ESM Developer

Exercise Guide

Version 1.0

TABLE OF CONTENTS

Preface

These training exercises are based on the Spring 2022 release of the Communications Cloud. For additional information about the topics covered in this module, see the documentation in Salesforce Industries Success Community at https://success.vlocity.com.

Overview

This training covers customizing and integrating Enterprise Sales Management (ESM). As you progress through this training, you complete practical, hands-on lab exercises designed for use with a training playground.

What You'll Learn

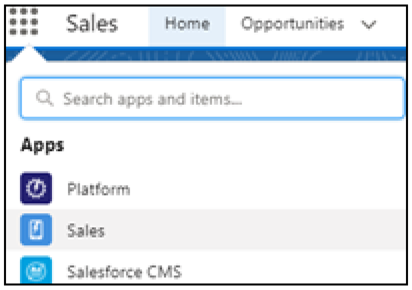
When you complete this training, you will be able to follow best practices to:

- Change themes using the Newport Design System.
- Create and implement custom LWCs in ESM.
- Customize ESM layouts, cards and OmniScripts to suit business needs.
- Configure ESM's out-of-the-box integrations.
- Customize ESM integration using DataRaptors, APIs and Integration Procedures.

Prerequisites

The prerequisites for this training include a solid understanding of basic Salesforce concepts and functionality. You should be competent with implementing the full range of OmniStudio tools and have experience developing application integrations.

Where You'll Work

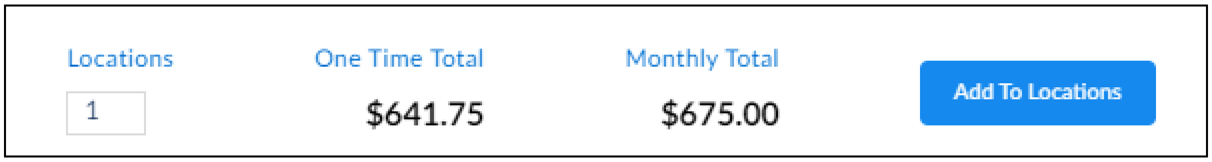


All the exercises assume you're working in the Sales application in your ESM playground. Open your playground, click on the **App Launcher** and select **Sales** from the list of apps to get started.

Exercise 1: Amend ESM Components using Newport Design System (NDS)

Scenario

You've been asked to update ESM's "look and feel" to suit your organization's corporate theme. The first thing you need to do is change the color of the text in the headings of the total bar of ESM quotes from gray to "science blue" (a color from your company themes), as shown here.



You know this is done using Newport Design System, but what do you change? And how do you change it?

Goals

- Set up and use Git.
- Set up the Newport Design System.
- Use Newport Design System to locate and implement style changes.

Tasks

1. Install the dependencies and clone NDS.
2. Identify components and code to be amended using Newport Storybook.
3. Build and deploy ESM customizations using NDS.

Time: 20 mins

Git and the Newport Design System

Use Newport Design System (NDS) to amend ESM's markup and CSS framework then apply it across user interfaces to fit your corporate theme.

The Storybook.js previewer is included as part of NDS to help with your customization and rebranding. It shows all the elements used in the ESM interfaces, and the associated CSS code. Once you've identified what you need to change, use the instructions supplied with NDS to amend the code using an editor such as Visual Studio, then deploy it to your org.

In the exercise, you'll install the required applications and review the ESM-specific components. Detailed step-by-step instructions on CSS builds and deployment are given in the NDS documentation, and will not be covered here.

Task 1: Install the dependencies and clone NDS

View the instructions to install NDS on your local computer here:
https://github.com/vlocityinc/newport-design-system

The steps are:

1. Install Git. Download the appropriate version of git for your computer from https://git-scm.com/downloads and run the installation.

2. If you don't already have Node.js (version 12 or later) installed, download the appropriate version for your computer from https://nodejs.org/en/download/ and run the installation.

3. Install gulp. Open your command prompt and run:

   ```
   npm install --global gulp-cli
   ```

4. Clone the Newport Design System from the Git repository. From your command prompt run:

   ```
   git clone https://github.com/vlocityinc/newport-design-system.git
   ```

5. Switch to your new folder. From your command prompt run:

```
cd newport-design-system
```

6. Switch to the right branch for your version of the package.
    a. To check the branches available, from your command prompt run:

    ```
    git branch -l
    ```

    b. To check out the appropriate version, run the checkout command. For example, v236 is the branch for version 236, so from your command prompt run:

    ```
    git checkout v236
    ```

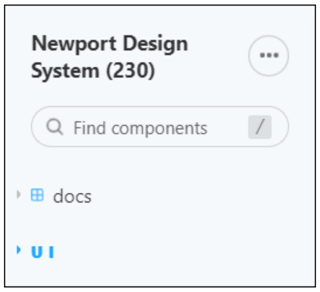
7. Install the dependencies. From your command prompt, run:

```
npm install
```

8. Launch the Newport Storybook. From your command prompt, run:

```
npm start
```

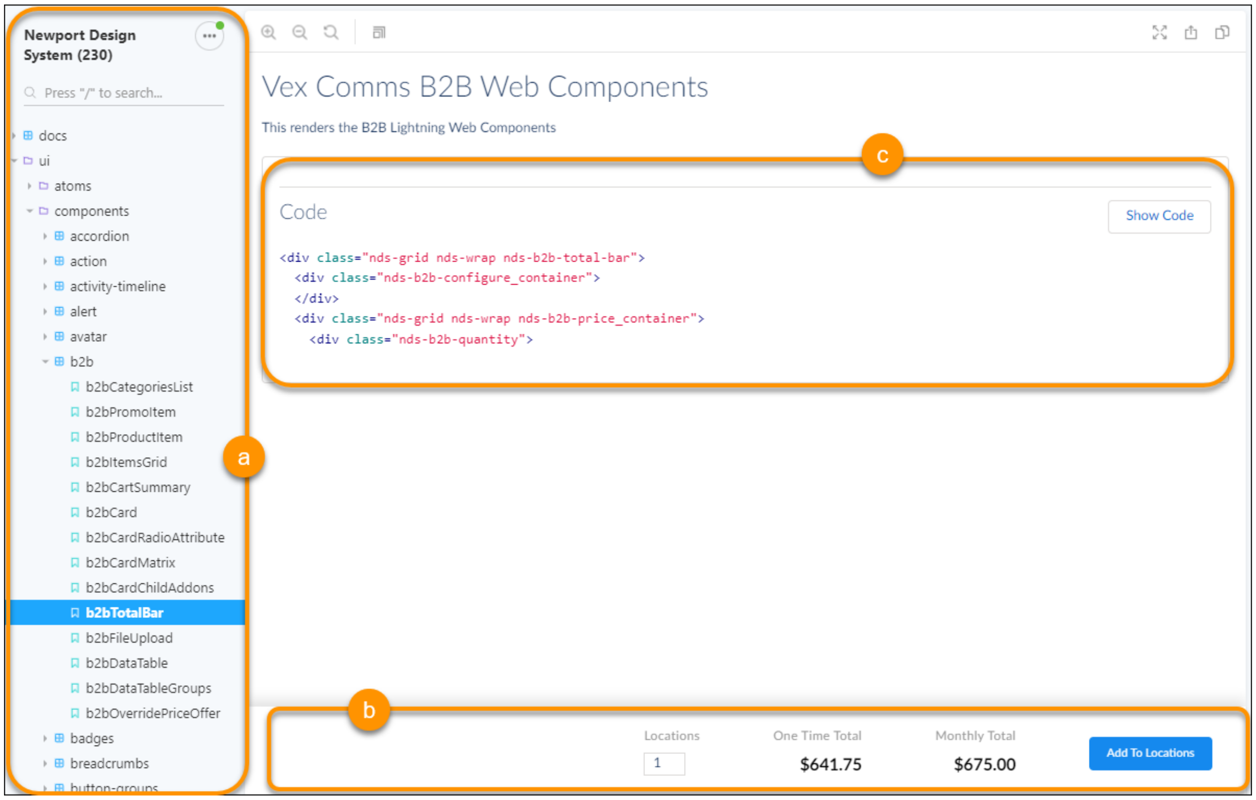
Task 2: Identify components and code to be amended using the Newport Storybook



If you've successfully completed Task 1, you should now see the Newport Storybook.

On the left is the navigation pane.

1. From the navigation pane (a) expand the **UI** list to see all elements of the UI.



```
<div class="nds-grid nds-wrap nds-b2b-total-bar">
  <div class="nds-b2b-configure_container">
  </div>
  <div class="nds-grid nds-wrap nds-b2b-price_container">
    <div class="nds-b2b-quantity">
```

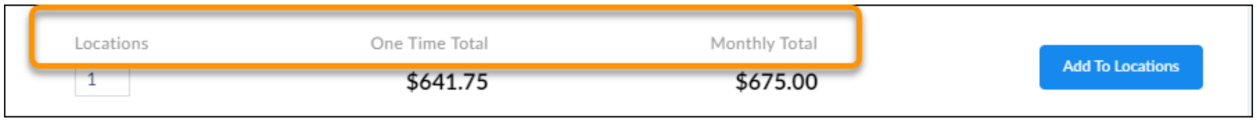
2. In the navigation pane (a) select **components** to view all the UI components. Components unique to ESM are contained in the b2b category, so expand the **b2b** list.

3. Scroll down the list of b2b components and select the **b2bTotalBar**.

4. Now you see a preview of the b2bTotalBar component (b) and the code to deliver that component (c). Click **Show Code** in the code panel to see a more detailed view of the code.

What is the name of the component you need to configure in your code editor to change the color of these labels to science blue?

Answer:	



Task 3: Build and deploy ESM customizations using NDS

Now you know what component you need to amend, open your code editor (in the examples it's Visual Studio) and make the changes.

1. In Visual Studio, open the **_b2bTotalBar.scss** component, which you can find in the **UI/components/b2b/components/configure-offer** folder.

2. Locate the .nds-b2b-configure_label element you decided to edit in the previous task and amend the color from **nobel** to **science-blue**:

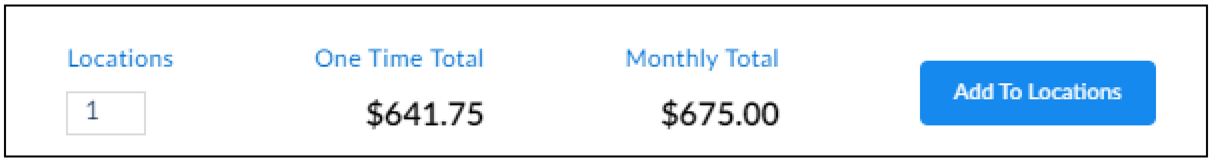
   Change code from:

```
.nds-b2b-configure_label {
  color: $nobel;
  font-size: 0.75rem;
  letter-spacing: 0.4px;
  margin-bottom: 0.4rem;
}
```

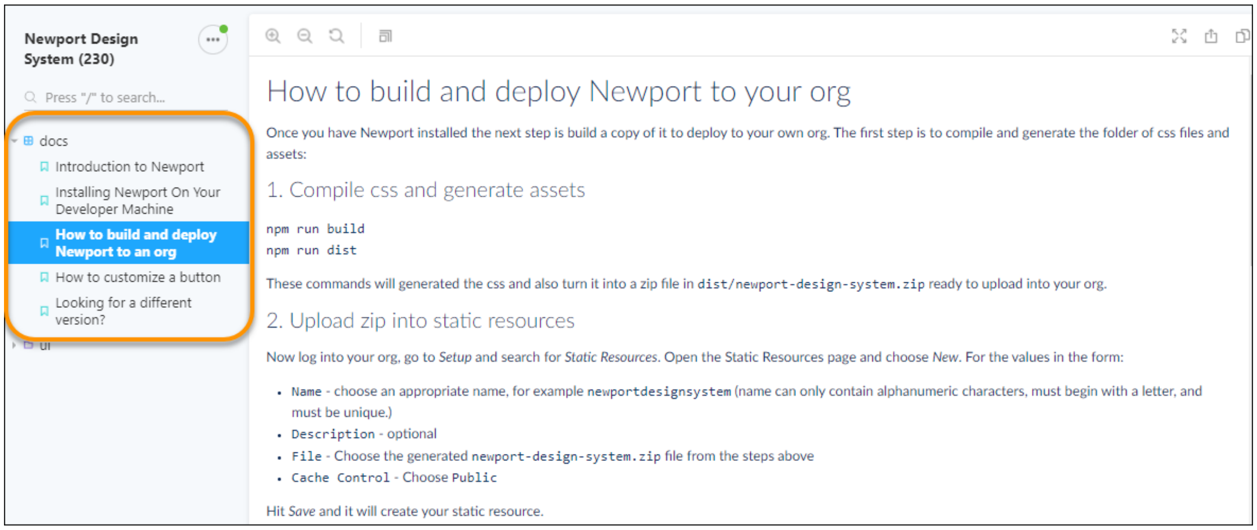
To

```
.nds-b2b-configure_label {
  color: $science-blue;
  font-size: 0.75rem;
  letter-spacing: 0.4px;
  margin-bottom: 0.4rem;
}
```

3. Save your changes. Return to Storybook and view the b2bTotalBar component. Notice the color of the labels has changed from gray to blue.



4. The final step is to build and deploy Newport to your org. In Storybook, navigate to the **docs** folder and select **How to build and deploy Newport to an org**.


Newport Design System (230)
Press "/" to search...
docs
Introduction to Newport
Installing Newport On Your Developer Machine
How to build and deploy Newport to an org
How to customize a button
Looking for a different version?
ui

How to build and deploy Newport to your org

Once you have Newport installed the next step is build a copy of it to deploy to your own org. The first step is to compile and generate the folder of css files and assets:

1. Compile css and generate assets

```
npm run build
npm run dist
```

These commands will generated the css and also turn it into a zip file in `dist/newport-design-system.zip` ready to upload into your org.

2. Upload zip into static resources

Now log into your org, go to *Setup* and search for *Static Resources*. Open the Static Resources page and choose *New*. For the values in the form:

- `Name` - choose an appropriate name, for example `newportdesignsystem` (name can only contain alphanumeric characters, must begin with a letter, and must be unique.)
- `Description` - optional
- `File` - Choose the generated `newport-design-system.zip` file from the steps above
- `Cache Control` - Choose Public

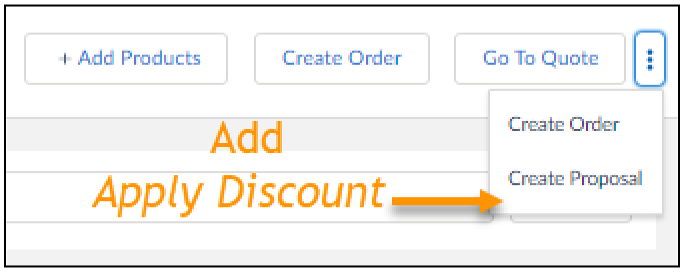
Hit *Save* and it will create your static resource.

Follow the steps provided.

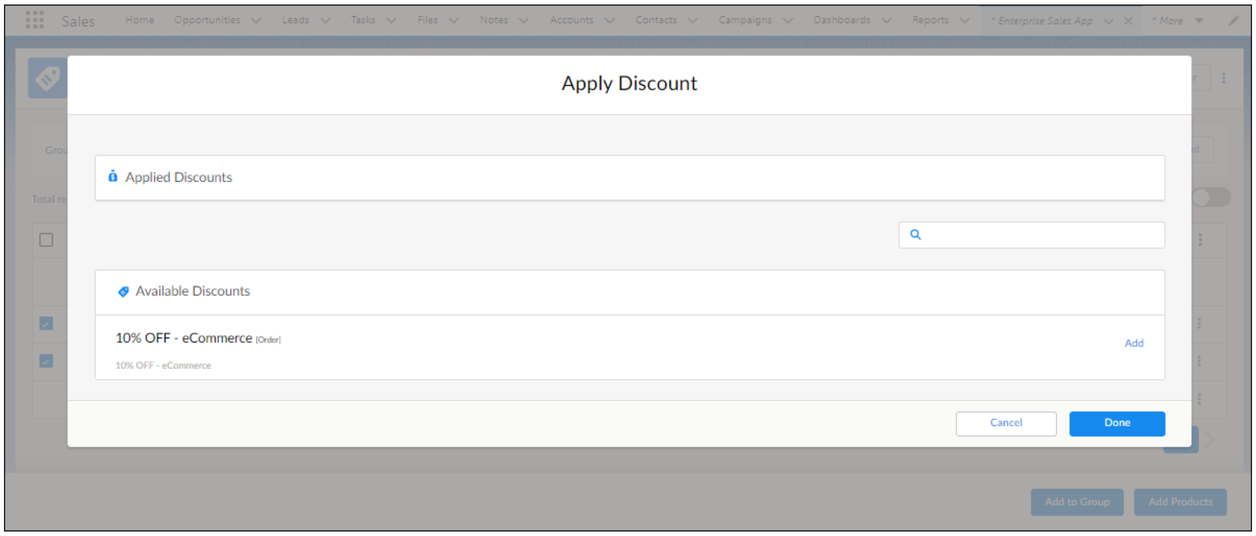
Exercise 2: Add “Apply Discount” to the Cart Action List

Scenario

Now that you’ve implemented the requested design changes, your boss is keen for you to add some new functionality to ESM. Your organization wants an “Apply Discount” option added to the action list on the Cart, as shown here.



When the Apply Discount option is selected from the action list, the Apply Discount modal will be displayed:



The salesperson can select from the list of available discounts to apply to the customer quote.

Goals

- Create new LWCs that extend those in the managed package.
- Implement additional features in ESM using LWCs and OmniStudio Cards.

Tasks

1. Create a new LWC to extend the b2bCartSummary LWC from the managed package.
2. Configure the new custom LWC.
3. Create and configure an LWC to extend the b2bSampleApp LWC.
4. Clone and configure the b2bSampleApp card

Time: 30 mins

ESM LWC Customizations

In this example you'll need to extend the existing b2bCartSummary LWC, and the associated b2bSampleApp LWC. However, both these LWCs are part of the managed package, so you can't directly amend them. Instead, you'll create a new LWC for each, which extends the LWC from the managed package. You'll then be able to add the required action to the action list array associated with the action button in the Cart summary by amending your new custom b2bCartSummary LWC.

NOTE:
When extending an LWC, the amount of coding required varies:

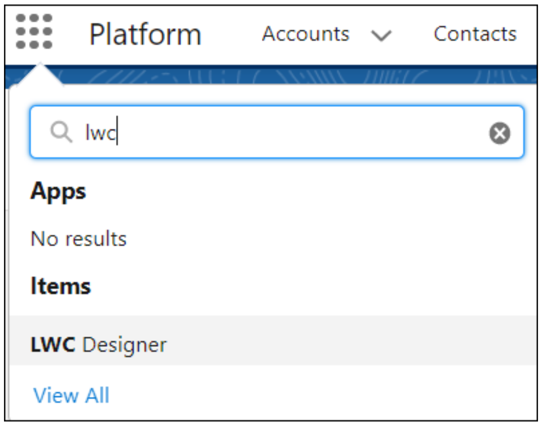
- If slots are available for you to extend, then you need only write the HTML and JavaScript required to implement the change.
- If no slots are available, the entire HTML template must be replaced, and the JavaScript required to implement the change should be added.

In the example you work through here, a slot is provided to extend, so we include only part of the whole HTML template.

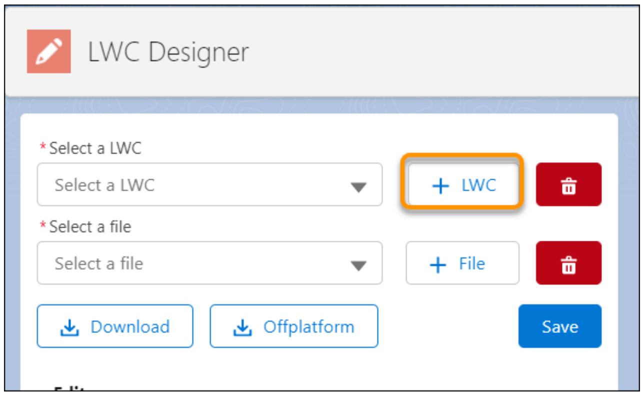
Within your organization, you'll need to determine the naming conventions you use for customization. In this example, we've added "c" to the beginning of the names to help identify them.

Task 1: Create a new LWC by extending the b2bCartSummary LWC

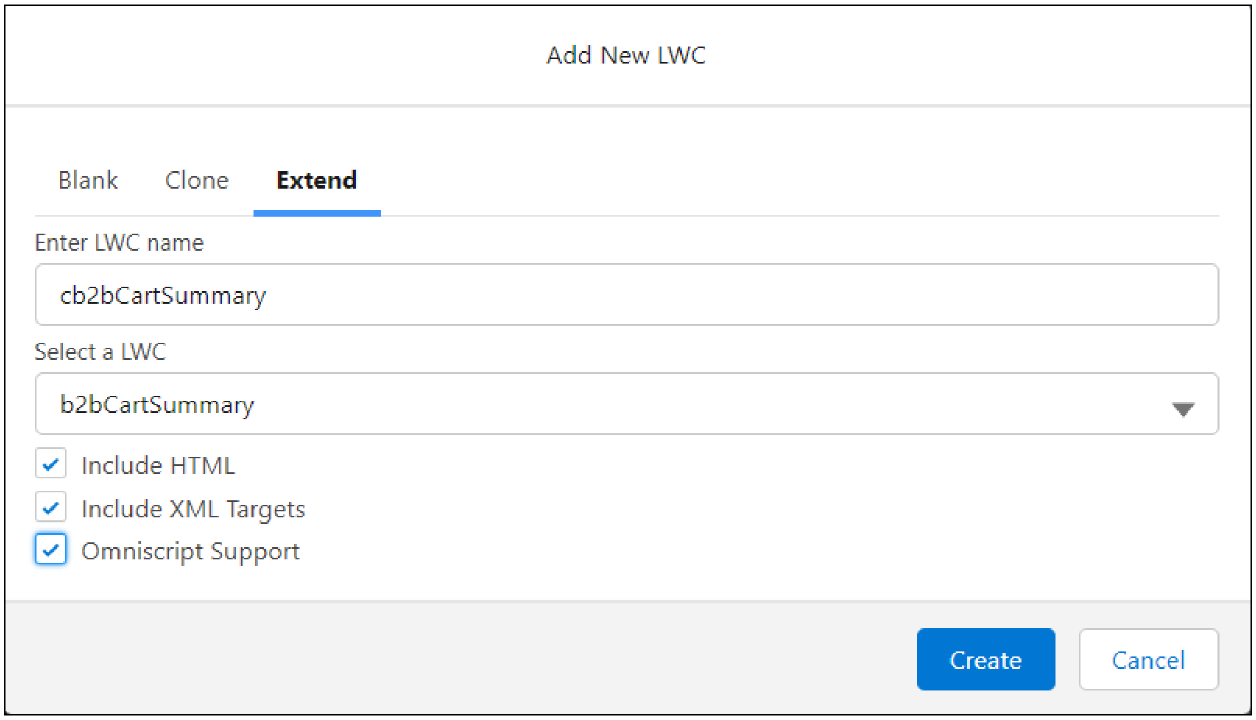
1. From the **App Launcher** type `LWC` in the search box, and select **LWC Designer**.



2. In the LWC Designer, select **+LWC** to create a new LWC.



3. In the **Add New LWC** dialog box, select **Extend**.



Add New LWC

Enter LWC name: cb2bCartSummary

Select a LWC: b2bCartSummary

✓ Include HTML
✓ Include XML Targets
✓ Omniscript Support

4. Complete the details of your new LWC:

Field	Data
Enter LWC name	`cb2bCartSummary`
Select a LWC	b2bCartSummary
Include HTML	✔
Include XML Targets	✔
OmniScript Support	✔

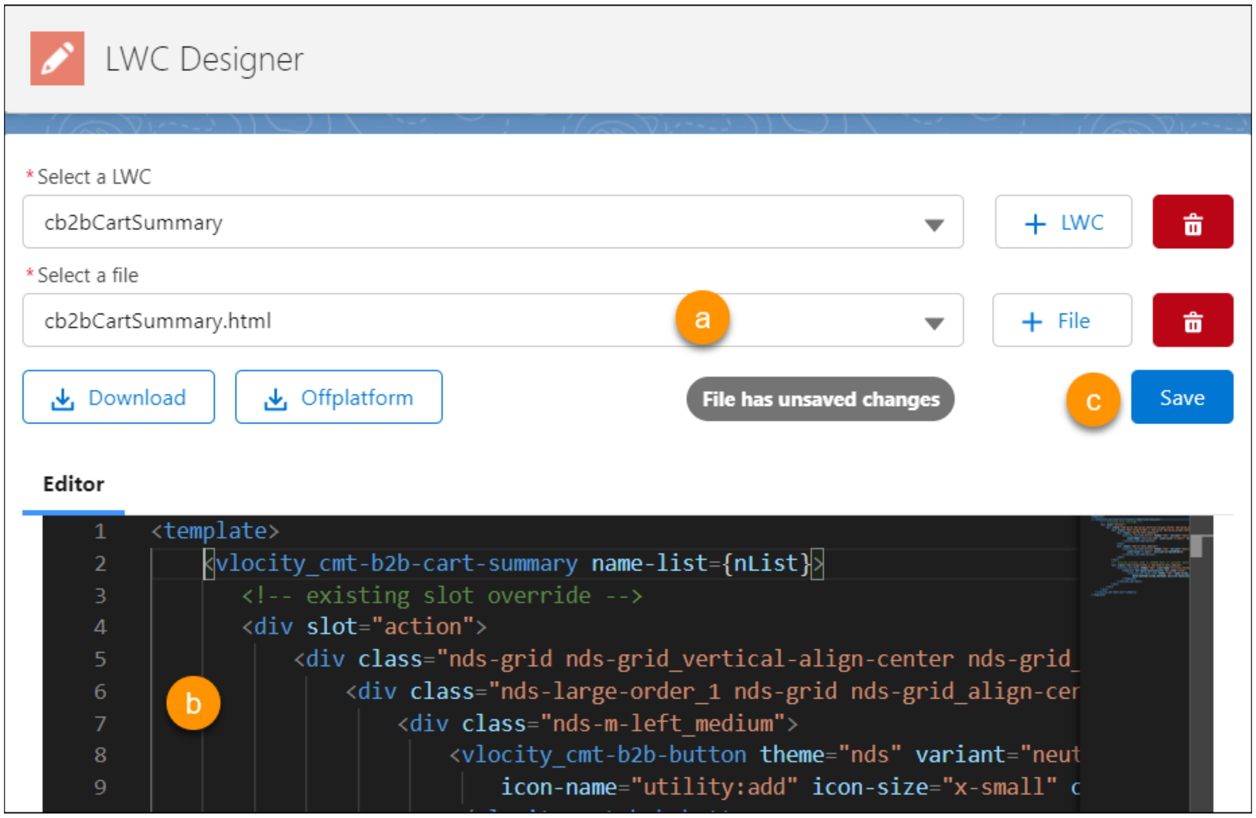
5. Click **Create**.

   Your new custom cb2bCartSummary LWC is created and is opened in LWC Designer.

Task 2: Configure the new custom LWC

In LWCs there are three files you need to configure: the HTML file, the JavaScript file, and the metadata file.

1. Configure the HTML file.
   a. In **Select a file**, select **cb2bCartSummary.html**.



```
<template>
    <vlocity_cmt-b2b-cart-summary name-list={nList}>
        <!-- existing slot override -->
        <div slot="action">
            <div class="nds-grid nds-grid_vertical-align-center nds-grid_
                <div class="nds-large-order_1 nds-grid nds-grid_align-cen
                    <div class="nds-m-left_medium">
                        <vlocity_cmt-b2b-button theme="nds" variant="neut
                            icon-name="utility:add" icon-size="x-small" c
```

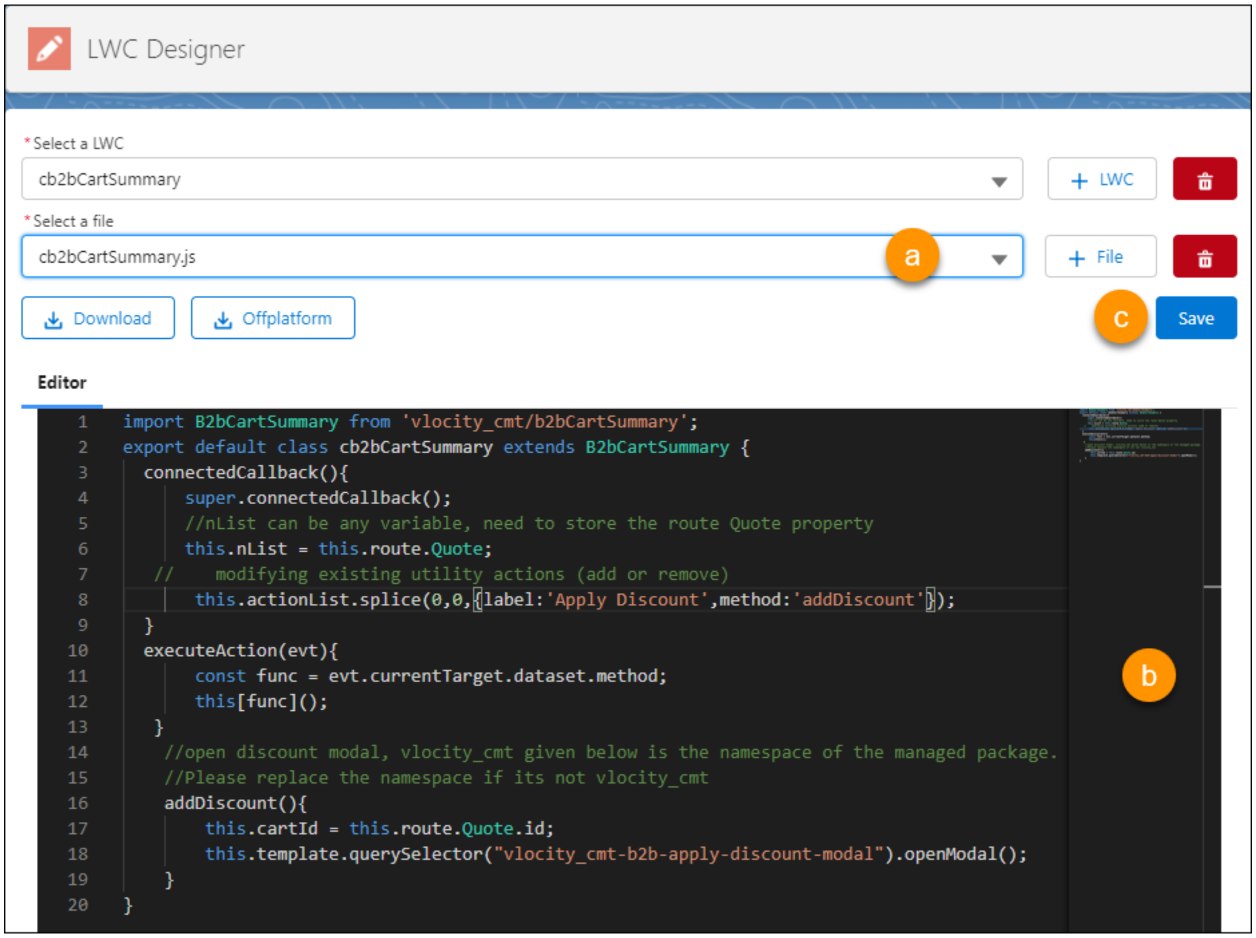
   b. In the editor, enter your required HTML. Unsure where to start? - use the sample HTML given below.

   c. Once you're done, click **Save** to save your changes.

Sample HTML code for cb2bCartSummary

```
<template>
    <vlocity_cmt-b2b-cart-summary name-list={nList}>
       <!-- existing slot override -->
       <div slot="action">
           <div class="nds-grid nds-grid_vertical-align-center
nds-grid_align-end">
               <div class="nds-large-order_1 nds-grid nds-grid_align-center
nds-wrap">
                   <div class="nds-m-left_medium">
                       <vlocity_cmt-b2b-button theme="nds" variant="neutral"
label={labels.CMEXAddProducts}
                           icon-name="utility:add" icon-size="x-small"
onclick={addProduct}>
                       </vlocity_cmt-b2b-button>
                   </div>
                   <div class="nds-m-left_medium">
                       <vlocity_cmt-b2b-button theme="nds" variant="neutral"
label={labels.CMEXCreateOrders}
                           icon-size="x-small" onclick={createOrder}>
                       </vlocity_cmt-b2b-button>
                   </div>
               </div>
               <!-- utility actions code is needed here if customer wants to
reuse the existing feature provided, along with the new button
implementation-->
               <div class="nds-large-order_2 nds-b2b-m-left_medium">
                   <vlocity_cmt-menu theme="nds"
icon-name="utility:threedots_vertical" position="right">
                       <template for:each={actionList} for:item="item">
                           <vlocity_cmt-menu-item theme="nds" name={item.label}
key={item.label} record={item}
                               data-method={item.method}
onclick={executeAction}></vlocity_cmt-menu-item>
                       </template>
                   </vlocity_cmt-menu>
               </div>
           </div>
       </div>
   </vlocity_cmt-b2b-cart-summary>
   <vlocity_cmt-b2b-apply-discount-modal cart-id={cartId}
onapplieddiscount={handleDiscount}></vlocity_cmt-b2b-apply-discount-modal>
</template>
```

2. Configure the JavaScript for cb2bCartSummary.



*Select a LWC: cb2bCartSummary

*Select a file: cb2bCartSummary.js

```
import B2bCartSummary from 'vlocity_cmt/b2bCartSummary';
export default class cb2bCartSummary extends B2bCartSummary {
  connectedCallback(){
      super.connectedCallback();
      //nList can be any variable, need to store the route Quote property
      this.nList = this.route.Quote;
   //    modifying existing utility actions (add or remove)
       this.actionList.splice(0,0,{label:'Apply Discount',method:'addDiscount'});
  }
  executeAction(evt){
       const func = evt.currentTarget.dataset.method;
       this[func]();
   }
    //open discount modal, vlocity_cmt given below is the namespace of the managed package.
    //Please replace the namespace if its not vlocity_cmt
    addDiscount(){
        this.cartId = this.route.Quote.id;
        this.template.querySelector("vlocity_cmt-b2b-apply-discount-modal").openModal();
    }
}
```

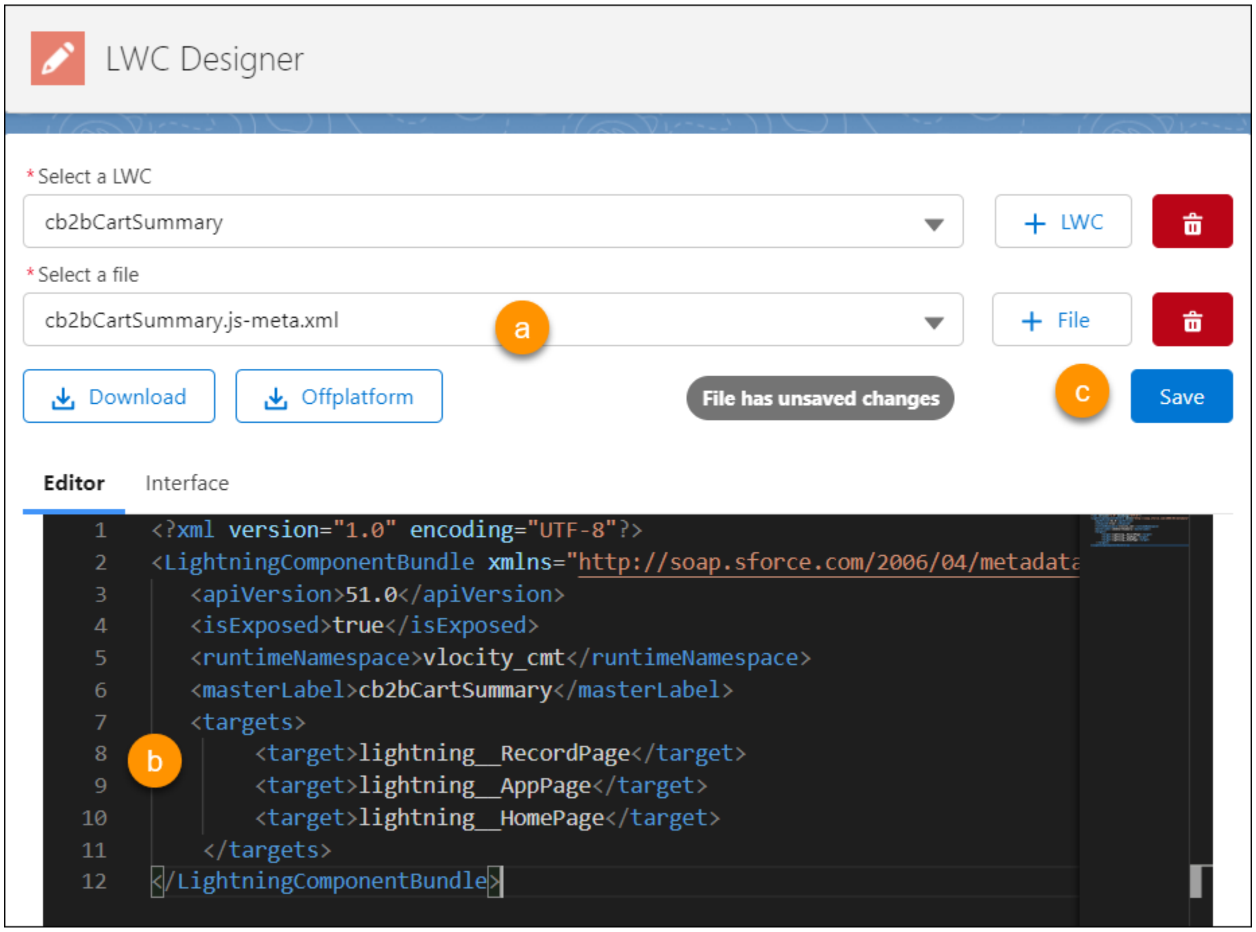
a. In **Select a file**, select **cb2bCartSummary.js**.
b. In the editor, enter the required JavaScript. In this example, the action addDiscount is added with the label Apply Discount. The addDiscount method opens the discount modal for the quote. You can use the sample JavaScript to get you started.
c. Once you've added your JavaScript, click **Save**.

Sample JavaScript to Add an Apply Discount Action

```
import b2bCartSummary from 'vlocity_cmt/b2bCartSummary';
export default class cb2bCartSummary extends b2bCartSummary {
  connectedCallback(){
      super.connectedCallback();
      //nList can be any variable, need to store the route Quote
property
      this.nList = this.route.Quote;
   //    modifying existing utility actions (add or remove)
       this.actionList.splice(0,0,{label:'Apply
Discount',method:'addDiscount'});
  }
  executeAction(evt){
       const func = evt.currentTarget.dataset.method;
       this[func]();
   }
    //open discount modal, vlocity_cmt given below is the namespace
of the managed package.
    //Please replace the namespace if its not vlocity_cmt
    addDiscount(){
        this.cartId = this.route.Quote.id;

this.template.querySelector("vlocity_cmt-b2b-apply-discount-modal").o
penModal();
    }
}
```

3. Update the metadata XML file for cb2bCartSummary

The final step in configuring this LWC is to configure the metadata XML file.



```xml
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata
   <apiVersion>51.0</apiVersion>
   <isExposed>true</isExposed>
   <runtimeNamespace>vlocity_cmt</runtimeNamespace>
   <masterLabel>cb2bCartSummary</masterLabel>
   <targets>
        <target>lightning__RecordPage</target>
        <target>lightning__AppPage</target>
        <target>lightning__HomePage</target>
    </targets>
</LightningComponentBundle>
```

a. In **Select a file**, select **cb2bCartSummary.js-meta.xml**.
b. In the editor, update the metadata. Ensure that the metadata specifies **isExposed** as **true** and the runtimeNamespace as **vlocity_cmt**. Also check the **API version**. There is sample metadata code below you can copy and paste into your editor as a starting point.
c. Click **Save**.

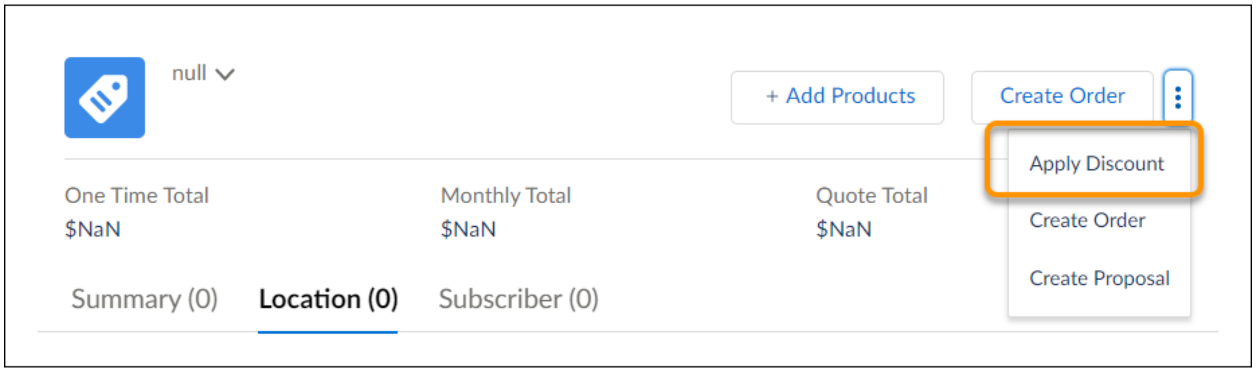
Sample Metadata Code for cb2bCartSummary

```
<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">

   <apiVersion>51.0</apiVersion>

   <isExposed>true</isExposed>

   <runtimeNamespace>vlocity_cmt</runtimeNamespace>

   <masterLabel>cb2bCartSummary</masterLabel>

   <targets>

        <target>lightning__RecordPage</target>

        <target>lightning__AppPage</target>

        <target>lightning__HomePage</target>

    </targets>

</LightningComponentBundle>
```

4: Review Your Changes

You’ve probably noticed error messages appearing in the preview pane to the right of the configuration pane in the LWC Designer as you’ve been making your amendments.

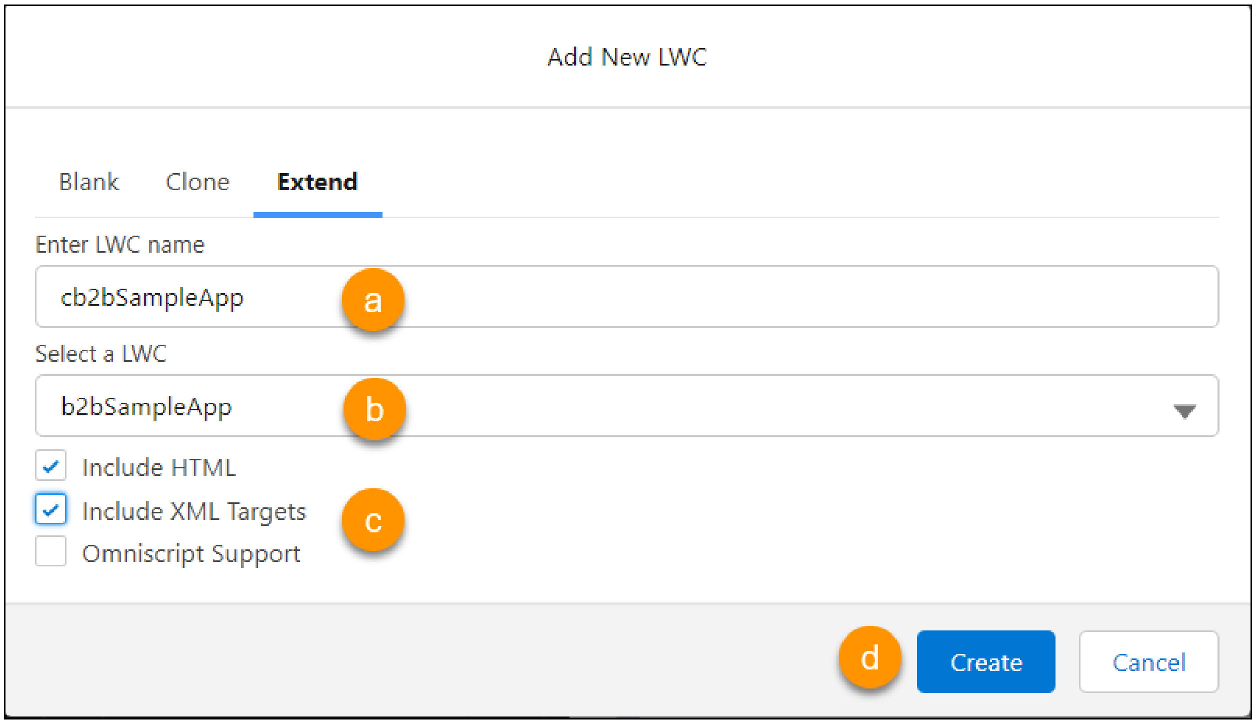
Once you’ve saved your changes, refresh your browser to see a preview of the Cart summary, with your new action listed. You may need to reselect your LWC to preview it.



Task 3: Create and configure an LWC to extend the b2bSampleApp LWC

Now that you've created a new cb2bCartSummary to extend the existing LWC, you'll want to ensure that the parent LWC, b2bSampleApp, refers to your new LWC. If you haven't already, you'll need to create a new LWC extending the b2bSampleApp. Then you'll need to configure your cb2bSampleApp to include the new cb2bCartSummary.

1. Follow steps 1 to 3 from Task 1 of this exercise to create a new LWC, extending an existing LWC. Then complete the new LWC details:

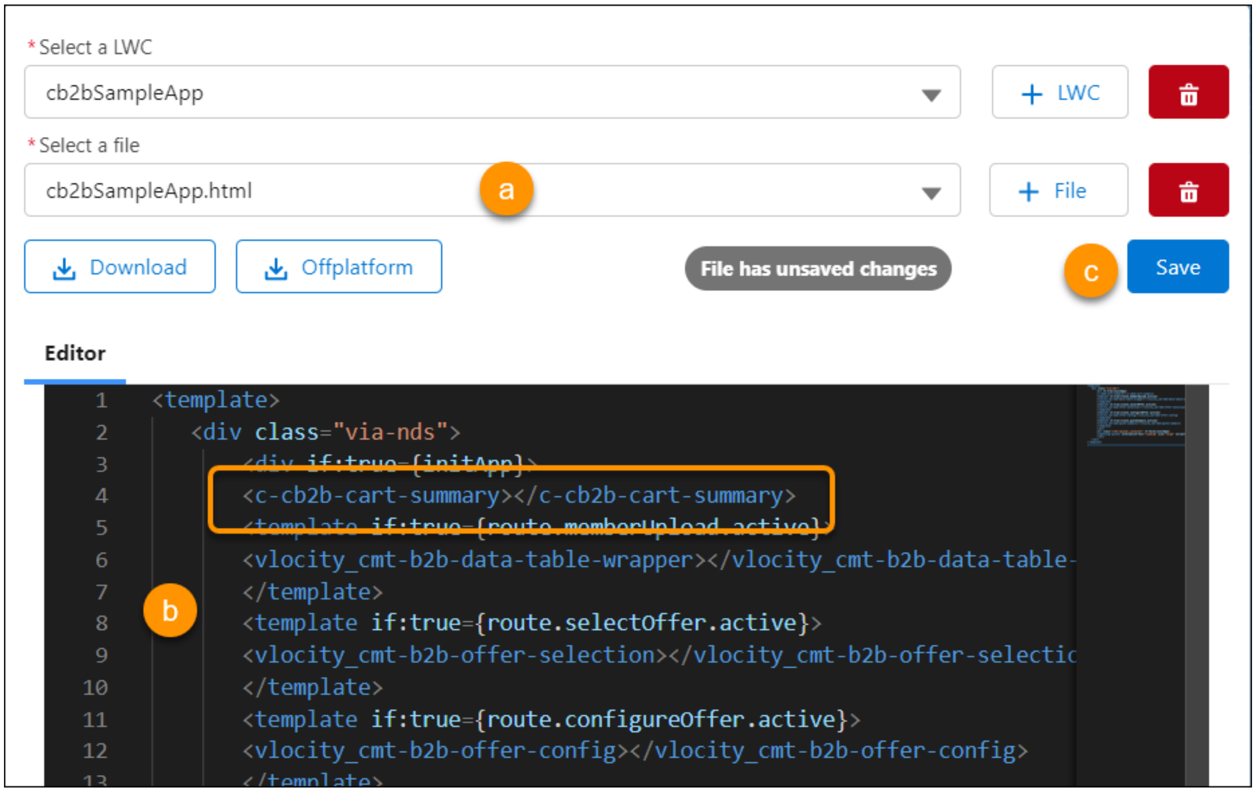


Enter LWC name: cb2bSampleApp
Select a LWC: b2bSampleApp
[✓] Include HTML
[✓] Include XML Targets
[ ] Omniscript Support

Field	Data
Enter LWC Name	`cb2bSampleApp`
Select a LWC	b2bSampleApp
Include HTML	✔
Include XML Targets	✔

OmniScript Support	✔

2. Click **Create**. A new custom LWC is created and is opened in LWC Designer.
3. Configure the HTML file for cb2bSampleApp.



```
<template>
    <div class="via-nds">
        <div if:true={initApp}>
        <c-cb2b-cart-summary></c-cb2b-cart-summary>
        <template if:true={route.memberUpload.active}>
        <vlocity_cmt-b2b-data-table-wrapper></vlocity_cmt-b2b-data-table-
        </template>
        <template if:true={route.selectOffer.active}>
        <vlocity_cmt-b2b-offer-selection></vlocity_cmt-b2b-offer-selectic
        </template>
        <template if:true={route.configureOffer.active}>
        <vlocity_cmt-b2b-offer-config></vlocity_cmt-b2b-offer-config>
        </template>
```

a. In **Select a file**, select **cb2bSampleApp.html.**

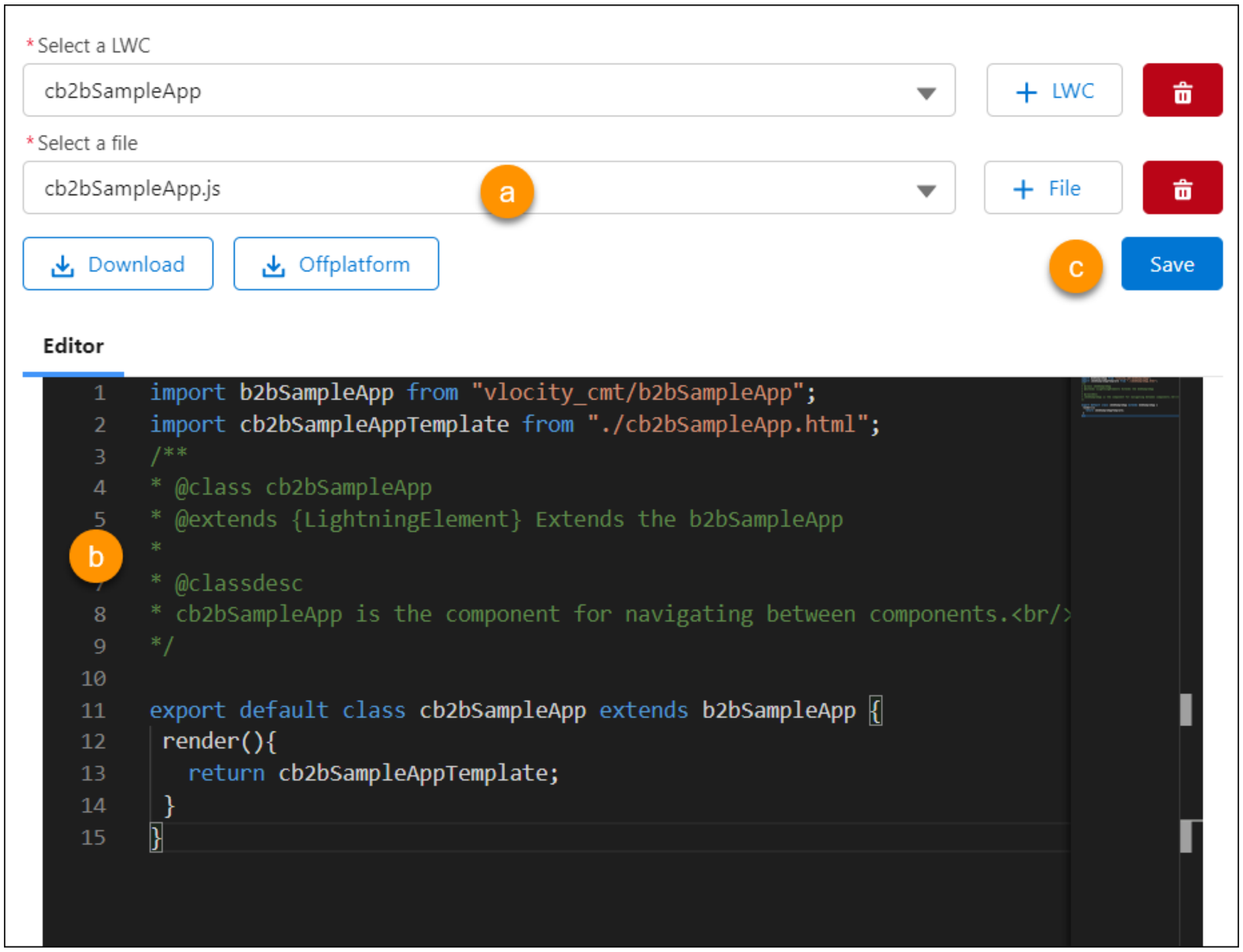
b. In the editor, enter the required HTML. You can use the sample HTML below to create this file. Notice the cb2b-cart-summary is called, replacing the original b2b-cart-summary.

c. Click **Save** to save your changes.

Sample HTML code for cb2bSampleApp

```
<template>

   <div class="via-nds">

       <div if:true={initApp}>

       <c-cb2b-cart-summary></c-cb2b-cart-summary>

       <template if:true={route.memberUpload.active}>

<vlocity_cmt-b2b-data-table-wrapper></vlocity_cmt-b2b-data-table-wrap
per>

       </template>

       <template if:true={route.selectOffer.active}>

<vlocity_cmt-b2b-offer-selection></vlocity_cmt-b2b-offer-selection>

       </template>

       <template if:true={route.configureOffer.active}>

       <vlocity_cmt-b2b-offer-config></vlocity_cmt-b2b-offer-config>

       </template>

       <template if:true={route.quoteSummary.active}>

<vlocity_cmt-b2b-quote-summary></vlocity_cmt-b2b-quote-summary>

       </template>

       </div>

       <div class="slds-spinner_container" if:false={initApp}>

       <lightning-spinner alternative-text="Loading" size="large"
variant="brand"></lightning-spinner>

       </div>

   </div>

</template>
```

3. Configure the JavaScript file for cb2bSampleApp.



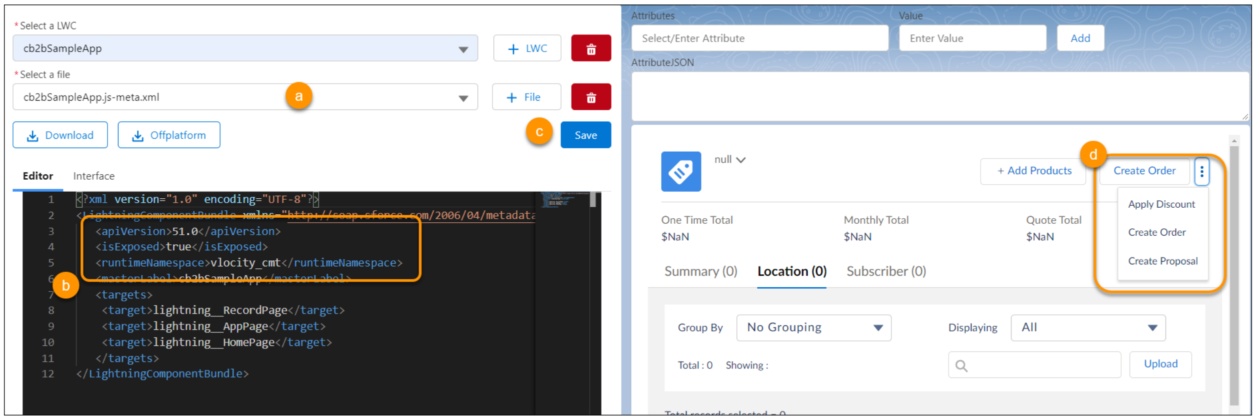
```
import b2bSampleApp from "vlocity_cmt/b2bSampleApp";
import cb2bSampleAppTemplate from "./cb2bSampleApp.html";
/**
* @class cb2bSampleApp
* @extends {LightningElement} Extends the b2bSampleApp
*
* @classdesc
* cb2bSampleApp is the component for navigating between components.<br/>
*/

export default class cb2bSampleApp extends b2bSampleApp {
 render(){
   return cb2bSampleAppTemplate;
 }
}
```

a. In **Select a file**, select **cb2bSampleApp.js**.
b. In the editor, enter the required JavaScript. You can use the sample JavaScript below.
c. Click **Save** to save your changes.

Sample JavaScript for cb2bSampleApp

```
import b2bSampleApp from "vlocity_cmt/b2bSampleApp";

import cb2bSampleAppTemplate from "./cb2bSampleApp.html";

/**

* @class cb2bSampleApp

* @extends {LightningElement} Extends the b2bSampleApp

*

* @classdesc

* cb2bSampleApp is the component for navigating between
components.<br/><br/>

*/


export default class cb2bSampleApp extends b2bSampleApp {

 render(){

   return cb2bSampleAppTemplate;

 }

}
```

3. Update the metadata XML file for cb2bSampleApp.



```
<?xml version="1.0" encoding="UTF-8"?>
<LightningComponentBundle xmlns="http://soap.sforce.com/2006/04/metadata
  <apiVersion>51.0</apiVersion>
  <isExposed>true</isExposed>
  <runtimeNamespace>vlocity_cmt</runtimeNamespace>
  <masterLabel>cb2bSampleApp</masterLabel>
  <targets>
   <target>lightning__RecordPage</target>
   <target>lightning__AppPage</target>
   <target>lightning__HomePage</target>
  </targets>
</LightningComponentBundle>
```

a. In **Select a file**, select **cb2bSampleApp.js-meta.xml**.

b. In the editor, update the metadata. Ensure that the metadata specifies:

   i. **isExposed** as **true**

   ii. **runtimeNamespace** as **vlocity_cmt**.

   iii. The correct API version.

   You can use the sample XML below to update this file.

c. Click **Save** to save your changes.

d. Refresh your browser and reselect the cb2bSampleApp LWC. You should see a preview of your new sample app with the added action button displayed.

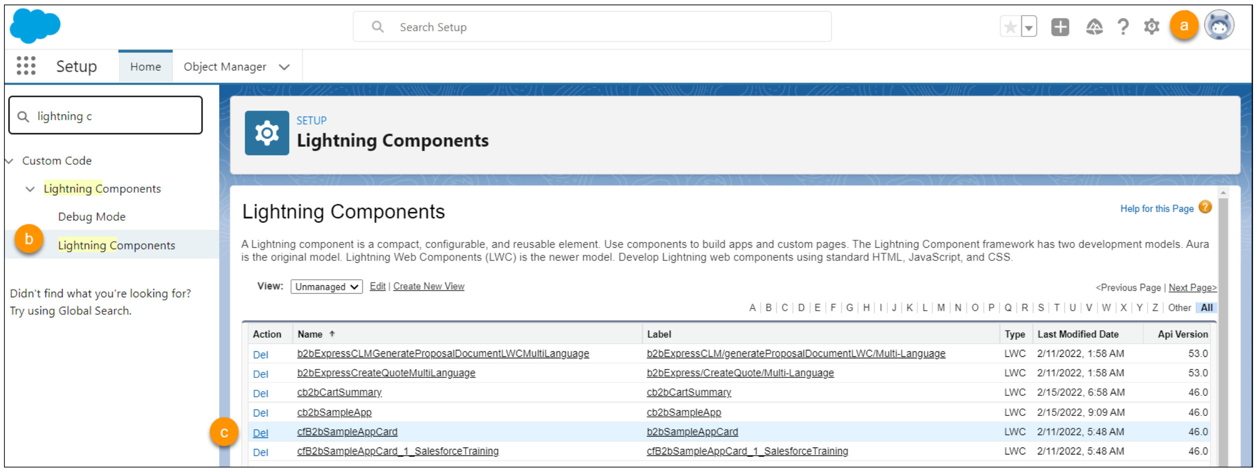
Sample XML for cb2bSampleApp

```
<?xml version="1.0" encoding="UTF-8"?>

<LightningComponentBundle
xmlns="http://soap.sforce.com/2006/04/metadata">

   <apiVersion>51.0</apiVersion>

   <isExposed>true</isExposed>

   <runtimeNamespace>vlocity_cmt</runtimeNamespace>

   <masterLabel>cb2bSampleApp</masterLabel>

   <targets>

    <target>lightning__RecordPage</target>

    <target>lightning__AppPage</target>

    <target>lightning__HomePage</target>

   </targets>

</LightningComponentBundle>
```

Task 4: Clone and configure the b2bSampleApp card

The b2bSampleApp card can't be updated directly, so start by cloning the card.

1. First remove the **cfB2bSampleAppCard** to avoid any build conflicts.



Lightning Components

A Lightning component is a compact, configurable, and reusable element. Use components to build apps and custom pages. The Lightning Component framework has two development models. Aura is the original model. Lightning Web Components (LWC) is the newer model. Develop Lightning web components using standard HTML, JavaScript, and CSS.

View: Unmanaged | Edit | Create New View

Action	Name	Label	Type	Last Modified Date	Api Version
Del	b2bExpressCLMGenerateProposalDocumentLWCMultiLanguage	b2bExpressCLM/generateProposalDocumentLWC/Multi-Language	LWC	2/11/2022, 1:58 AM	53.0
Del	b2bExpressCreateQuoteMultiLanguage	b2bExpress/CreateQuote/Multi-Language	LWC	2/11/2022, 1:58 AM	53.0
Del	cb2bCartSummary	cb2bCartSummary	LWC	2/15/2022, 6:58 AM	46.0
Del	cb2bSampleApp	cb2bSampleApp	LWC	2/15/2022, 9:09 AM	46.0
Del	cfB2bSampleAppCard	b2bSampleAppCard	LWC	2/11/2022, 5:48 AM	46.0
Del	cfB2bSampleAppCard_1_SalesforceTraining	cfB2bSampleAppCard_1_SalesforceTraining	LWC	2/11/2022, 5:48 AM	46.0

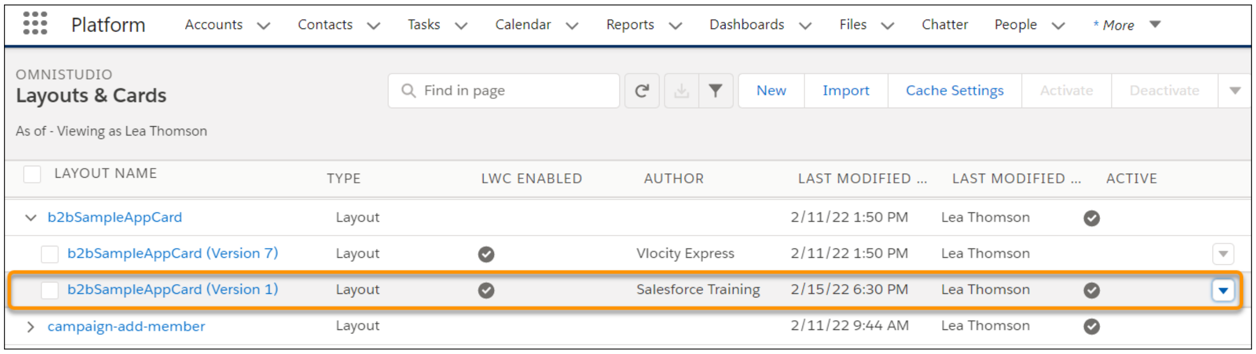
a. In your playground, click the **Setup icon** ⚙ and select **Setup**.

b. In **Quick Find**, enter `Lightning C` and click **Lightning Components.**

c. In the list of Lightning components, go to **cfB2bSampleAppCard**. Click **Del** and then click **OK** to confirm.

NOTE:
If there are multiple b2bSampleAppCard components with the prefix cf, delete all these components.

2. Create a new OmniStudio Card by cloning the existing **b2bSampleAppCard**.

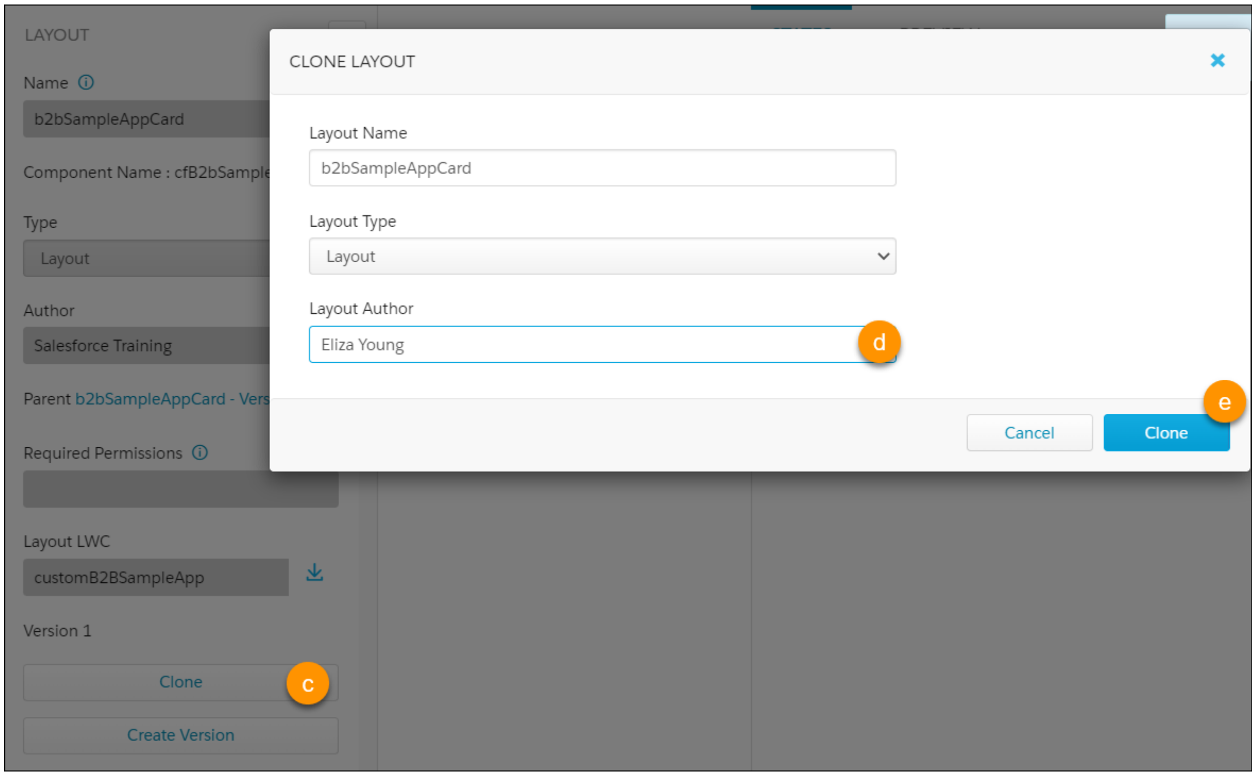


OMNISTUDIO
Layouts & Cards
As of - Viewing as Lea Thomson

LAYOUT NAME	TYPE	LWC ENABLED	AUTHOR	LAST MODIFIED ...	LAST MODIFIED ...	ACTIVE
b2bSampleAppCard	Layout			2/11/22 1:50 PM	Lea Thomson	✓
b2bSampleAppCard (Version 7)	Layout	✓	Vlocity Express	2/11/22 1:50 PM	Lea Thomson	
b2bSampleAppCard (Version 1)	Layout	✓	Salesforce Training	2/15/22 6:30 PM	Lea Thomson	✓
campaign-add-member	Layout			2/11/22 9:44 AM	Lea Thomson	✓

a. In the **App Launcher**, enter `Cards`. There are two results. The first is for Industries CPQ flexcards in the managed package, and the second is the layouts and cards you can amend. Select the second **Vlocity Cards** from the results list.

b. Open the active **b2bSampleAppCard** (in this example, Version 1 - but yours may be Version 7).



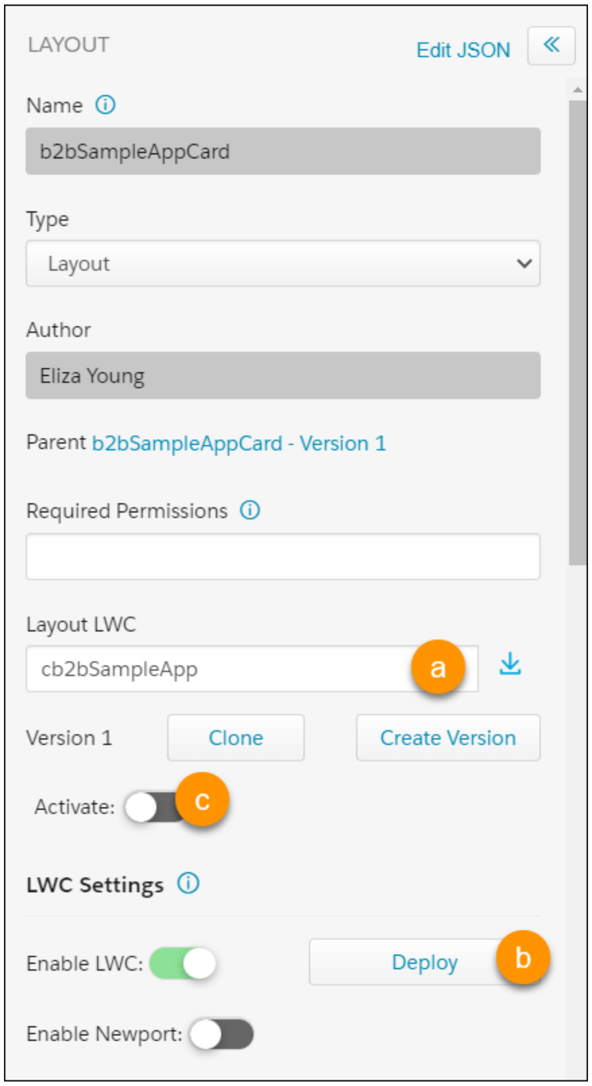
c. Click **Clone** to clone the card.

d. Enter a name in **Layout Author**.

e. Click **Clone** to clone the card. The card is cloned and the new card is opened. You can now update this card.

3. Configure the new card.

The final step is to configure the new card to use your cb2bSampleApp LWC, then deploy and activate it.



Name: b2bSampleAppCard
Type: Layout
Author: Eliza Young
Parent b2bSampleAppCard - Version 1
Required Permissions
Layout LWC: cb2bSampleApp
Version 1
Activate:
LWC Settings
Enable LWC:
Enable Newport:

a. In **Layout LWC**, type and select the name of the custom LWC you just created: `cb2bSampleApp`.

b. Click **Deploy**.

c. Once the card is deployed, click **Activate** to make it the active card.

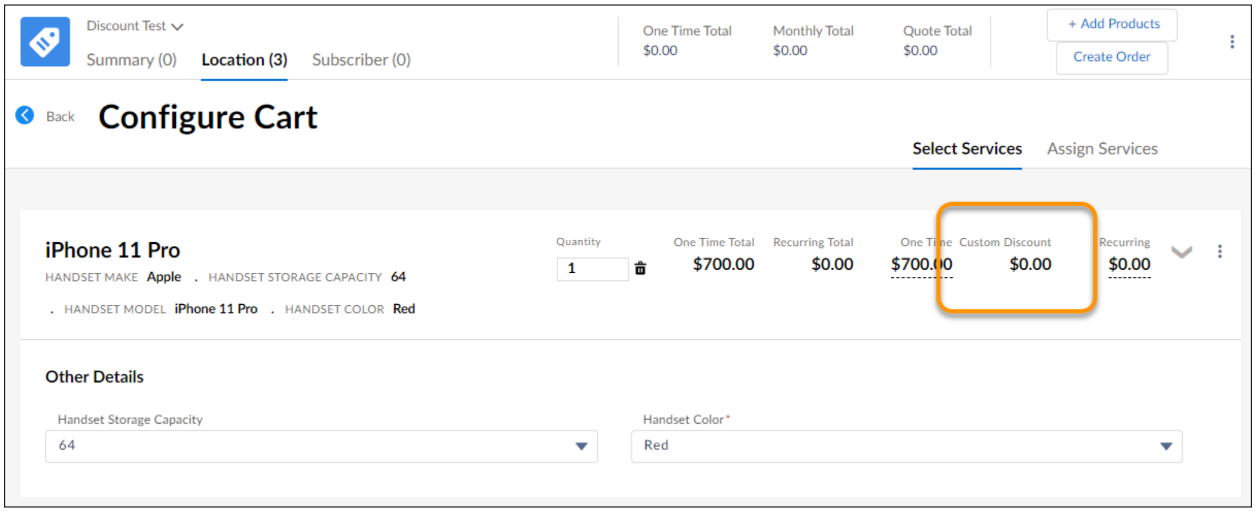
Create a new Enterprise Quote to test your changes. The updated UI now shows the new Apply Discount in the actions list in the Cart.

When you select the Apply Discount option from the actions list, a discount modal is displayed.

Exercise 3: Add a Table Column to the Quote Summary

Scenario

Riding on the waves of success from your previous customization, you've decided to tackle the next customization request: salespeople want to be able to see, for each quote line item, any custom discounts that have been applied. Here's how it should look:



Goal

- Clone and extend an ESM Card
- Create a new session variable.
- Amend the layout of an ESM Card.

Task

1. Clone and extend an ESM Card
2. Add a new session variable to the Card layout

Time: 15 mins

Cards and Session Variables

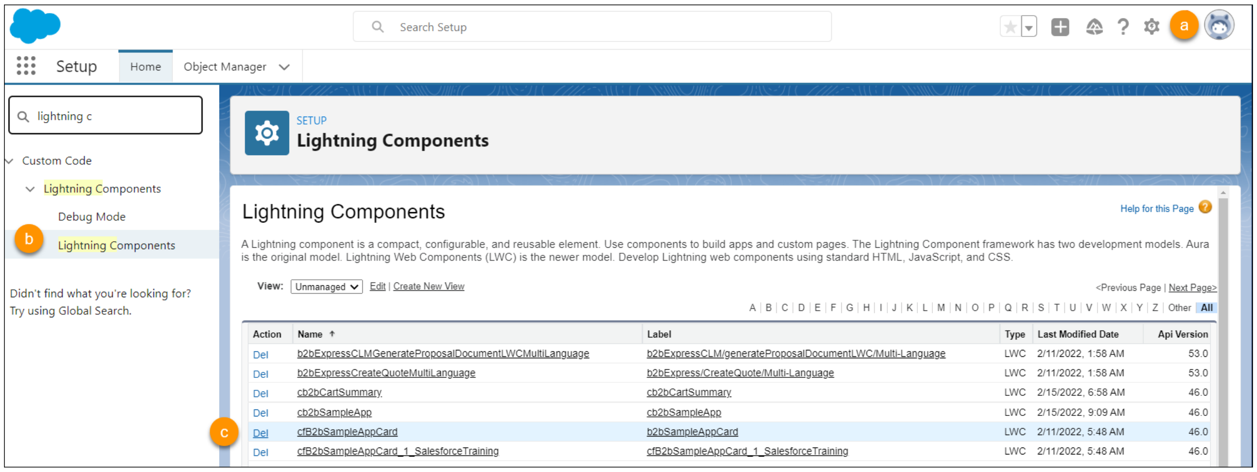
Session variables are used in ESM to pass information selected by the user to ESM for actioning. The sample app card used by ESM already contains session variables, but you're going to add a new one. To do this, you need to know the API name of the variable, and you'll also need to be familiar with editing JSON files.

Task 1: Clone and Extend an ESM Card

The first step in the process is to open and clone the b2bSampleAppCard card, which is used by ESM.

Before you start, to avoid later confusion, delete the temporary file associated with the card.

1. Remove the **cfB2bSampleAppCard**.



Lightning Components

A Lightning component is a compact, configurable, and reusable element. Use components to build apps and custom pages. The Lightning Component framework has two development models. Aura is the original model. Lightning Web Components (LWC) is the newer model. Develop Lightning web components using standard HTML, JavaScript, and CSS.

View: Unmanaged Edit | Create New View <Previous Page | Next Page>

A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | Other | All

Action	Name ↑	Label	Type	Last Modified Date	Api Version
Del	b2bExpressCLMGenerateProposalDocumentLWCMultiLanguage	b2bExpressCLM/generateProposalDocumentLWC/Multi-Language	LWC	2/11/2022, 1:58 AM	53.0
Del	b2bExpressCreateQuoteMultiLanguage	b2bExpress/CreateQuote/Multi-Language	LWC	2/11/2022, 1:58 AM	53.0
Del	cb2bCartSummary	cb2bCartSummary	LWC	2/15/2022, 6:58 AM	46.0
Del	cb2bSampleApp	cb2bSampleApp	LWC	2/15/2022, 9:09 AM	46.0
Del	cfB2bSampleAppCard	b2bSampleAppCard	LWC	2/11/2022, 5:48 AM	46.0
Del	cfB2bSampleAppCard_1_SalesforceTraining	cfB2bSampleAppCard_1_SalesforceTraining	LWC	2/11/2022, 5:48 AM	46.0

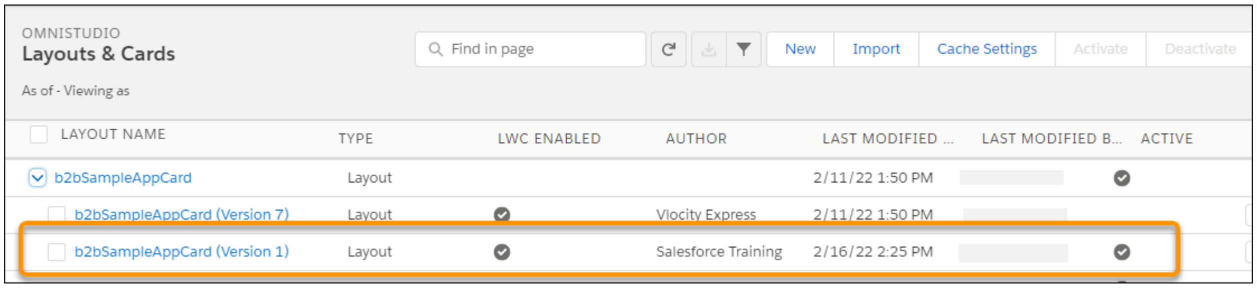
   a. In your playground, click the **Setup icon** ⚙ and select **Setup**.

   b. In **Quick Find**, enter `Lightning C` and click **Lightning Components.**

   c. In the list of Lightning components, go to **cfB2bSampleAppCard**. Click **Del** and then click **OK** to confirm.

NOTE:
If there are multiple components with the prefix cf, delete all these components.

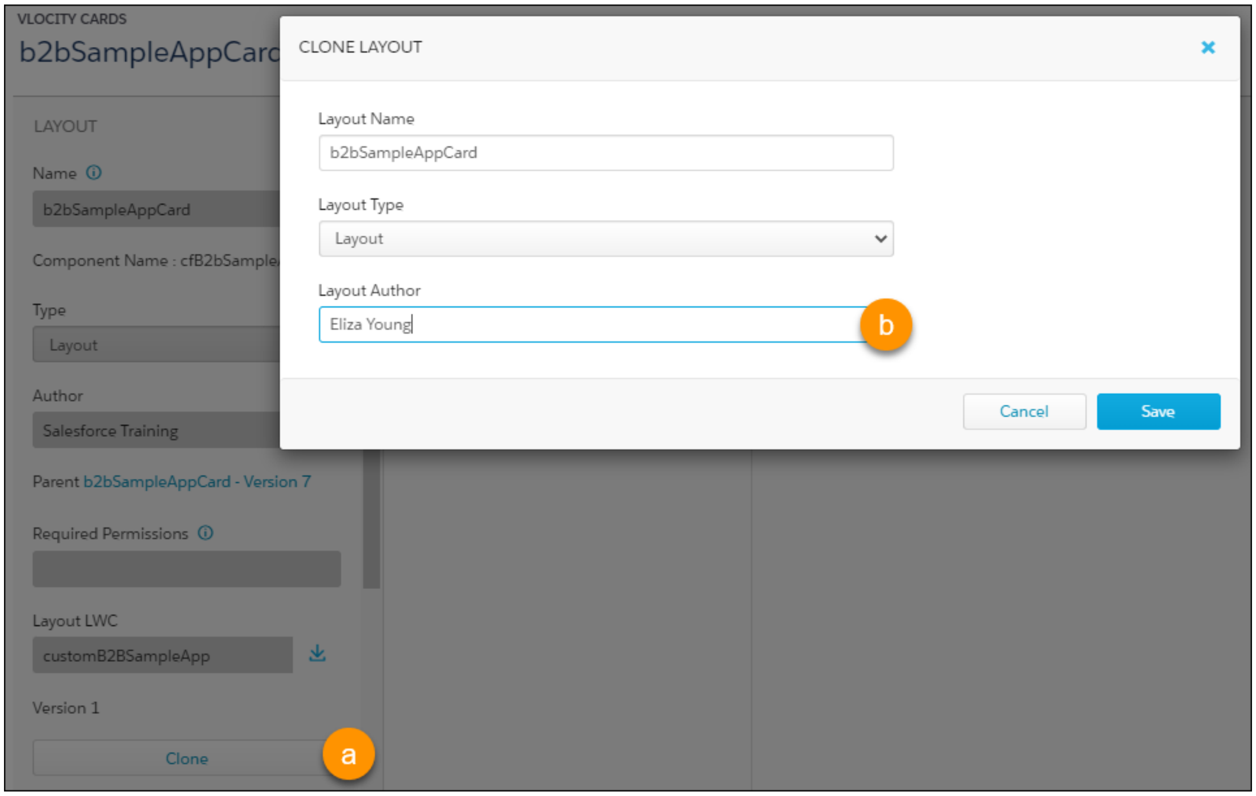
2. Locate and open the active Card.
   a. From the **App Launcher**, type `card` and select the second **Vlocity Cards** from the results list.



LAYOUT NAME	TYPE	LWC ENABLED	AUTHOR	LAST MODIFIED ...	LAST MODIFIED B...	ACTIVE
b2bSampleAppCard	Layout			2/11/22 1:50 PM		✓
b2bSampleAppCard (Version 7)	Layout	✓	Vlocity Express	2/11/22 1:50 PM		
b2bSampleAppCard (Version 1)	Layout	✓	Salesforce Training	2/16/22 2:25 PM		✓

   b. Click the name of the active **b2bSampleAppCard**.

3. Clone the Card.



a. From the navigation pane of the Card, click **Clone** (a).

> **NOTE:**
> If you've already cloned the b2bSampleAppCard, so you have your own custom Card, you'll probably want to create a new version rather than cloning it. Instead of clicking **Clone**, click **Create Version**.

b. Complete the Layout Author details then click **Save** (b) to clone the card.

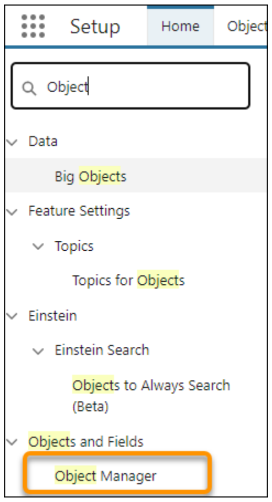
Task 2: Add a New Session Variable to the Card Layout

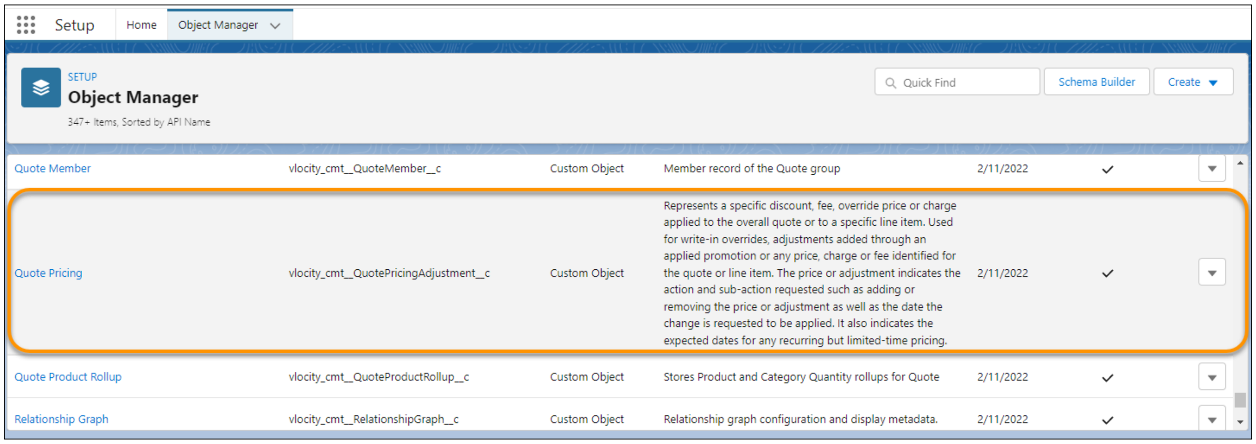
Scroll down the navigation pane to see the session variables. You're going to add a new session variable, but instead of clicking Add Session Variables, you're going to edit the JSON of the Card. But first, you need to work out the name of the variable you're going to use.

1. Locate the API name of the variable to use.

   a. Go to Salesforce's **Setup**. In **Quick Search** type `Object`, and select **Object Manager** from the results list.

   b. Custom objects created specifically for Salesforce Industries Communications, Media and Energy Clouds tend to use the vlocity_cmt namespace - so it's easier to find the variable you're looking for if you sort by API Name. To do this, click **API NAME** in the Object Manager list header, then scroll down to the objects that start with **vlocity_cmt**. You're looking for quote discount information, so the API name will start with **vlocity_cmt__quote**.



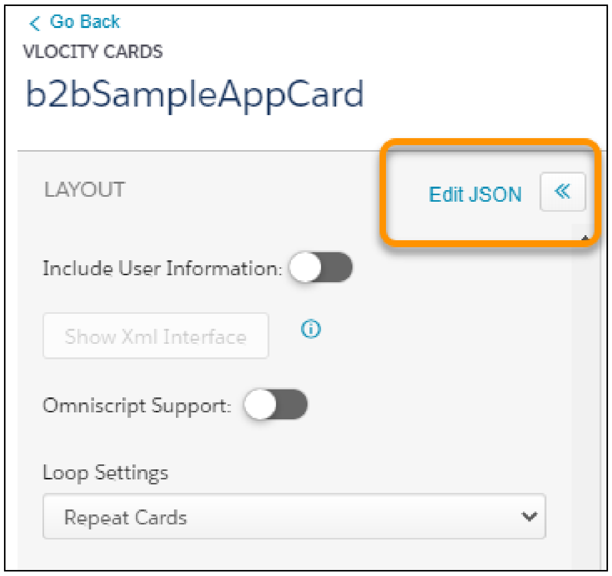


Quote Member	vlocity_cmt__QuoteMember__c	Custom Object	Member record of the Quote group	2/11/2022	✓
Quote Pricing	vlocity_cmt__QuotePricingAdjustment__c	Custom Object	Represents a specific discount, fee, override price or charge applied to the overall quote or to a specific line item. Used for write-in overrides, adjustments added through an applied promotion or any price, charge or fee identified for the quote or line item. The price or adjustment indicates the action and sub-action requested such as adding or removing the price or adjustment as well as the date the change is requested to be applied. It also indicates the expected dates for any recurring but limited-time pricing.	2/11/2022	✓
Quote Product Rollup	vlocity_cmt__QuoteProductRollup__c	Custom Object	Stores Product and Category Quantity rollups for Quote	2/11/2022	✓
Relationship Graph	vlocity_cmt__RelationshipGraph__c	Custom Object	Relationship graph configuration and display metadata.	2/11/2022	✓

   c. The object you need is **vlocity_cmt__QuotePricingAdjustment__c**. Now you've established which API name to use, it's time to return to your Card and add the details.

2. Add the API name and a label to the Card

   a. To reopen the Card you created, click the **App Launcher**. In **Quick Search** type `Cards` and select **Vlocity Cards** from the results list. Open the Card you created (it should have your name as the author, and will not be active).

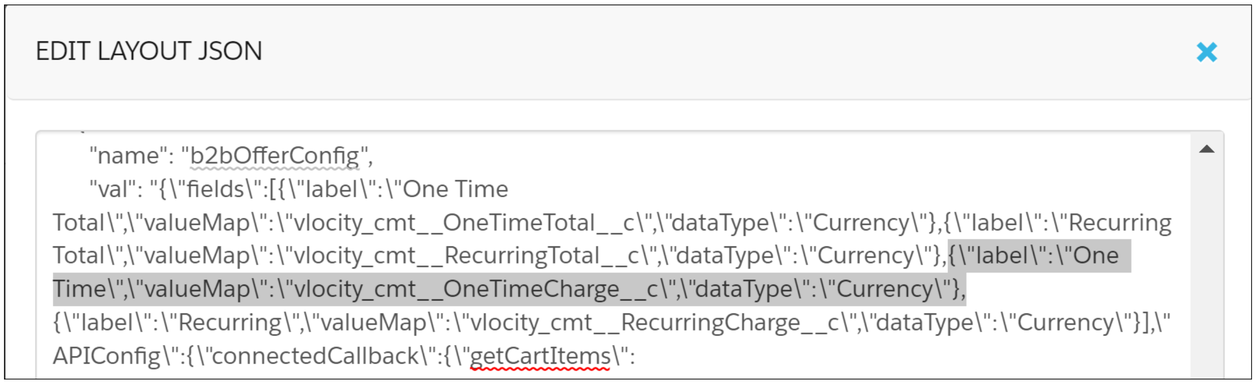


   b. Click **Edit JSON.**

   c. Scroll down to the session variables. In the JSON code, you'll find these after `"sessionVars".`

   Where do you want to add your new variable? You want to add it to the b2bOfferConfig session variable. You'll find this in the JSON as `"name": "b2bOfferConfig"`, If you're confident writing JSON then go ahead and add it! If you're not 100% sure, the easiest way is to copy a field that's already there and amend the details to suit your needs. Copy the field that reads:

```
{\"label\":\"One
Time\",\"valueMap\":\"vlocity_cmt__OneTimeCharge__c\",\"dataTyp
e\":\"Currency\"},
```



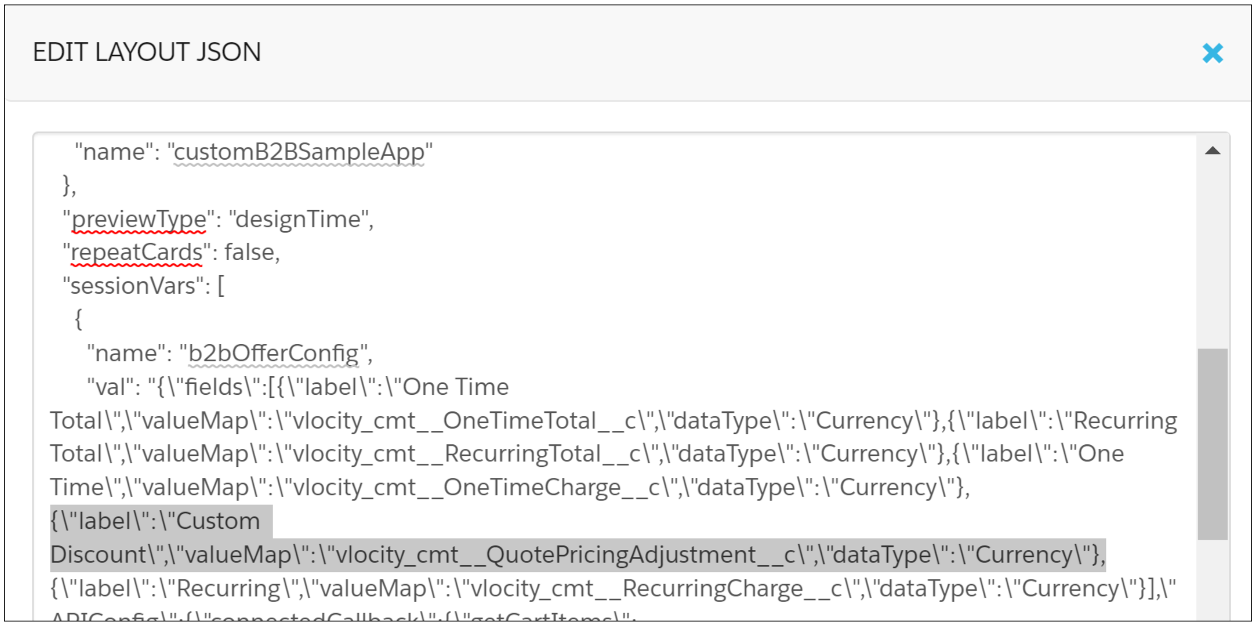
EDIT LAYOUT JSON

```
    "name": "b2bOfferConfig",
    "val": "{\"fields\":[{\"label\":\"One Time
Total\",\"valueMap\":\"vlocity_cmt__OneTimeTotal__c\",\"dataType\":\"Currency\"},{\"label\":\"Recurring
Total\",\"valueMap\":\"vlocity_cmt__RecurringTotal__c\",\"dataType\":\"Currency\"},{\"label\":\"One
Time\",\"valueMap\":\"vlocity_cmt__OneTimeCharge__c\",\"dataType\":\"Currency\"},
{\"label\":\"Recurring\",\"valueMap\":\"vlocity_cmt__RecurringCharge__c\",\"dataType\":\"Currency\"}],\"
APIConfig\":{\"connectedCallback\":{\"getCartItems\":
```

d. Paste it immediately afterwards, then amend the details to:

```
{\"label\":\"Custom
Discount\"",\"valueMap\":\"vlocity_cmt__QuotePricingAdjustment__c\",\"
dataType\":\"Currency\"},
```

Your JSON should now look like this:



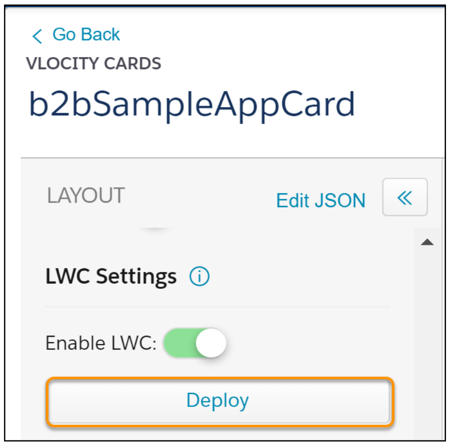
EDIT LAYOUT JSON

```
   "name": "customB2BSampleApp"
  },
  "previewType": "designTime",
  "repeatCards": false,
  "sessionVars": [
   {
    "name": "b2bOfferConfig",
    "val": "{\"fields\":[{\"label\":\"One Time
Total\",\"valueMap\":\"vlocity_cmt__OneTimeTotal__c\",\"dataType\":\"Currency\"},{\"label\":\"Recurring
Total\",\"valueMap\":\"vlocity_cmt__RecurringTotal__c\",\"dataType\":\"Currency\"},{\"label\":\"One
Time\",\"valueMap\":\"vlocity_cmt__OneTimeCharge__c\",\"dataType\":\"Currency\"},
{\"label\":\"Custom
Discount\",\"valueMap\":\"vlocity_cmt__QuotePricingAdjustment__c\",\"dataType\":\"Currency\"},
{\"label\":\"Recurring\",\"valueMap\":\"vlocity_cmt__RecurringCharge__c\",\"dataType\":\"Currency\"}],\"
```

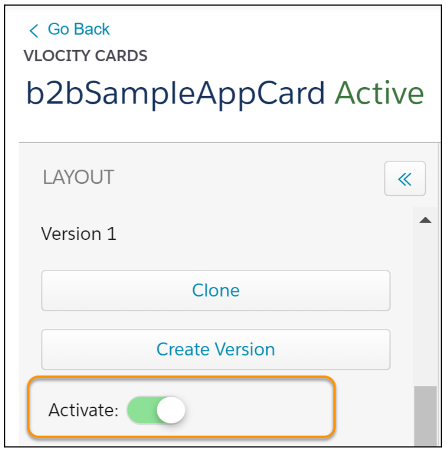
e. Click **Save** to save your changes.

3. Deploy, Activate and Test the Card

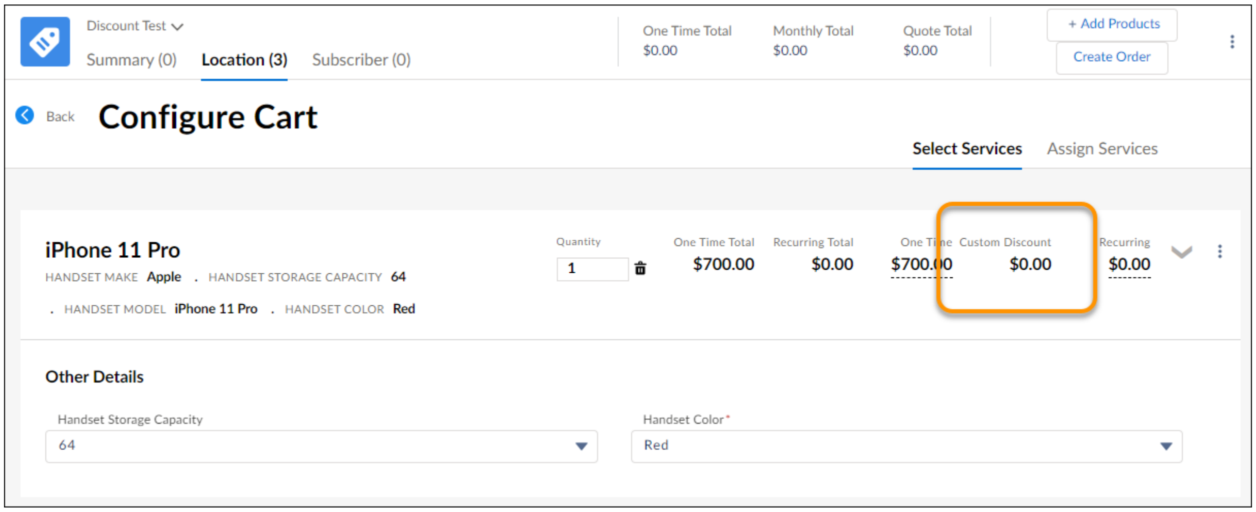
Once you've saved your changes, deploy your card and activate it. Once the activation is complete you can check your changes in ESM.



a. Deploy the card changes. Scroll down the Layout section of the card to the LWC settings and click **Deploy**.



b. Activate the card changes. Scroll up the Layout section of the card and click the **Activate** slider on.
c. Create a new enterprise quote for Acme with the name Discount Test. Create a new subscriber, or upload subscribers using a csv file. Select a subscriber then add a product.


Discount Test
Summary (0) Location (3) Subscriber (0)

One Time Total	Monthly Total	Quote Total
$0.00	$0.00	$0.00

Configure Cart

iPhone 11 Pro
HANDSET MAKE Apple . HANDSET STORAGE CAPACITY 64
. HANDSET MODEL iPhone 11 Pro . HANDSET COLOR Red

Quantity	One Time Total	Recurring Total	One Time	Custom Discount	Recurring
1	$700.00	$0.00	$700.00	$0.00	$0.00

Other Details

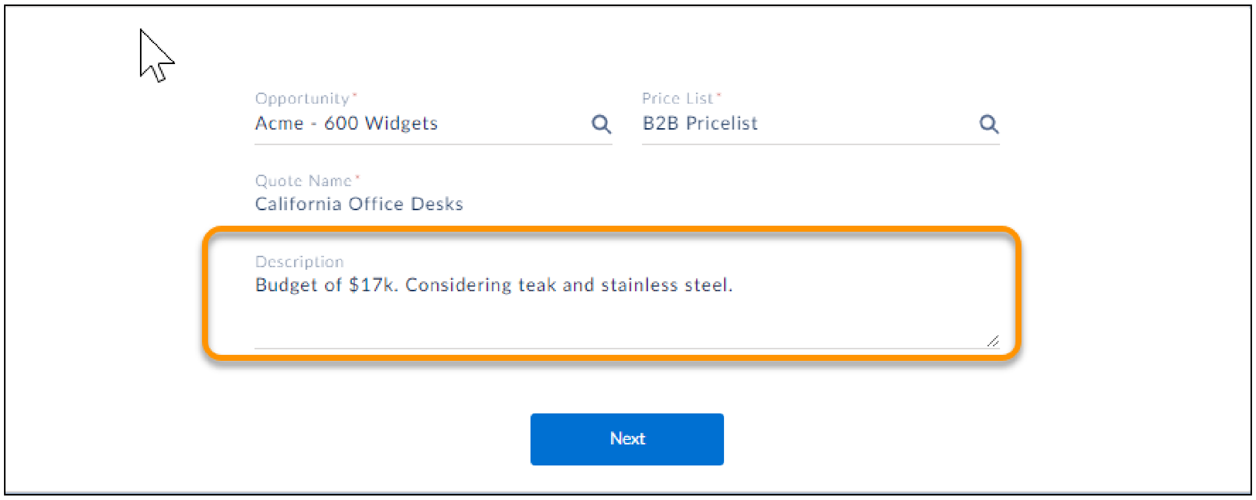
Handset Storage Capacity: 64
Handset Color*: Red

d. Notice the Custom Discount appears in the Cart.

Exercise 4: Configure and Test the CreateQuote OmniScript

Scenario

The change requests just keep on coming! This time, you've been asked to add a "Description" field into the quote process. This will allow the sales team to add details such as expected budget when they first create the quote. An example of the expected design is shown here:



Opportunity*	Price List*
Acme - 600 Widgets	B2B Pricelist
Quote Name*	
California Office Desks	
Description	
Budget of $17k. Considering teak and stainless steel.	

Goals

- Identify fields to add to the quote process.
- Extend and configure the CreateQuote OmniScript.
- Test the CreateQuote OmniScript amendments.

Tasks

1. Identify the required data.
2. Create a new version of the CreateQuote OmniScript.
3. Add a new Text Area input field to Step1.
4. Test and deploy the changes.

Time: 20 mins

OmniScript Configuration

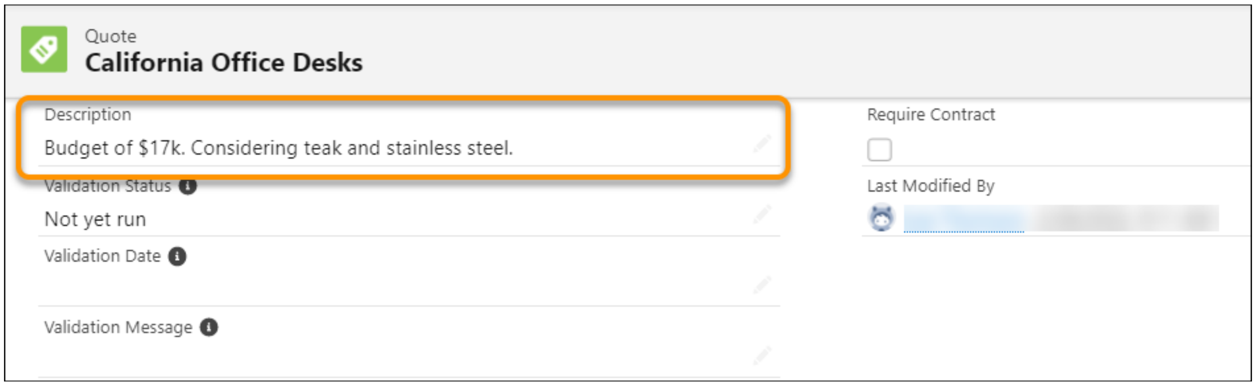
OmniScripts are an important tool in the ESM Developer's toolkit. They're used to guide your end users through sales and service processes. They are context-sensitive, and can provide personalized responses based on input, and integration to enterprise applications and data.

This course isn't designed to show you how to customize OmniScripts. Instead, it gives you the opportunity to make a small, simple change to one of the key ESM OmniScripts: CreateQuote. In later modules, you'll incorporate the added information into DataRaptors and Integration Procedures.

This exercise helps you gain a better understanding of the structure of an important ESM OmniScript, including its input and output constructs. To learn more, we recommend you complete the Build Guided Experiences with OmniStudio trail in Trailhead.

Task 1: Identify the Required Data

In this example you want the user to enter data in your OmniScript that will be passed into the description on your quote record, as shown here.

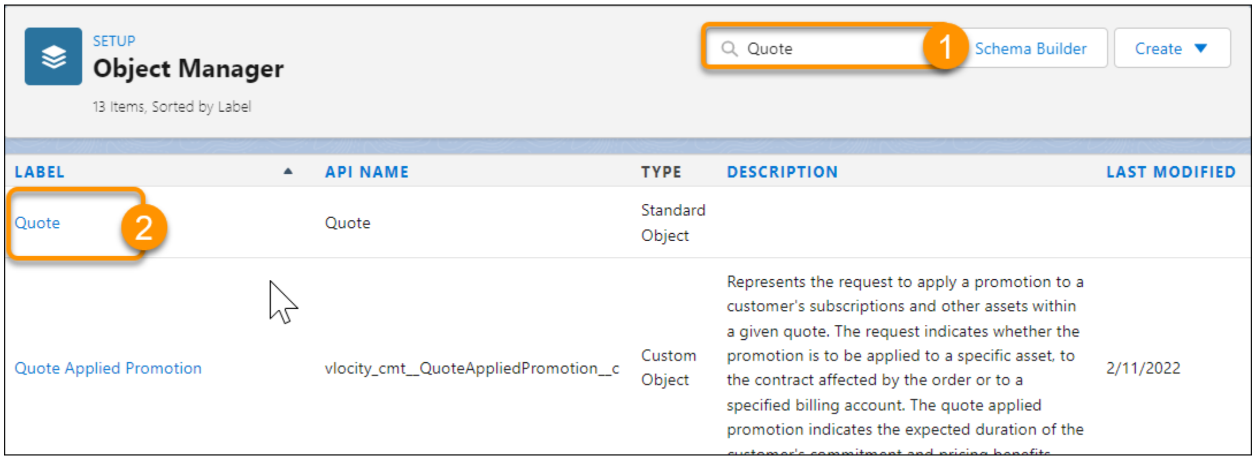


Quote
California Office Desks

Description: Budget of $17k. Considering teak and stainless steel.
Validation Status: Not yet run
Validation Date
Validation Message
Require Contract
Last Modified By

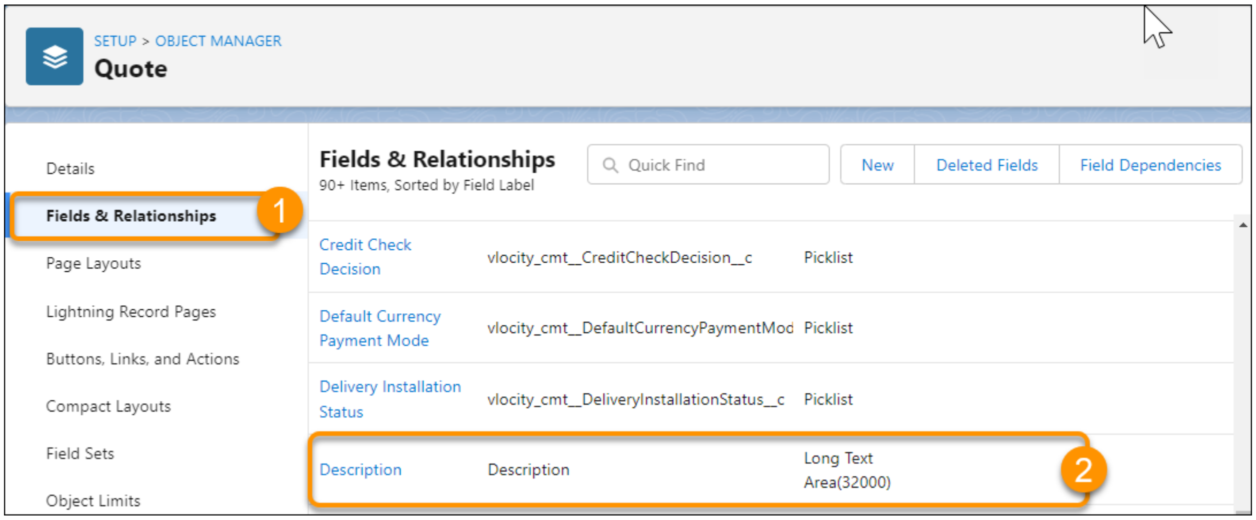
But how do you find the appropriate Salesforce field to pass the data into? You guessed it - the Object Manager. In your ESM training playground, open the Quote object from the Object Manager:

1. Go to Salesforce's **Setup** then in **Quick Find** type `Object Manager`. Select **Object Manager** from the results list.



LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED
Quote	Quote	Standard Object		
Quote Applied Promotion	vlocity_cmt__QuoteAppliedPromotion__c	Custom Object	Represents the request to apply a promotion to a customer's subscriptions and other assets within a given quote. The request indicates whether the promotion is to be applied to a specific asset, to the contract affected by the order or to a specified billing account. The quote applied promotion indicates the expected duration of the	2/11/2022

2. In the **Object Manager**, in **Quick Find** (1) type `Quote`. You'll see the standard objects and the custom objects associated with the quote.
3. Open the **Quote** standard object by clicking on the label (2).


SETUP > OBJECT MANAGER
Quote

Credit Check Decision	vlocity_cmt__CreditCheckDecision__c	Picklist
Default Currency Payment Mode	vlocity_cmt__DefaultCurrencyPaymentMod	Picklist
Delivery Installation Status	vlocity_cmt__DeliveryInstallationStatus__c	Picklist
Description	Description	Long Text Area(32000)

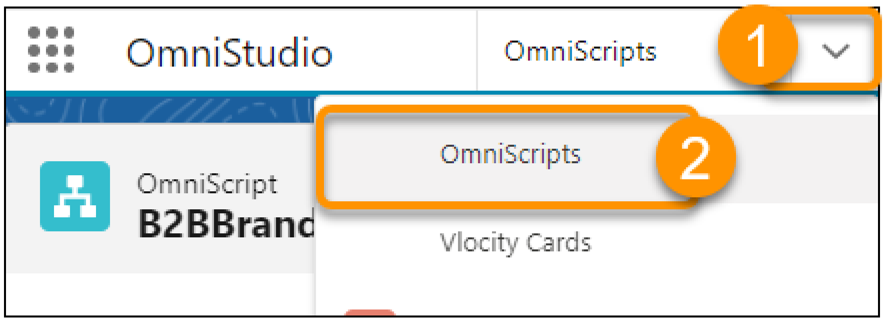
4. Click **Fields & Relationships** in the Quote object navigation pane (1), then locate the **Description** field (2). Note the information displayed, as you'll need this to set up a corresponding field in your OmniScript:

Field	Data
Field Label	Description
Field Name	Description
Data Type	Long Text Area(32000)

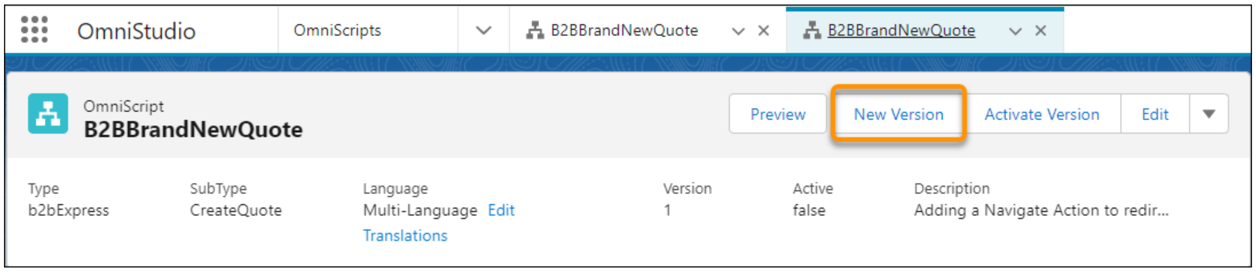
Task 2: Create a new version of the CreateQuote OmniScript

Now that you know the information you need to add to your OmniScript, it's time to create a new version of your OmniScript.

1. In your ESM training playground, click the **App Launcher** then in **Quick Find** type `OmniStudio`. Select **OmniStudio** from the results list.



2. In OmniStudio, click the **tab menu** (1) and select **OmniScripts** (2).
3. Locate the b2bExpress/CreateQuote OmniScript and open the active version.



Type	SubType	Language	Version	Active	Description
b2bExpress	CreateQuote	Multi-Language Edit Translations	1	false	Adding a Navigate Action to redir...

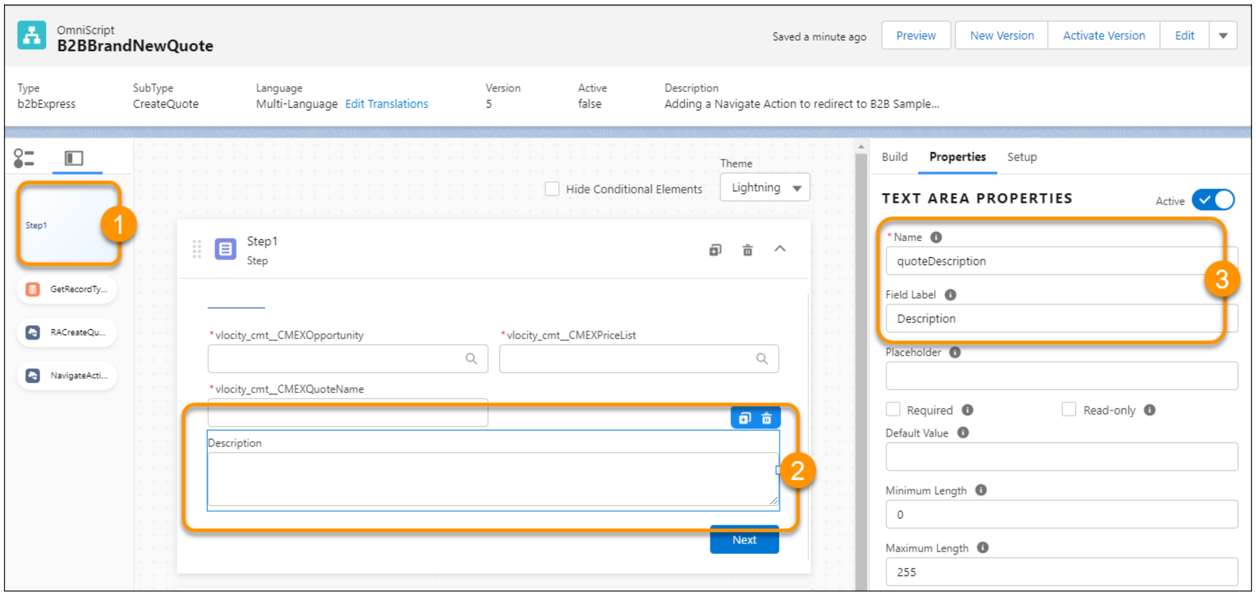
4. Click **New Version** to create a new version of the CreateQuote OmniScript.

NOTE:
ALWAYS create a new version of your OmniScript when you make an amendment. That way, if you make a mistake, it is easier to roll back.

Task 3: Add a New Text Area Field to Step1

Check you have your new version of the CreateQuote OmniScript open before editing it.

1. Open **Step1** (1).
2. In the **Build** tab, search for `Text` inputs to add to Step1.
3. Your input type needs to match the data type of the field into which you'll be passing the data. Refer to the end of Task 1 and note that the Description field in the Quote object is of type Long Text Area. Drag a **Text Area** input element onto Step1 of your OmniScript (2).



Type: b2bExpress
SubType: CreateQuote
Language: Multi-Language Edit Translations
Version: 5
Active: false
Description: Adding a Navigate Action to redirect to B2B Sample...

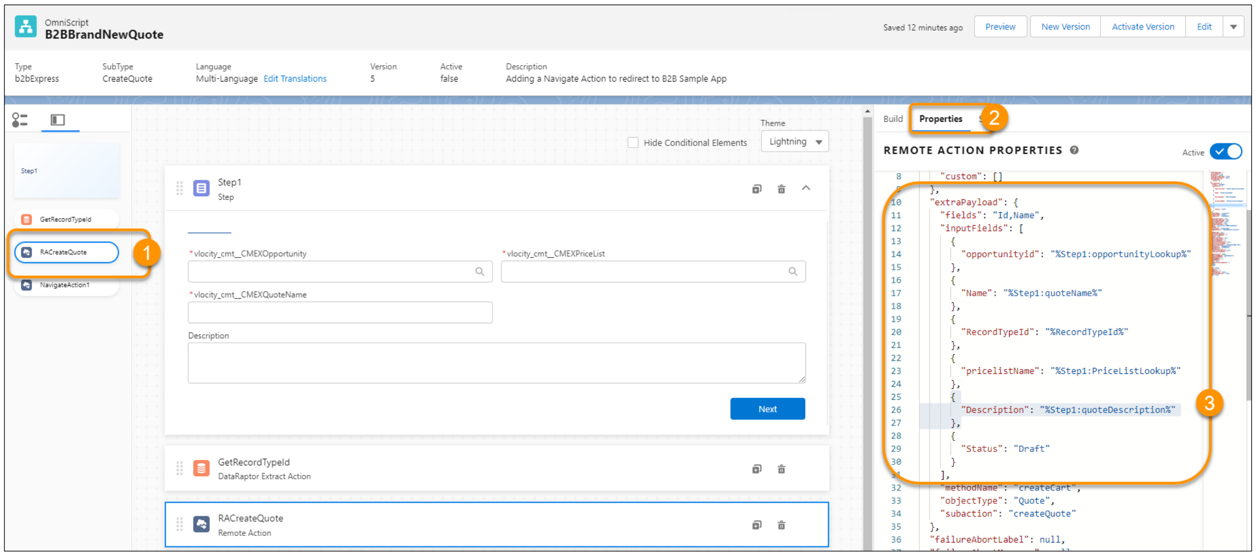
*vlocity_cmt__CMEXOpportunity
*vlocity_cmt__CMEXPriceList
*vlocity_cmt__CMEXQuoteName
Description

*Name: quoteDescription
Field Label: Description
Placeholder
Required
Read-only
Default Value:
Minimum Length: 0
Maximum Length: 255

4. Click on your new input field and in the **Properties** tab amend its properties (3):

Field	**Value**
Name	`quoteDescription`
Field Label	`Description`

5. The final step is to amend the CreateQuote remote action in the OmniScript to pass the information from your new Description field into the new Salesforce quote.
   a. Click the remote action **RACreateQuote** (1).



```
    "custom": []
  },
  "extraPayload": {
    "fields": "Id,Name",
    "inputFields": [
      {
        "opportunityid": "%Step1:opportunityLookup%"
      },
      {
        "Name": "%Step1:quoteName%"
      },
      {
        "RecordTypeId": "%RecordTypeId%"
      },
      {
        "pricelistName": "%Step1:PriceListLookup%"
      },
      {
        "Description": "%Step1:quoteDescription%"
      },
      {
        "Status": "Draft"
      }
    ],
    "methodName": "createCart",
    "objectType": "Quote",
    "subaction": "createQuote"
  },
  "failureAbortLabel": null,
```

b. In the **Properties** tab of RACreateQuote (2), scroll to the bottom and select **Edit Properties as JSON**.

c. To add the quoteDescription from Step1 into your remote action, scroll down to the extraPayload array, and add this code:

```
{
  "Description": "%Step1:quoteDescription%"
 },
```

NOTE:

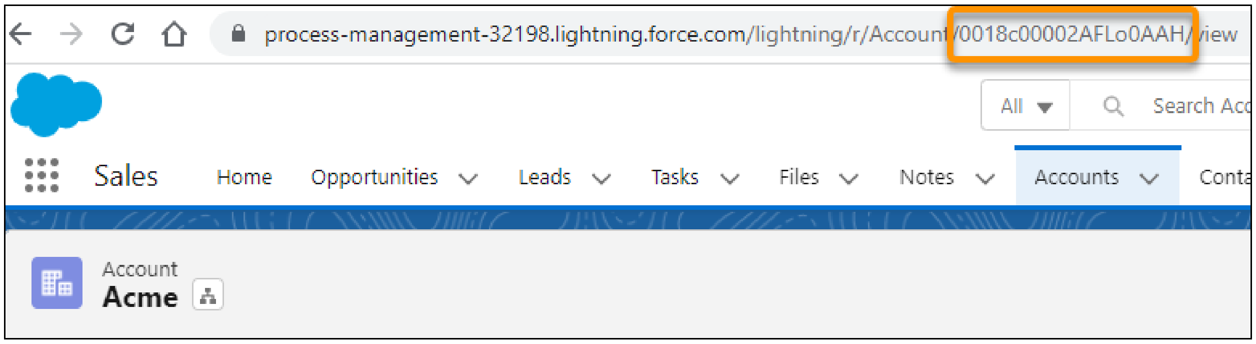
1. “Description” identifies the name of the Salesforce Quote field you identified in the Object Manager in Task 1 of this exercise.

2. “%Step1:quoteDescription%” references the name of the field you added to this OmniScript at the beginning of this task.

Task 4: Test and deploy the changes

As you learned in the lesson, the CreateQuote OmniScript uses the Account ID as the Context ID when creating the quote. Therefore, to test the OmniScript you'll need an account ID.

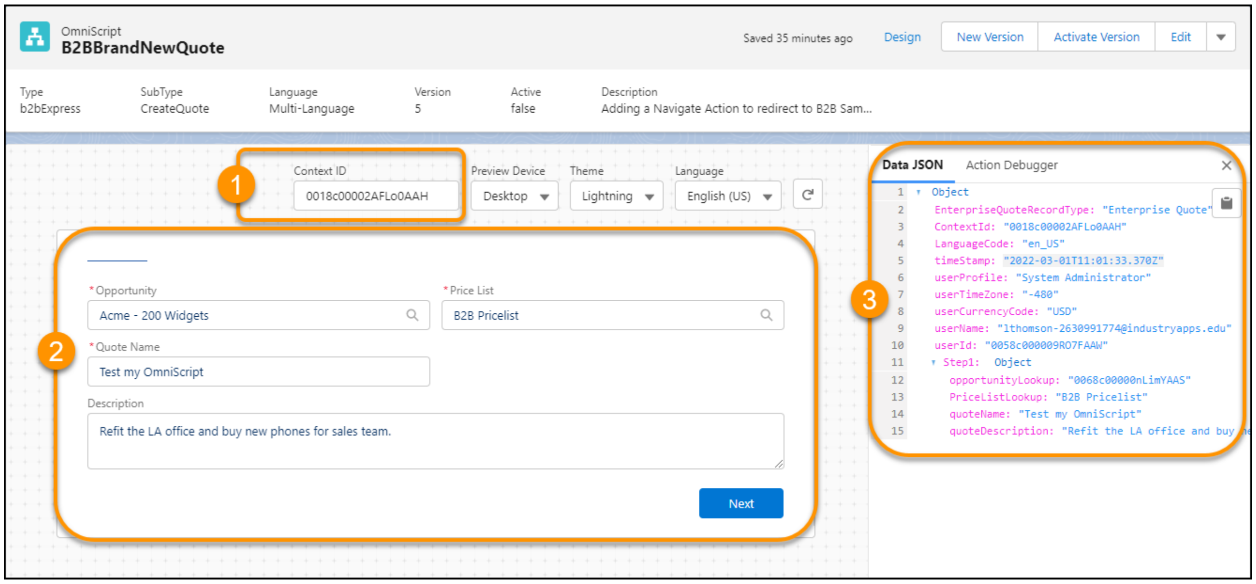
1. In a new tab, open the **Acme** account and copy the unique identifier from the URL.



process-management-32198.lightning.force.com/lightning/r/Account/0018c00002AFLo0AAH/view

Account
Acme

2. Click **Preview** on the OmniScript (in the top right of the window) and paste the account unique identifier into the **Context ID** field (1).



Type	SubType	Language	Version	Active	Description
b2bExpress	CreateQuote	Multi-Language	5	false	Adding a Navigate Action to redirect to B2B Sam...

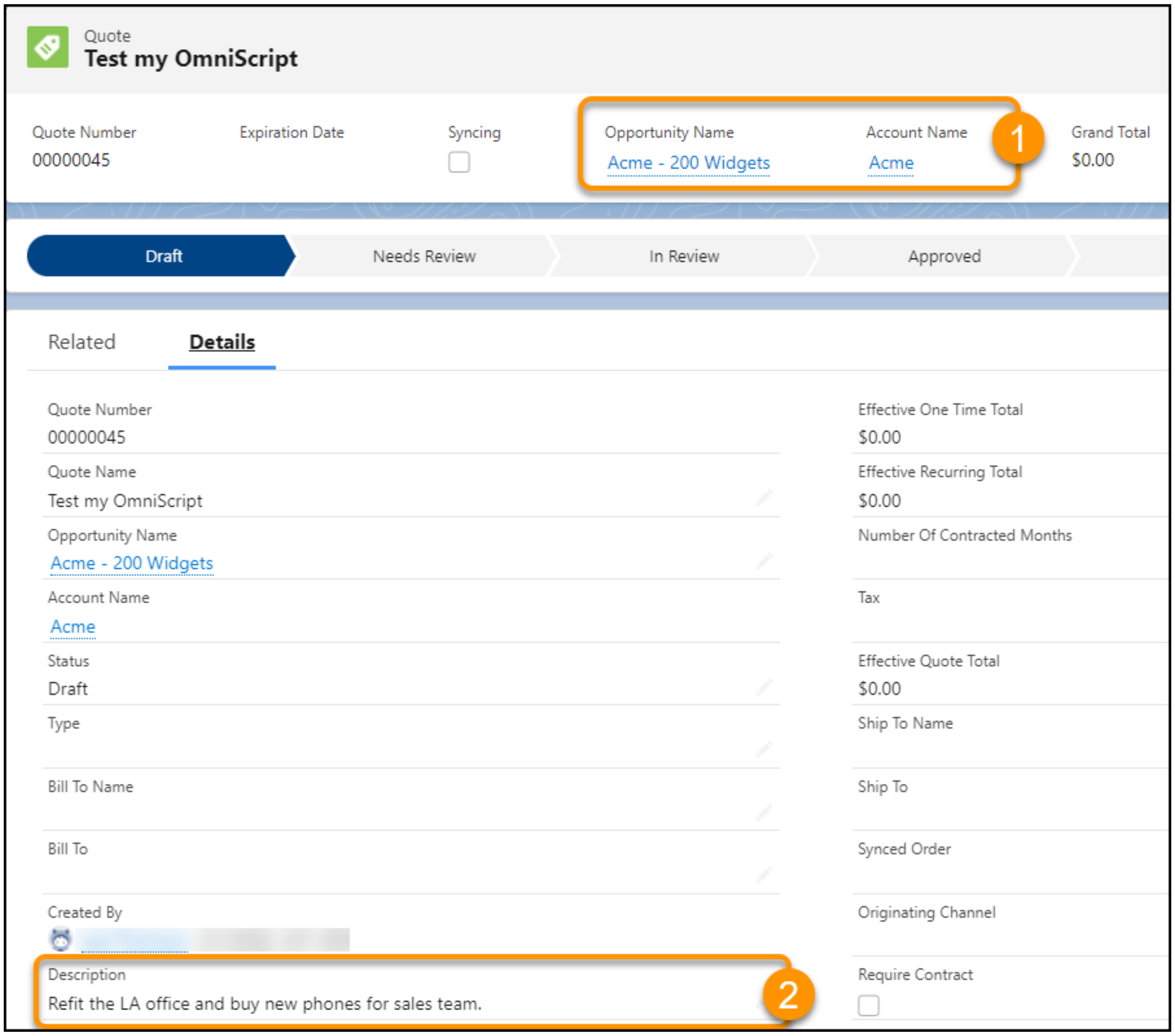
Context ID: 0018c00002AFLo0AAH
Preview Device: Desktop
Theme: Lightning
Language: English (US)

*Opportunity: Acme - 200 Widgets
*Price List: B2B Pricelist
*Quote Name: Test my OmniScript
Description: Refit the LA office and buy new phones for sales team.

```
▾ Object
   EnterpriseQuoteRecordType: "Enterprise Quote"
   ContextId: "0018c00002AFLo0AAH"
   LanguageCode: "en_US"
   timeStamp: "2022-03-01T11:01:33.370Z"
   userProfile: "System Administrator"
   userTimeZone: "-480"
   userCurrencyCode: "USD"
   userName: "lthomson-2630991774@industryapps.edu"
   userId: "0058c000009RO7FAAW"
  ▾ Step1:  Object
     opportunityLookup: "0068c00000nLimYAAS"
     PriceListLookup: "B2B Pricelist"
     quoteName: "Test my OmniScript"
     quoteDescription: "Refit the LA office and buy
```

3. Complete the details of the new quote (2). Notice your opportunity selection relates specifically to the Acme account. As you add the data, the Data JSON tab (3) displays the fields and the data they contain.
4. Click **Next** to create the quote.

5. Switch to the Acme account in Salesforce and locate the new quote (1).



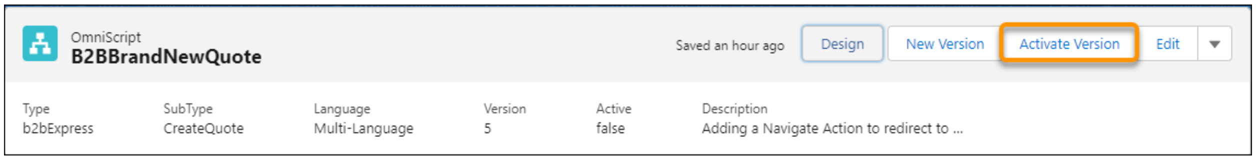
Quote
Test my OmniScript

Quote Number	Expiration Date	Syncing	Opportunity Name	Account Name	Grand Total
00000045		☐	Acme - 200 Widgets	Acme	$0.00

Quote Number: 00000045	Effective One Time Total: $0.00
Quote Name: Test my OmniScript	Effective Recurring Total: $0.00
Opportunity Name: Acme - 200 Widgets	Number Of Contracted Months:
Account Name: Acme	Tax:
Status: Draft	Effective Quote Total: $0.00
Type:	Ship To Name:
Bill To Name:	Ship To:
Bill To:	Synced Order:
Created By:	Originating Channel:
Description: Refit the LA office and buy new phones for sales team.	Require Contract: ☐

Notice the details you entered in the OmniScript have been added to the quote, including the description (2).

6. Now you know your OmniScript works, switch back to the OmniScript tab and click **Activate Version** to deploy and activate your new version.



Type	SubType	Language	Version	Active	Description
b2bExpress	CreateQuote	Multi-Language	5	false	Adding a Navigate Action to redirect to ...

7. Once the compilation and deployment is complete, click **Done**. Your OmniScript is activated, and all new enterprise quotes will include the Description field in the creation process.

Exercise 5: Create a DataRaptor

Scenario

You've been asked to add some custom fields to your quotes and orders. As a first step, you want to update the master order when it's created so it shows the quote description. If you get this right, you know it's the same process for any custom field, so the rest should be easy!

First, you need to do some investigative work to locate the appropriate fields, then create a new DataRaptor to extract the description from the active quote.

Goals

- Locate objects and fields required for a specific DataRaptor
- Create a DataRaptor for a predefined purpose.
- Test the DataRaptor.

Tasks

1. Locate the required fields for the DataRaptor.
2. Create the DataRaptor.
3. Test the DataRaptor.

Time: 10 mins

ESM DataRaptors

DataRaptors are OmniStudio tools used by ESM to read, transform, and load data. For example, an OmniScript will often call DataRaptors to:

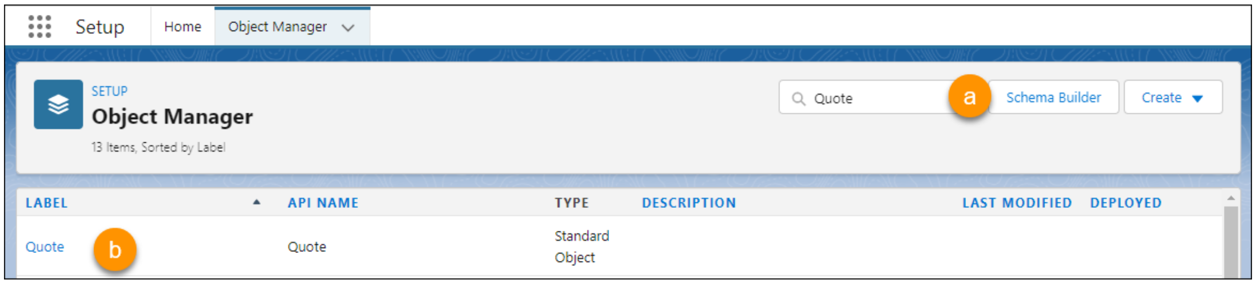
1. extract information from the database so it can be viewed by the user.
2. perform transformations on the data.
3. loading the updated data back into the database.

DataRaptors tend to be used for single-operation purposes. However they can be called in combination by integration procedures to create powerful multi-operation processes.
In this exercise, you'll create a simple DataRaptor that will later be used in Exercise 6 to extend the CreateOrders integration procedure.

Task 1: Locate the required fields for the DataRaptor

In this exercise, you're creating a DataRaptor to extract the quote description from the current quote. You'll need to check in the Object Manager to locate the appropriate field.

1. In the Object Manager, locate and open the Quote object.



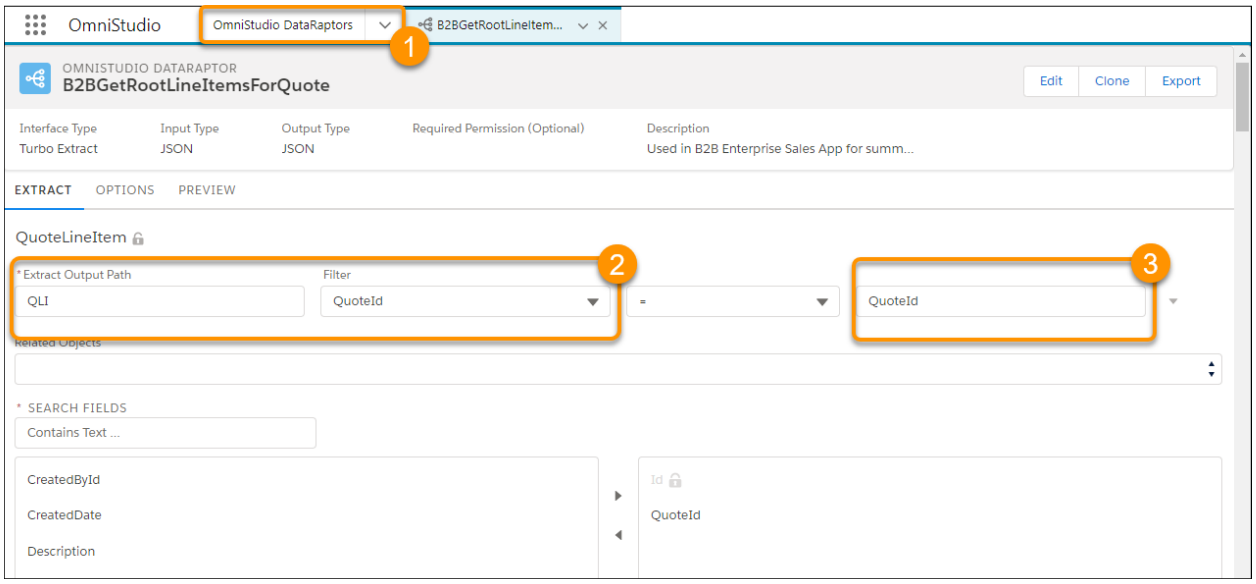
LABEL	API NAME	TYPE	DESCRIPTION	LAST MODIFIED	DEPLOYED
Quote	Quote	Standard Object			

   a. Go to Salesforce's **Setup** and use **Quick Find** to search for the `Object Manager`. Open the **Object Manager**.
   b. Use the Object Manager **Quick Find** (a) to search for the `Quote` object, then click the **Quote** label (b) to open the Quote object.

2. Click **Fields & Relationships** in the navigation pane on the left. You'll see there are two Quote totals: the Grand Total, which is the total that appears on the standard Salesforce Quote UI, and the Effective Quote Total, which appears in the Enterprise Quote Summary. It's the **Effective Quote Total** you need, so make a note of the details here:

Details for the Effective Quote Total Field
Field Label:
Field Name:
API Name:
Data Type:

You've identified the details of the quote description,but how do you identify the active quote? Let's check how other DataRaptors do it.



OMNISTUDIO DATARAPTOR
B2BGetRootLineItemsForQuote

Interface Type	Input Type	Output Type	Required Permission (Optional)	Description
Turbo Extract	JSON	JSON		Used in B2B Enterprise Sales App for summ...

QuoteLineItem

* Extract Output Path	Filter		
QLI	QuoteId	=	QuoteId

* SEARCH FIELDS
Contains Text ...

CreatedById
CreatedDate
Description

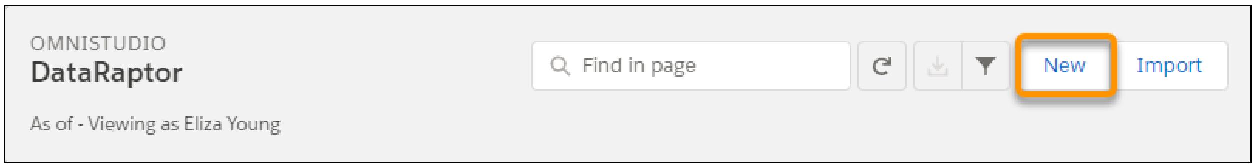
Id
QuoteId

3. Click the **App Launcher** then search for and select **OmniStudio**.
4. From the **OmniStudio Navigation** menu (1), choose **OmniStudio DataRaptors** to see a list of DataRaptors available in ESM.
5. Open **B2BGetRootLineItemsForQuote**. This is an ESM Turbo Extract DataRaptor which extracts information about quote lines items to display in the Quote Summary page.
6. How does this DataRaptor identify which quote line items to extract? It filters where the Quote Line Item's Quote Id (QLI.QuoteId) (2) is equal to QuoteId (3). So, the active quote is identified by a field called QuoteId, which is the ContextID for the DataRaptor Extract.

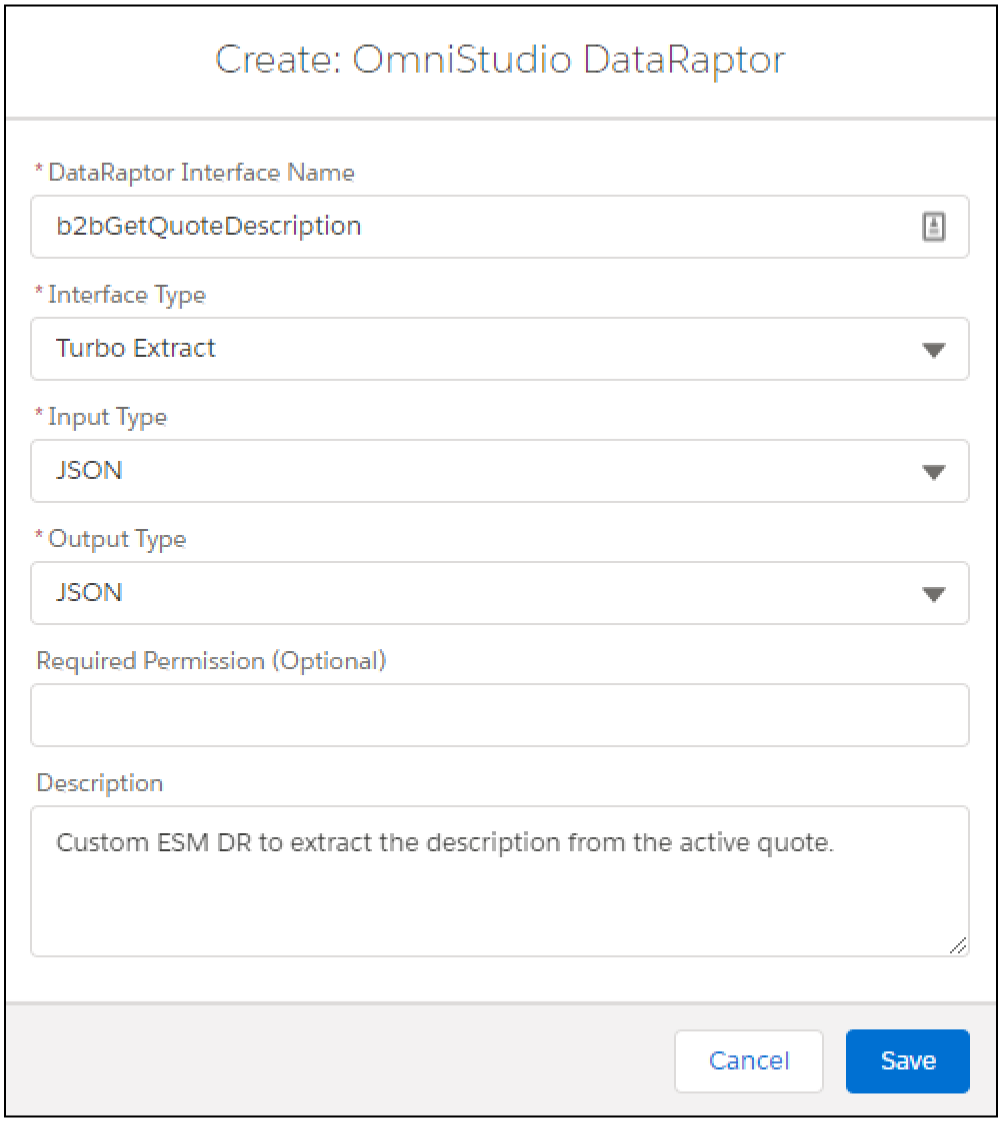
So, you'll need to use the QuoteId from the active quote object to filter the quote records and locate the Description data for your quote.

Task 2: Create your DataRaptor

You've collected all the information you need for your DataRaptor, so it's time to create it.



1. In the OmniStudio DataRaptor workspace header, click **New** then complete the details of your new DataRaptor.


Create: OmniStudio DataRaptor

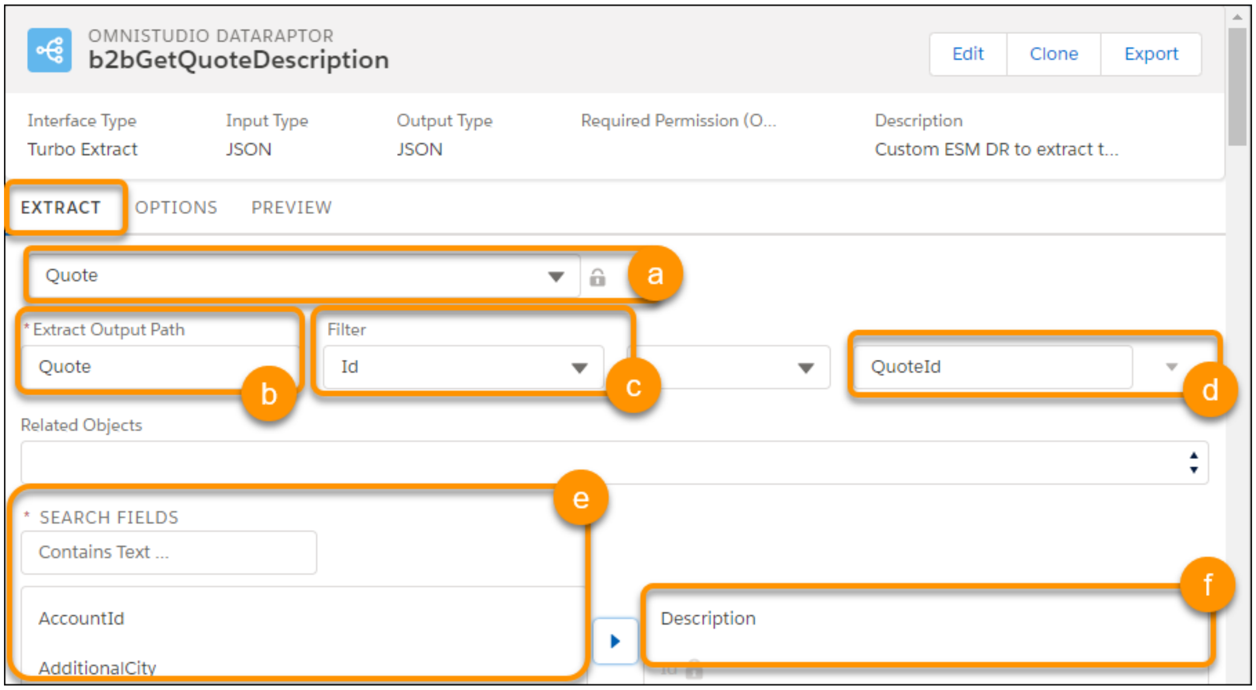
* DataRaptor Interface Name: b2bGetQuoteDescription
* Interface Type: Turbo Extract
* Input Type: JSON
* Output Type: JSON

Required Permission (Optional):

Description: Custom ESM DR to extract the description from the active quote.

Name	Content
DataRaptor Interface Name:	`b2bGetQuoteDescription`
Interface Type:	Turbo Extract
Input Type:	JSON
Output Type:	JSON
Required Permission:	[leave blank]
Description:	`Custom ESM DR to extract the description from the active quote.`

2. Once you're done, click **Save**.
3. Now you need to complete the details for the DataRaptor. Check you're on the Extract tab of the DataRaptor workspace.


OMNISTUDIO DATARAPTOR
b2bGetQuoteDescription

Interface Type	Input Type	Output Type	Required Permission (O...	Description
Turbo Extract	JSON	JSON		Custom ESM DR to extract t...

Quote

Extract Output Path: Quote

Filter: Id — QuoteId

Related Objects

SEARCH FIELDS: Contains Text ...

AccountId
AdditionalCity

Description

a. Select the Salesforce Object you'll be extracting the data from.
b. Identify the output path.

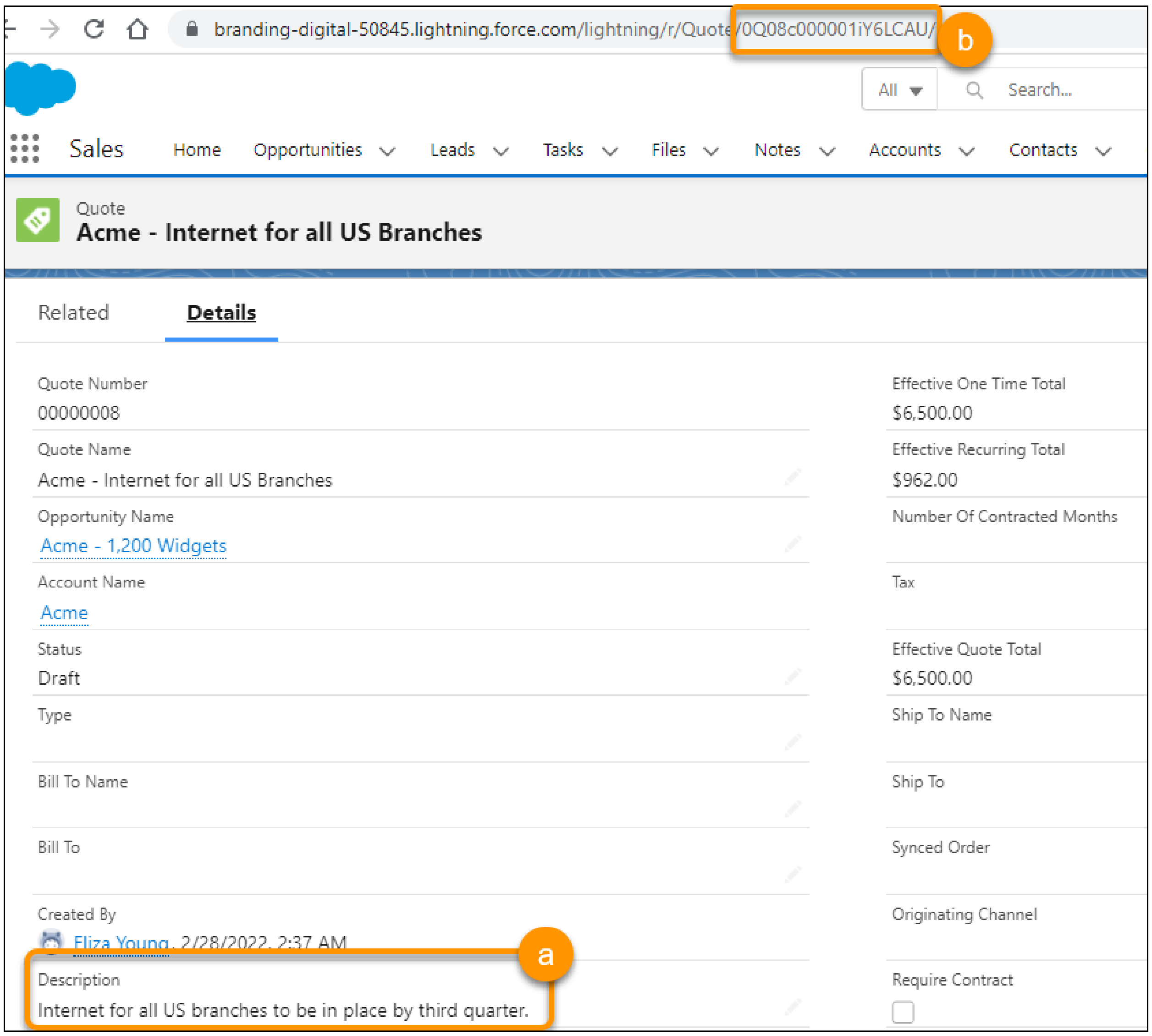
c. What field will you be checking within each record as the filter?
d. What will you be comparing against in the filter?
e. What information will you extract? Add fields from the search list to the active list by clicking ▸.

Name	Content
Object:	Quote
Extract Output Path:	Quote
Filter:	Id = `QuoteId`
Fields:	Description

Task 3: Test the DataRaptor

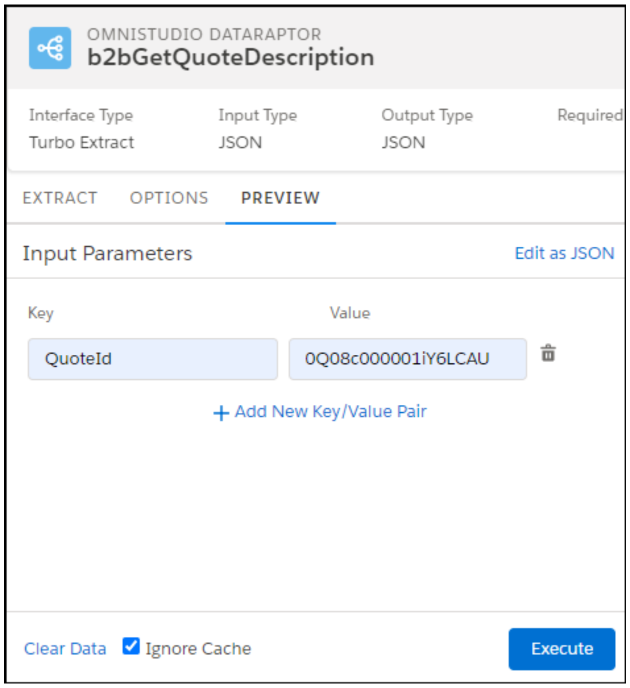
To test your DataRaptor works, you'll need the QuoteId for an existing quote that has a completed description.

1. Open Salesforce in a new window, and from the **App Launcher** find and click **Sales**.
2. Open a customer quote and, in the **Details** tab, check it has a completed **description** (a).


branding-digital-50845.lightning.force.com/lightning/r/Quote/0Q08c000001iY6LCAU/
b
All
Search...
Sales
Home
Opportunities
Leads
Tasks
Files
Notes
Accounts
Contacts
Quote
Acme - Internet for all US Branches
Related
Details
a

Quote Number: 00000008	Effective One Time Total: $6,500.00
Quote Name: Acme - Internet for all US Branches	Effective Recurring Total: $962.00
Opportunity Name: Acme - 1,200 Widgets	Number Of Contracted Months:
Account Name: Acme	Tax:
Status: Draft	Effective Quote Total: $6,500.00
Type:	Ship To Name:
Bill To Name:	Ship To:
Bill To:	Synced Order:
Created By: Eliza Young, 2/28/2022, 2:37 AM	Originating Channel:
Description: Internet for all US branches to be in place by third quarter.	Require Contract: ☐

3. Copy the 18-character unique quote identifier from the URL (b). You'll use this to test your DataRaptor.
4. Return to the DataRaptor you've just created, and click on the **PREVIEW** tab.
5. In the Input Parameters pane, click **+Add New Key/Value Pair** and add these details:

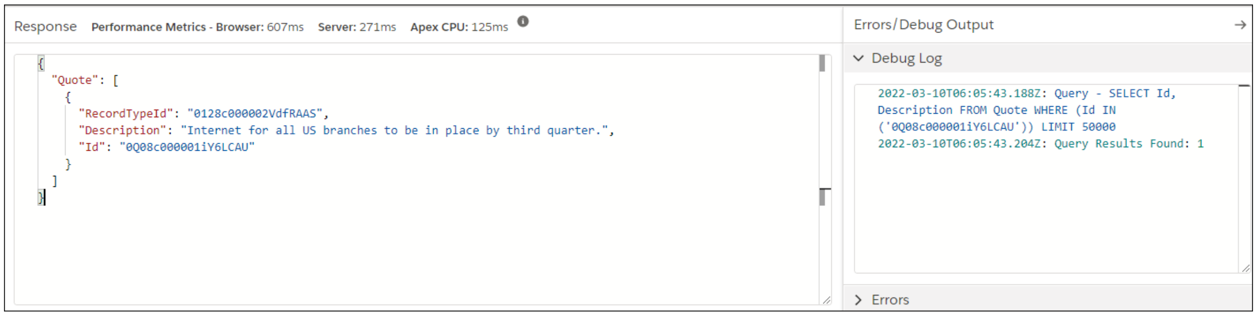


Interface Type	Input Type	Output Type	Required
Turbo Extract	JSON	JSON	

Key	Value
QuoteId	0Q08c000001iY6LCAU

Name	Content
Key	QuoteId
Value	[Paste the Unique Quote Reference]

6. Once you're done, click **Execute**. Your DataRaptor will run, extracting the data from the quote. You should see the record type ID, description from your quote and the quote Id in the Response pane. If not, check the Errors pane on the right for where you went wrong.



```
{
  "Quote": [
    {
      "RecordTypeId": "0128c000002VdfRAAS",
      "Description": "Internet for all US branches to be in place by third quarter.",
      "Id": "0Q08c000001iY6LCAU"
    }
  ]
}
```

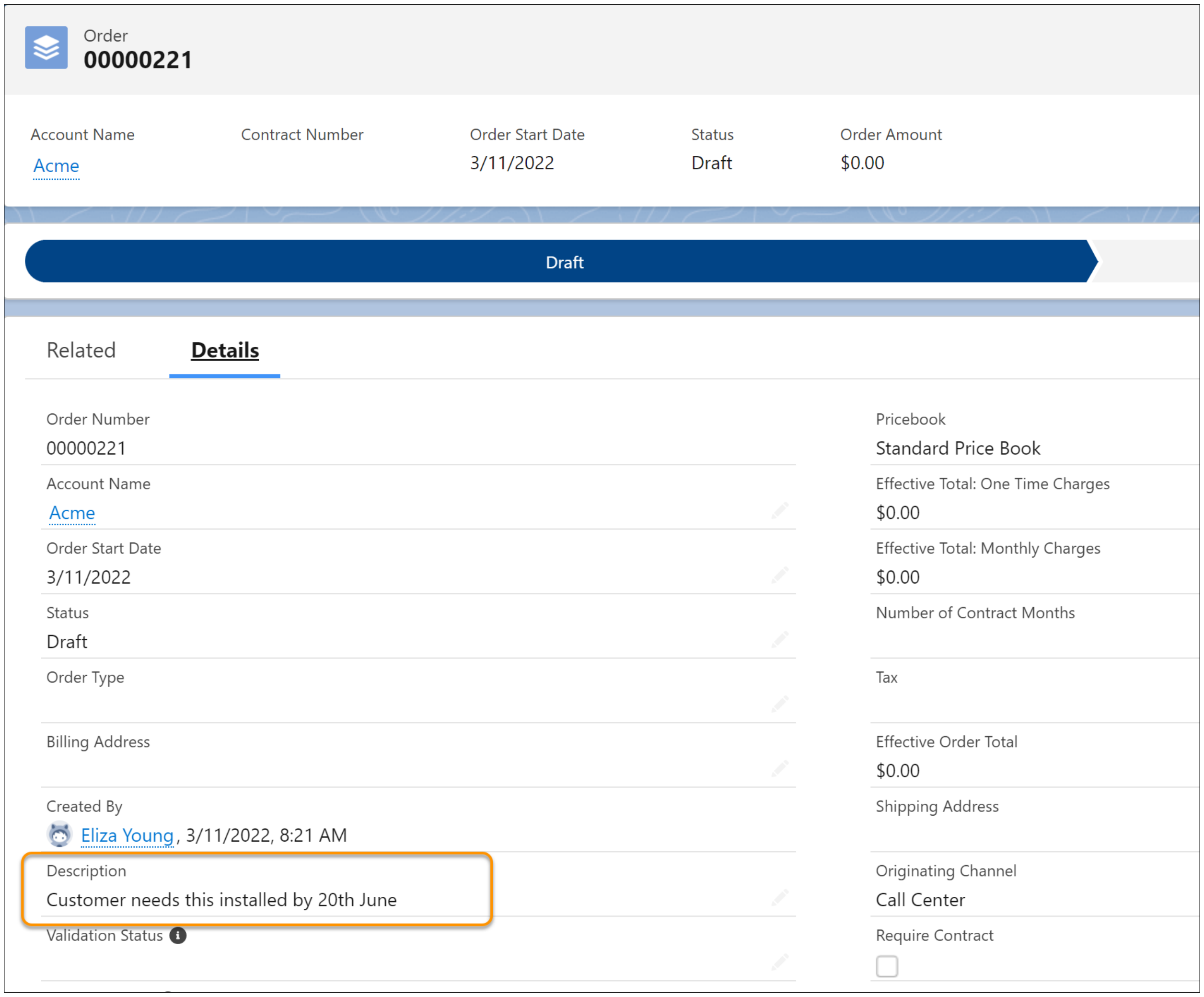
```
2022-03-10T06:05:43.188Z: Query - SELECT Id,
Description FROM Quote WHERE (Id IN
('0Q08c000001iY6LCAU')) LIMIT 50000
2022-03-10T06:05:43.204Z: Query Results Found: 1
```

If your DataRaptor returned the correct information, well done! You're ready to move onto the next exercise.

Exercise 6: Customize an Integration Procedure

Scenario

You have the data you need to add to the order. Now it's time to add it into an integration procedure so it works alongside other DataRaptors, calculations and actions to add the Quote Description to the Description of the primary order when the order is created, as shown here:


Order
00000221

Account Name	Contract Number	Order Start Date	Status	Order Amount
Acme		3/11/2022	Draft	$0.00

Order Number: 00000221	Pricebook: Standard Price Book
Account Name: Acme	Effective Total: One Time Charges: $0.00
Order Start Date: 3/11/2022	Effective Total: Monthly Charges: $0.00
Status: Draft	Number of Contract Months:
Order Type:	Tax:
Billing Address:	Effective Order Total: $0.00
Created By: Eliza Young, 3/11/2022, 8:21 AM	Shipping Address:
Description: Customer needs this installed by 20th June	Originating Channel: Call Center
Validation Status:	Require Contract: ☐

However, you're a bit confused about which Integration Procedure you should use. Time to do some further investigation.

Goals

- Locate and review the appropriate ESM Integration Procedure.
- Add a DataRaptor to an ESM Integration Procedure.
- Edit an Integration Procedure to map extracted fields to destination fields.
- Test an ESM Integration Procedure.

Tasks

1. Locate and review the appropriate ESM Integration Procedure.
2. Add a DataRaptor to the Integration Procedure.
3. Extend field mapping in the Integration Procedure.
4. Test the Integration Procedure.

Time: 20 mins

ESM Integration Procedures

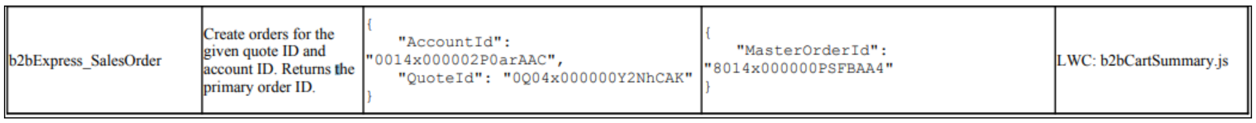
ESM uses Vlocity Integration Procedures (VIPs) to interact with many types of data, including REST APIs and DataRaptors, and process the data in multiple steps.ESM's VIPs can be invoked from OmniScripts and Cards. Many are invoked using ESM APIs, which are in turn called by the ESM SDK and LWCs, which you learned about earlier in this course.

There are over 30 VIPs used by ESM. In this exercise, you learn how to locate the VIP you need. You'll then be reviewing and extending one of the key ESM VIPs: the SalesOrder.

Task 1: Locate and review the appropriate ESM Integration Procedure

Go to the ESM documentation and take a look at the Integration Procedures information. If you haven't already, download the ESM Integration Procedures pdf. This is a really useful document, as it lists all the VIPs used by ESM, what they do, their input, output, and where the information is used.

Scroll through or search the list to find the VIP that creates orders.

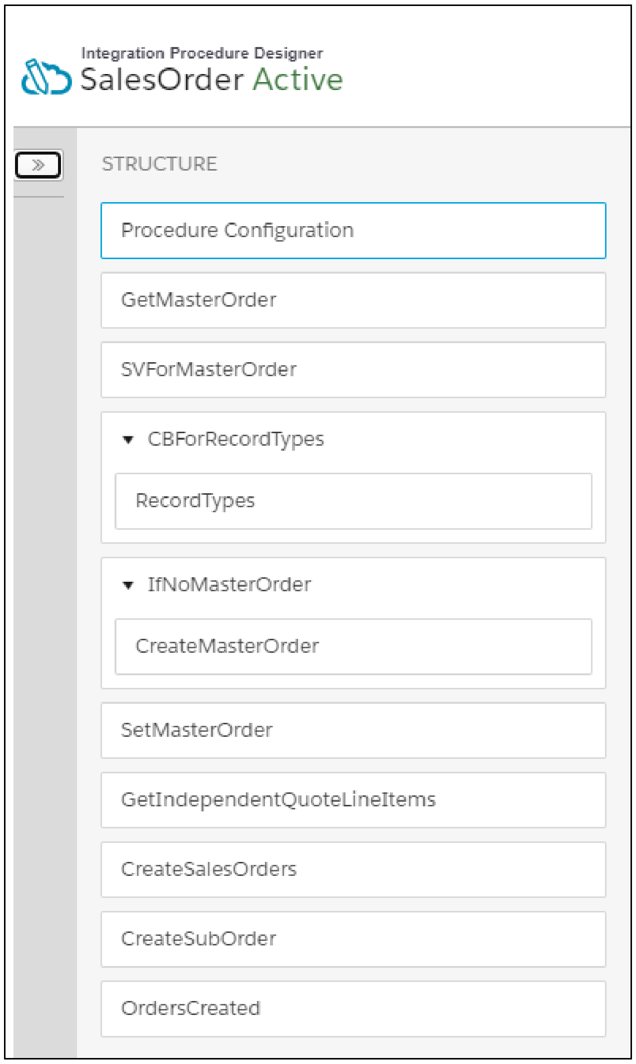


b2bExpress_SalesOrder	Create orders for the given quote ID and account ID. Returns the primary order ID.	`{ "AccountId": "0014x000002P0arAAC", "QuoteId": "0Q04x000000Y2NhCAK" }`	`{ "MasterOrderId": "8014x000000PSFBAA4" }`	LWC: b2bCartSummary.js

Notice this VIP uses the Account Id and Quote Id to create an order, then returns a primary Order Id. Let's take a closer look at the SalesOrder VIP.

1. From the **App Launcher**, locate and select **OmniStudio**. From the OmniStudio Navigation Menu, select **OmniStudio Integration Procedures**.
2. Locate and open the **b2bExpress/SalesOrder** Integration Procedure.

Review the structure. It seems like this integration procedure is doing a bit more than just creating a primary order!


Integration Procedure Designer
SalesOrder Active

STRUCTURE

- Procedure Configuration
- GetMasterOrder
- SVForMasterOrder
- ▾ CBForRecordTypes
  - RecordTypes
- ▾ IfNoMasterOrder
  - CreateMasterOrder
- SetMasterOrder
- GetIndependentQuoteLineItems
- CreateSalesOrders
- CreateSubOrder
- OrdersCreated

1. Procedure Configuration holds general details including the name, description, and version.
2. GetMasterOrder calls the GetMasterOrderDetails DataRaptor (extract action).
3. SVForMasterOrder sets the ID, Status and Date on the primary order.
4. CBForRecordTypes references a cache block in the org cache to hold the information.
5. IfNoMasterOrder checks if the primary order ID is blank, then retrieves the primary order ID.
6. SetMasterOrder creates a primary order record if it doesn't exist, then sets the primary order ID.
7. GetIndependentQuoteLineItems calls the GetIndependentQuoteLineItems DataRaptor (extract action), which retrieves all the quote line items in the active quote which do not have a quote group ID or a quote member ID.
8. CreateSalesOrders calls another integration procedure, b2bExpress_CreateSubOrder, which:
   a. Loops through the quote line items, collecting the quote line item IDs
   b. Creates a sub order using the checkout method from the CpqAppHandler, and adding the Quote ID as the context ID, the Cart ID as the Quote ID. The

string of quote line item IDs created in the previous step are sent to the method to be cloned and included in the sub order.

   c. Calls the UpdateSubOrder DataRaptor, passing to it the account ID, Order ID, ParentOrderId and RecordTypeID identified earlier.
   d. Updates the Sub Order with the Account ID, order ID, parent order ID and record type ID.
   e. Returns the SubOrder details in JSON format.

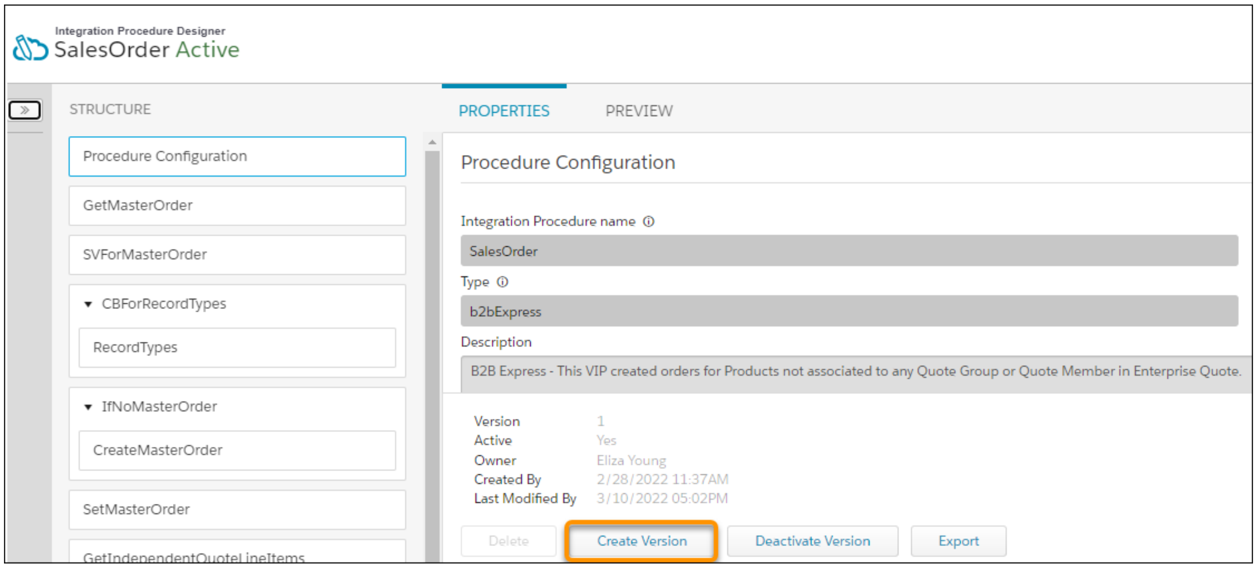
9. CreateSubOrder calls the createSubOrderEvent method, which uses the account ID, primary order ID, Quote ID, service account record type ID and sub order record type ID to create the required sub orders.

10. OrdersCreated returns the created orders in JSON format.

This is the integration procedure you need to amend, but what needs to be added? - and where? Take a few minutes to properly explore this Integration Procedure before continuing with the exercise.

What needs to be done?

1. First, create a new version of the active integration procedure. You don't want to break this one and not be able to roll it back!



Procedure Configuration

Integration Procedure name: SalesOrder

Type: b2bExpress

Description: B2B Express - This VIP created orders for Products not associated to any Quote Group or Quote Member in Enterprise Quote.

Version	1
Active	Yes
Owner	Eliza Young
Created By	2/28/2022 11:37AM
Last Modified By	3/10/2022 05:02PM

To create a new version, click **Create Version** at the bottom of the workspace.

2. Add a new DataRaptor Extract step, which uses the DataRaptor you created in the previous exercise to get the current quote's description. This should go after the Procedure Configuration block and before the primary order is created.

3. Add the quote description extracted by your new DataRaptor Extract step to the SetValues block for the primary order.

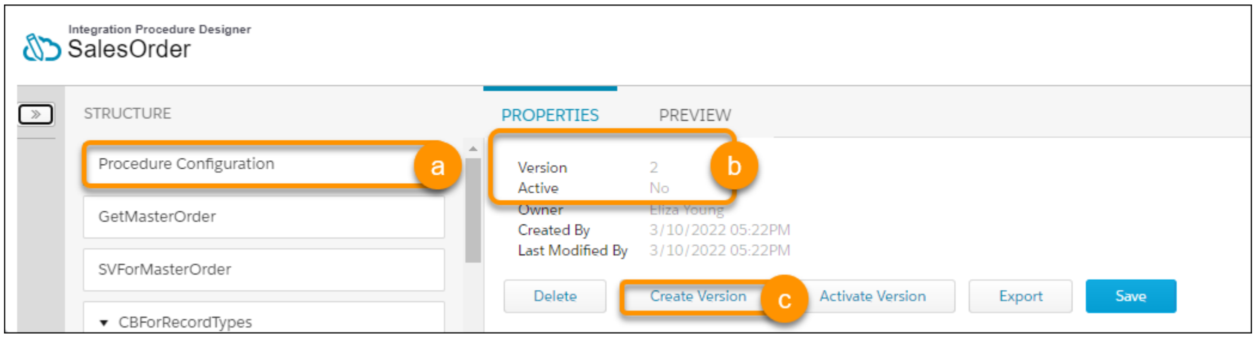
4. When the master order is created, you need to extend the extra payload in the field mapping to include the quote's description, mapped to the primary order description.

Task 2: Add a DataRaptor to the Integration Procedure.

In this task, you're going to add a DataRaptor Extract Action to your new version of the SalesOrder Integration Procedure. This Extract Action will call the b2bGetQuoteDescription DataRaptor you created in the previous exercise.

NOTE:
ALWAYS create a new version of your Integration Procedure before you make an amendment. That way, if you make a mistake, it is easier to roll back.

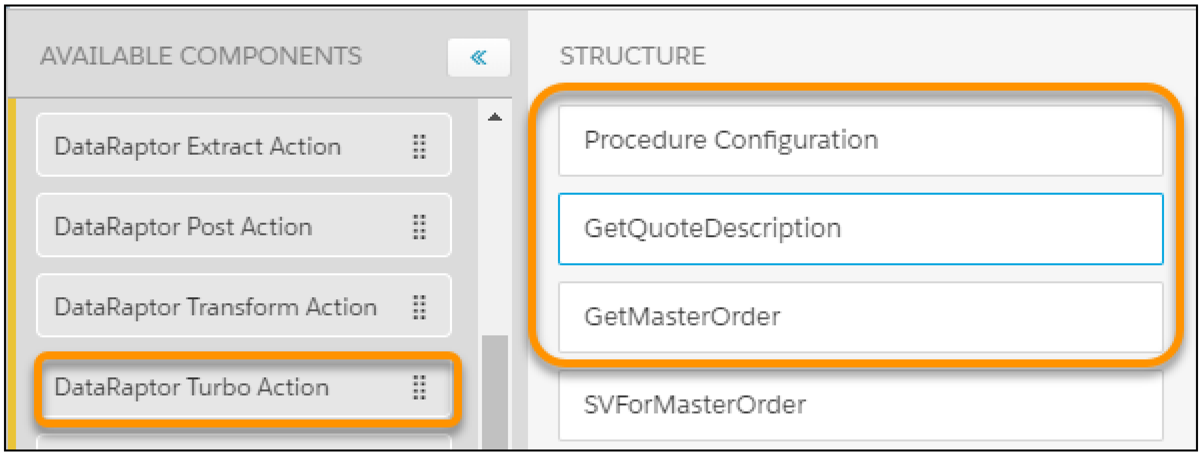
1. Check you're on the new version.



Version 2
Active No
Owner Eliza Young
Created By 3/10/2022 05:22PM
Last Modified By 3/10/2022 05:22PM

   a. In the SalesOrder VIP's **STRUCTURE** pane, click the **Procedure Configuration** block.
   b. Check the version number, and when it was created. It should be the one created by you a few minutes ago.
   c. If not, click **Create Version** to create a new version of the VIP before continuing with the exercise.

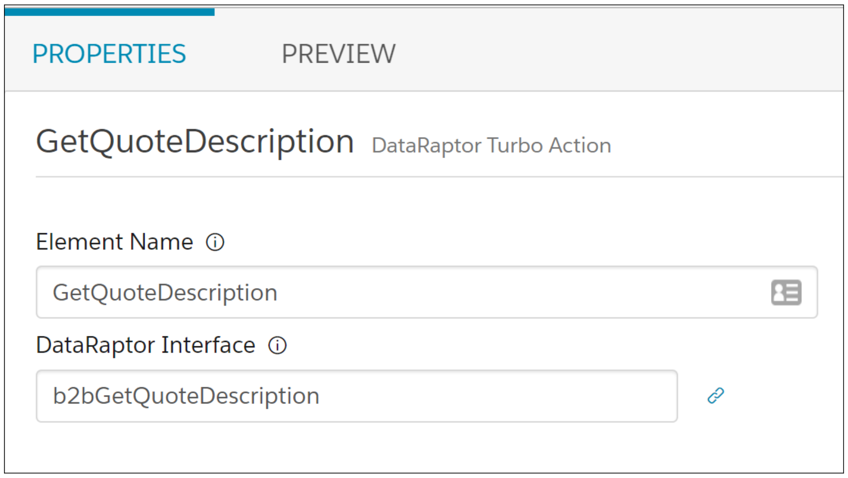
2. Drag a **DataRaptor Turbo Action** from the Available Components list to the VIP Structure, immediately after the Procedure Configuration block.



3. Configure the new DataRaptor Turbo Action to call the DataRaptor you created in the previous exercise.

Name	Content
Element Name:	`GetQuoteDescription`
DataRaptor Interface Name:	`b2bGetQuoteDescription`

The Element Name appears as the block name in the Integration Procedure structure, and the DataRaptor Interface is the name of the DataRaptor you created in the previous exercise.



GetQuoteDescription DataRaptor Turbo Action

Element Name: GetQuoteDescription

DataRaptor Interface: b2bGetQuoteDescription

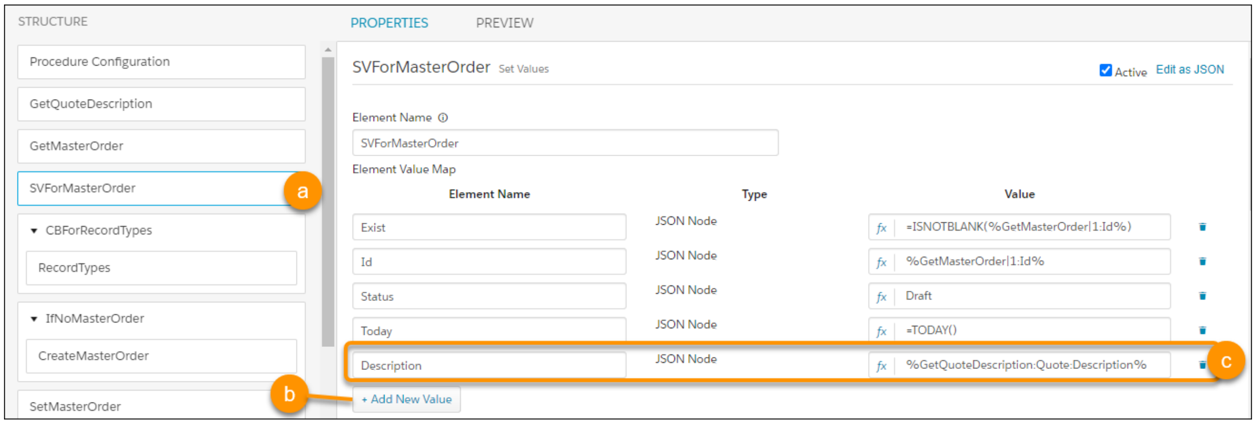
Task 3: Extend field mapping in the Integration Procedure.

Now you have the quote description information. The next step is to add it to the primary order when it is created.

The SalesOrder VIP has two blocks you're going to change:

- SVForMasterOrder, which sets the values to be added to the primary order.
- IfNoMasterOrder, which is a conditional block that checks if there is a primary order already created for the quote and, if not, executes the CreateMasterOrder remote action, which creates one. The CreateMasterOrder block maps the values set in SVForMasterOrder to input fields for the primary order.

1. First, set the values for the primary order in the **SVForMasterOrder** block.



SVForMasterOrder Set Values

Element Name: SVForMasterOrder

Element Value Map

Element Name	Type	Value
Exist	JSON Node	=ISNOTBLANK(%GetMasterOrder|1:Id%)
Id	JSON Node	%GetMasterOrder|1:Id%
Status	JSON Node	Draft
Today	JSON Node	=TODAY()
Description	JSON Node	%GetQuoteDescription:Quote:Description%

a. Click the **SVforMasterOrder** block from the structure panel.

b. Click **+ Add New Value** to add a new value in the Element Value Map.

c. Configure the element values:

Name	Content
Element Name:	`Description`
Value:	`%GetQuoteDescription:Quote:Description %`

2. Map the Value in the CreateMasterOrder Remote Action

Now you've set the primary order values, you need to map them in the extra payload of the **CreateMasterOrder** remote action.

a. In the **IfNoMasterOrder** block, click the **CreateMasterOrder** remote action.

b. Click **Edit as JSON** to edit the remote action using JSON.

c. Scroll down to the **extraPayload:input fields** in the code.

d. Add an extra key/value pair in the JSON to map the Description in the SVForMasterOrder block to the order description. The code is:

```
{
      "Description": "%SVForMasterOrder:Description%"
  },
```

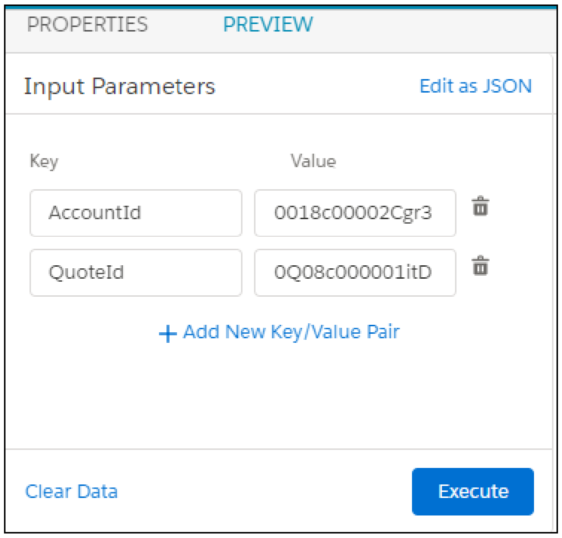
Here's an example of it shown in the code:

```
"extraPayload": {
      "fields": "Id",
      "inputFields": [
          {
              "effectivedate": "%SVForMasterOrder:Today%"
          },
          {
              "Description": "%SVForMasterOrder:Description%"
          },
          {
              "status": "%SVForMasterOrder:Status%"
          },
```

Task 4: Test the Integration Procedure

The final step is to check your changes by creating a new quote and using the integration procedure to convert it to an order.

1. Return to the Salesforce Sales application. Go to the **Acme** account and make a note of the unique account identifier, which is shown in the URL.
2. Create a new enterprise quote on the Acme account. Associate and configure products for two locations or subscribers.
3. IMPORTANT: If you made the changes to the OmniScript in Exercise 4, your quote description will be added as part of the enterprise quote creation process. If not, return to the Salesforce Quote record and add the quote description there.
4. Copy the unique identifier for the quote from the URL, as shown in Exercise 5, task 3.
5. Return to your SalesOrder integration procedure and click the **PREVIEW** tab. Add the Account Id and Quote Id to the input parameters.

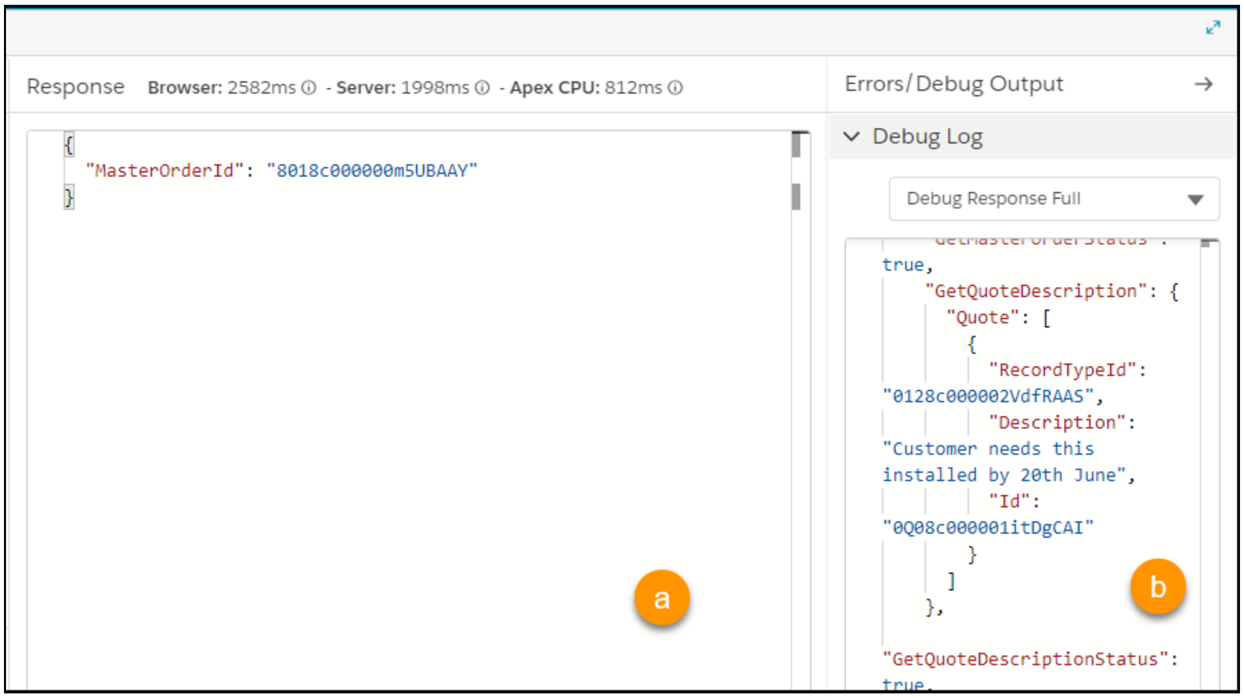


Key	Value
AccountId	0018c00002Cgr3
QuoteId	0Q08c000001itD

Parameter Name	Value
AccountId	Add the unique account identifier you copied from Acme's account URL.
QuoteId	Add the unique quote identifier you copied from your new quote's URL.

6. Click **Execute**.

7. If you've completed the tasks correctly, your order and sub-orders have been created for the customer, with the quote description shown as the order description on the primary order. You should see no errors (a), and you can check the debug log (b) to see how the information was passed from your VIP to your new order.

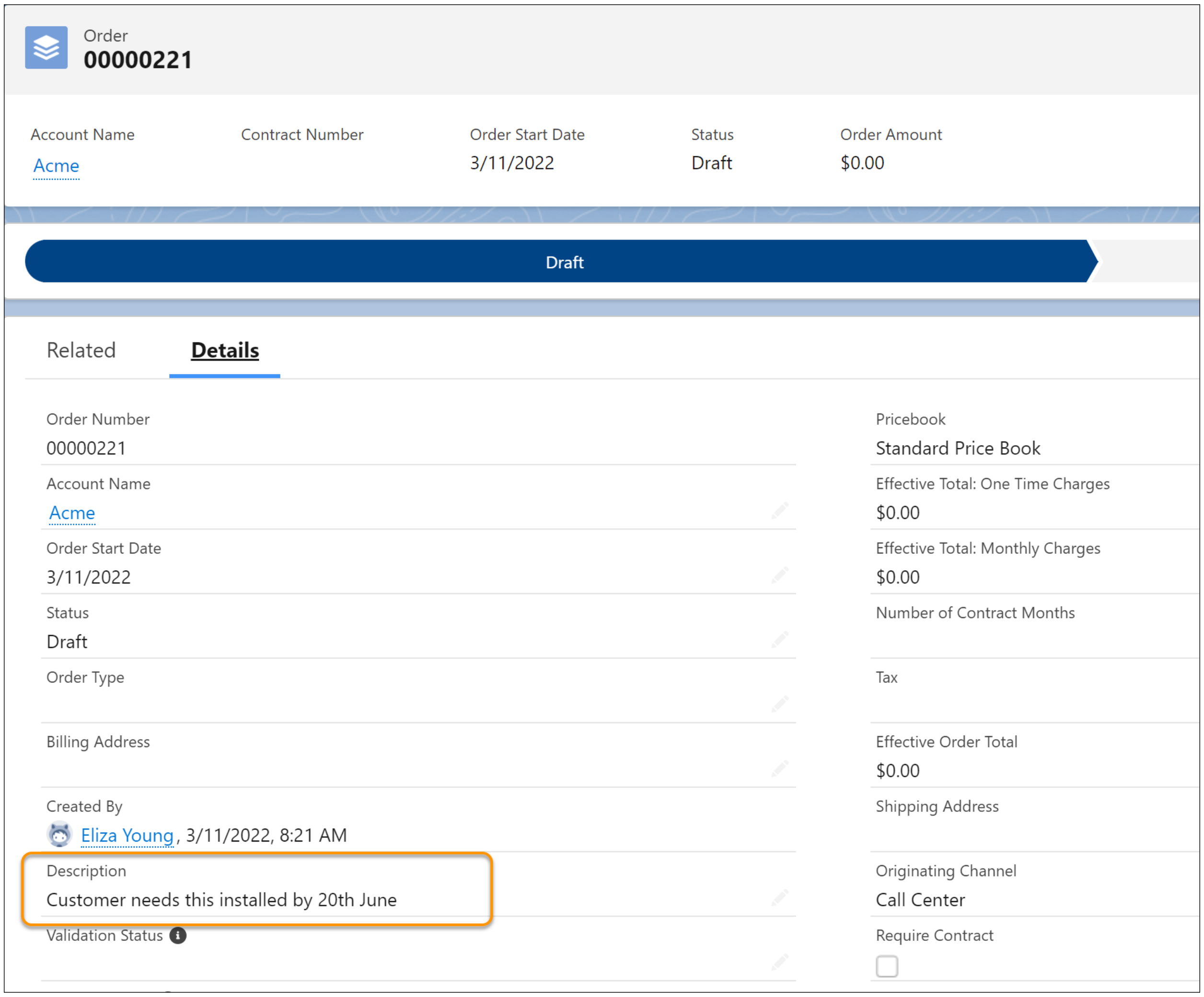


```
{
  "MasterOrderId": "8018c000000m5UBAAY"
}
```

```
true,
    "GetQuoteDescription": {
      "Quote": [
        {
          "RecordTypeId":
"0128c000002VdfRAAS",
          "Description":
"Customer needs this
installed by 20th June",
          "Id":
"0Q08c000001itDgCAI"
        }
      ]
    },

"GetQuoteDescriptionStatus":
true,
```

If you see errors in your response, work through the debug log to see where you went wrong then go back and amend your VIP.

8. The final step is to check your order. Return to the Sales application and open the primary order created for the quote.



Order
00000221

Account Name	Contract Number	Order Start Date	Status	Order Amount
Acme		3/11/2022	Draft	$0.00

Order Number	00000221	Pricebook	Standard Price Book
Account Name	Acme	Effective Total: One Time Charges	$0.00
Order Start Date	3/11/2022	Effective Total: Monthly Charges	$0.00
Status	Draft	Number of Contract Months	
Order Type		Tax	
Billing Address		Effective Order Total	$0.00
Created By	Eliza Young, 3/11/2022, 8:21 AM	Shipping Address	
Description	Customer needs this installed by 20th June	Originating Channel	Call Center
Validation Status		Require Contract	☐

Notice the description from the quote now also shows as the primary order description. Notice also that the quote description does not appear on any of the suborders.

Yay! All done!