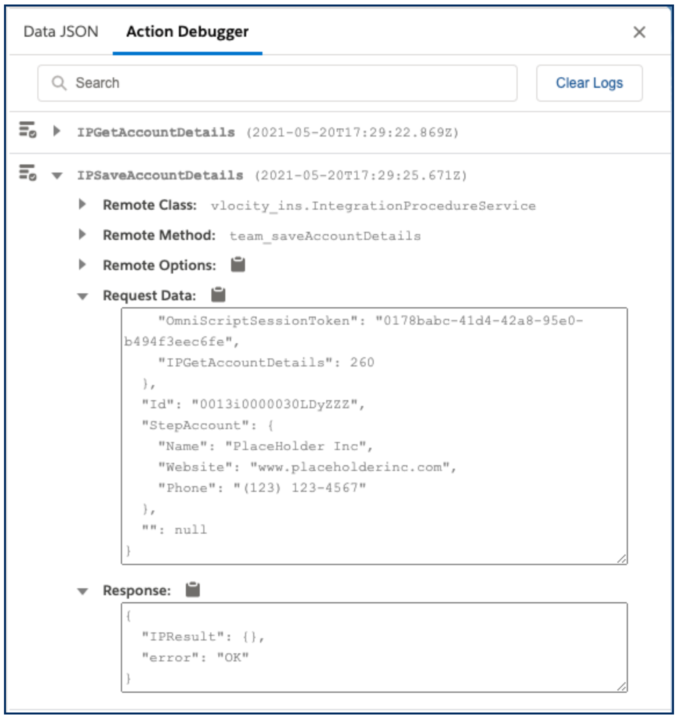
g. In the Canvas, click the **Next** button.

h. In the Action Debugger, collapse the node IPGetAccountDetails, and expand the following nodes:

   i. **IPSaveAccountDetails**, then

   ii. **Request Data** (resize the box if you wish to see the data without scrolling)

   iii. **Response**



```
IPGetAccountDetails (2021-05-20T17:29:22.869Z)
IPSaveAccountDetails (2021-05-20T17:29:25.671Z)
    Remote Class: vlocity_ins.IntegrationProcedureService
    Remote Method: team_saveAccountDetails
    Remote Options:
    Request Data:
        "OmniScriptSessionToken": "0178babc-41d4-42a8-95e0-
b494f3eec6fe",
        "IPGetAccountDetails": 260
      },
      "Id": "0013i0000030LDyZZZ",
      "StepAccount": {
        "Name": "PlaceHolder Inc",
        "Website": "www.placeholderinc.com",
        "Phone": "(123) 123-4567"
      },
      "": null
    }
    Response:
    {
      "IPResult": {},
      "error": "OK"
    }
```

i. Notice the request input and response expected JSON output.

   i. The entire JSON from the OmniScript is being sent to the Integration Procedure.

   ii. For the response, no value is present. This Integration Procedure only saves data, so a response is not expected.

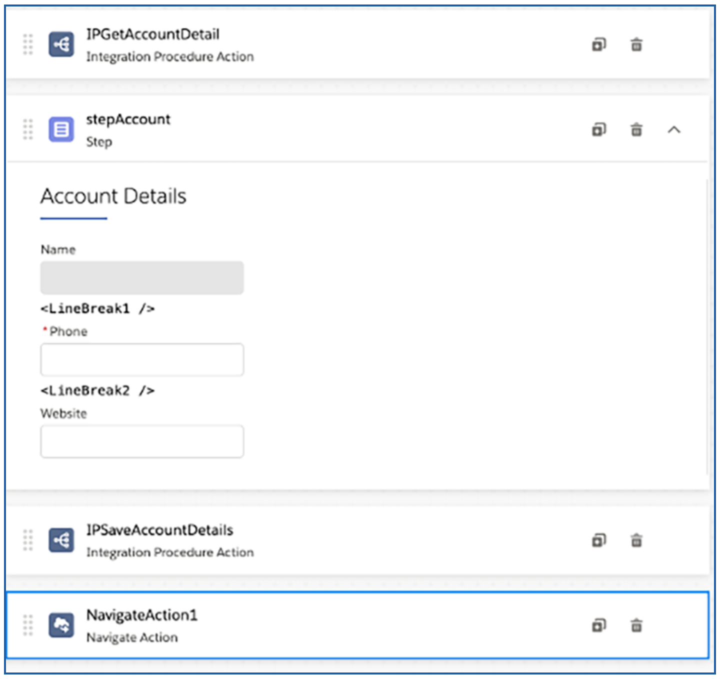
j. In the Header, click **Design**.

Task 4: Add a Navigate Action to the Edit Account OmniScript

Use a Navigation Action to specify where to send the user when the OmniScript finishes. Use the action to indicate how to open the record page for the Salesforce object that is coded into the ContextId by default.

Navigation Action elements don't run in Preview mode. You test the Navigation action when you run the OmniScript from a console.

1. Add an element to specify where to send the user when the OmniScript finishes.
   a. In the Build panel, locate a **Navigate Action** element and then drag it into the Canvas under **IPSaveAccountDetails**.


IPGetAccountDetail
Integration Procedure Action
stepAccount
Step
Account Details
Name
<LineBreak1 />
*Phone
<LineBreak2 />
Website
IPSaveAccountDetails
Integration Procedure Action
NavigateAction1
Navigate Action

   b. In the Navigate Action Properties panel, set or confirm the following values:

Field Name	Value
Page Reference Type	Record
Replace	✓

Object API Name	`Account`

c. In the Header, click **Activate Version**.

d. Click **Done** when the chevrons are all shaded and you see the Done confirmation icon.

e. Return to the **OmniScripts** tab.

f. Click the **LAST MODIFIED** column to sort the list by that value.

g. Expand **team/edit Account** and notice the **Active** check is next to V2 and that V1 is no longer active.

NOTE:
Once you are finished with an OmniScript, you must activate it, so it is available to you elsewhere. Remember that there is only one active version of an OmniScript at a time!

Review

Confirm your understanding by answering these questions.

1. What is the merge field syntax for OmniScript?
2. What are some uses for the OmniScript Action Debugger?
3. What does a Navigate Action do?

Create the Edit Contact OmniScript

Create a team/editContact OmniScript to solidify what you learned when you created the Edit Account OmniScript. We've summarized the requirements; refer to the step-by-step instructions above if you get stuck.

1. Build the basic structure of the OmniScript in the OmniScript Designer using descriptive labels. Use the following elements, element names, and settings*:

Element	Element Name	Setting
Setup panel		Hide Step Chart = Yes
Step	`StepContact`	
Text	`Name`	Read-only
Email	`Email`	Required
Telephone	`Phone`	

- Use line breaks and set control widths as you see fit.

*If no value is presented, choose your own (for example, for the Labels of each element).

2. Link the Edit Contact OmniScript to an Integration Procedure to get contact data from Salesforce. Add an Integration Procedure Action to your version of the team/editContact OmniScript with these settings:

Setting	Value
Name	`IPGetContactDetails`
Label	`IPGetContactDetails`
Integration Procedure	**team_getContactDetails**
Remote Properties > Extra Payload	`ContactId = %ContextId%`

Response JSON Path	`Contact|1`
Response JSON Node	`VlocityNoRootNode`

3. Verify that the OmniScript populates with the stub data.

   Hint: check the element names match the data field names and do you have to remove something from the Phone element to ensure the phone number displays?

4. Add a second Integration Procedure Action that saves the updated contact details. Use these settings.

Setting	Value
Name	`IPSaveContactDetails`
Integration Procedure	**team_saveContactDetails**

5. Use the Action Debugger to verify that the correct data is sent to the Integration Procedure.
6. Add a Navigate Action to end the OmniScript. Don't forget to configure it correctly for the Contact Object.
7. Activate Version 2 of the OmniScript.

Create the Edit Case OmniScript

Create a team/editCase OmniScript to solidify what you learned when you created the Edit Account OmniScript. We've summarized the requirements; refer to the step-by-step instructions above if you get stuck.

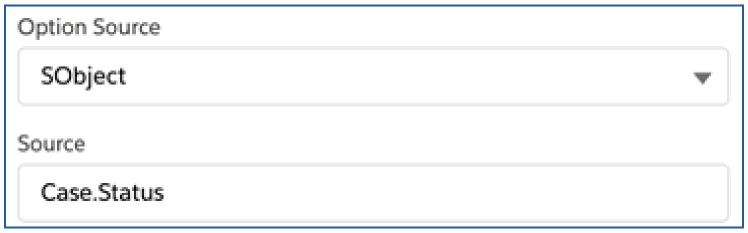
1. Build the basic structure of the OmniScript in the OmniScript Designer using descriptive labels. Use the following elements, element names, and settings*:

Element	Name	Setting
Setup panel		Hide Step Chart = Yes
Step	`StepCase`	
Text	`CaseNumber`	Read-only
Text	`Subject`	Required
Date	`CreatedDate`	Read-Only
Select	`Status`	Required

*If no value is presented, choose your own (for example, for the Labels of each element).

- Use line breaks and set control widths as you see fit but try to have the fields arranged in 2 rows with 2 fields in each row.

2. Tie the Select element to the Salesforce sObject to pre-populate the dropdown list values. To do this, select **SObject** in the **Option Source** field and type `Case.Status` in the **Source** field.


Option Source
SObject
Source
Case.Status

3. Link the Edit Case OmniScript to an Integration Procedure to get case data from Salesforce. Add an Integration Procedure Action to your version of the team/editCase OmniScript with these settings.

Setting	Value
Name	`IPGetCaseDetails`
Label	`IPGetCaseDetails`
Integration Procedure	**team_getCaseDetails**
Remote Properties > Extra Payload	`CaseId = %ContextId%`
Response JSON Path	`Case|1`
Response JSON Node	`VlocityNoRootNode`

4. Verify that the OmniScript populates with the stub data.

5. Link the Edit Case OmniScript to an Integration Procedure to get case data from Salesforce. Add an Integration Procedure Action to your version of the team/editCase OmniScript with these settings.

Setting	Value
Name	`IPSaveCaseDetails`
Integration Procedure	**team_saveCaseDetails**

6. Use the Action Debugger to verify that the correct data is sent to the Integration Procedure.
7. Add a Navigate Action to end the OmniScript. Don’t forget to configure it correctly for the Case Object.
8. Activate Version 2 of the OmniScript.