Recap: Getting it Right, From the Beginning  
In the *Prepare for Your Implementation*module in the *Get to Know ESM* trail you learned the importance of gathering together all the relevant information required before embarking on any major development work. You’ve considered these key implementation steps:

1. **Discovery**: you identify what's going wrong and how ESM can fix it.
2. **Definition**: you set the terms of the engagement.
3. **Design**: you formulate a detailed description of the work to be done in the project, and decide who’ll do it.
4. **Delivery**: you create the solution.
5. **Deployment** and Scaling: you launch the solution, and scale by extending the project scope. For example, you might extend the solution to include new products, geographical regions, or markets?
6. **Monitoring and Control**: you track project progress and review each stage of the project, including after handover, once the implementation is complete.

You understand the importance of considering the impact your changes have on ESM users, especially when they’re working with very large and complex quotes. For example:

* What impact do frequent API calls to external systems have on the speed and reliability of ESM?
* How easy will it be to maintain and update the changes you’ve made in the future?

You get it: customization is an activity to be approached with caution and a big dose of common sense!

Examples of Common Customizations  
ESM is a fully working, out-of-the-box application, but you’ll want to customize it to suit your organization. Here are some commonly performed customizations.

**Store additional information on your account records**

What information should be maintained in your account records? The data mastery section in the *Get to Know ESM* trail outlines which applications in your organization should be responsible for which data. Ask your Salesforce implementation team if you need guidance.

It’s common to require more information for quote members, such as additional address fields for locations, and this may impact your service account data. Extending your quote member information includes changes to Lightning web components, and OmniScripts.

**Change the quote-to-order process**

To align ESM with your organization’s quote-to-order process, you may need to tweak the way quotes are converted to orders for consumption by the order-management system. Standard ESM functionality is to convert the quote directly to a master order with a sub-order for each quote member, after a button click. However, you can amend or add extra steps to the underlying data structures and processes. For example:

* After the sales person submits the quote to order, a second team may step in to complete the order details. This team may support orchestration and fulfillment by providing additional technical information, for example, about the customer site.
* Perhaps you don’t want to group your orders into parent orders and suborders. In this case, you don’t have to use the out-of-the-box parent order records, but instead have a single order for each quote member.

**Add custom action buttons and fields to the UI**

Consider how your end users enter and check information. Then assess how you need to adapt the existing ESM user experience (UX) to suit. ESM's interfaces are built using Lightning web components. You can modify and extend the user interface to add or change buttons, actions, and fields. To do this, clone and modify the relevant Lightning web components, and incorporate Integration Procedures and DataRaptors to present the right data to your users at the right time. You can work through some practical examples of these tasks later in this course.

**Guide users through commonly performed tasks**

Use OmniStudio to build guided flows with OmniScripts. Choose from these ESM-specific guided flows:

* b2bExpress/CreateQuote, which guides the user through creating quotes for enterprise customers, and
* b2bExpressCLM/generateProposalDocument, which guides the user through creating and generating a proposal document for the customer. Amend these flows to suit your own requirements using OmniScripts, or create new guided flows. For more information on creating guided flows using OmniScripts, refer to the OmniStudio Trail in Trailhead.

**Pass information between ESM and other applications**

Of course, ESM is just one of many applications you have in your company. How can you improve other business processes during your digital transformation journey? What jobs need to be done, who needs to do them, and when?  
The nature and order of tasks, and who performs them, determine when applications trigger your business processes. Understand which application should be the data master for particular information. This knowledge informs your integration requirements, and integrations are often complex and time-consuming. Later in this trail, you'll delve deeper into the integration options available to you. Additional integrations are added periodically, so it's worth checking the release roadmap before starting any work to develop your own.

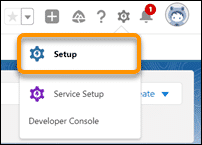
What's Already Available?  
What’s the most ignored common-sense decision relating to development? Checking you’re not wasting time by building something that already exists. Before you start your development work, **always**check if ESM has out-of-the-box features that you can tweak to suit your requirements.

Check Processes and Products  
Salesforce Industries Success Community is a great place to start when considering customizations to check you’re not reinventing the wheel in your project. The Community includes support groups, training materials, and documentation, including data models, configured examples, and information sheets for download. Community resources help ensure you’re following best practices in your configuration work.

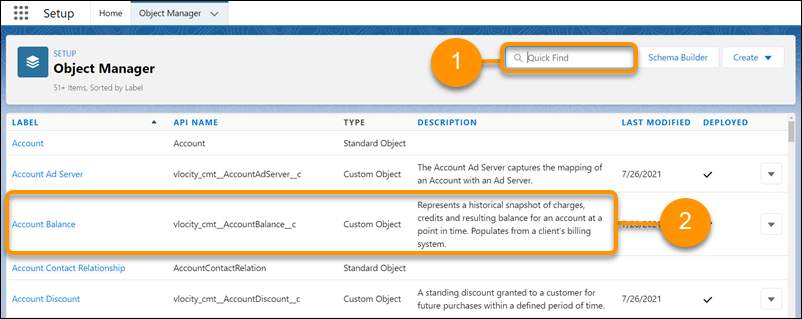
Check Existing Objects and Fields  
One of the most common requests when implementing new ESM projects is for extensions to the existing data model. Remember some fields may already be present in the data model but not visible on the out-of-the-box user interfaces.

To check if an object or field already exists in the system, use the Object Manager, which you can find in the Salesforce Setup menu of your org.

From Setup, use Quick Find to locate your Object Manager.



Use the Object Manager to view a list of all the objects in the ESM data model, including the names used in APIs, the object types, descriptions of the objects, and whether or not they are deployed. Objects whose API names end in \_\_c are specific to Salesforce Industries applications and are shown as Custom Objects, whereas Standard Objects are standard Salesforce objects.

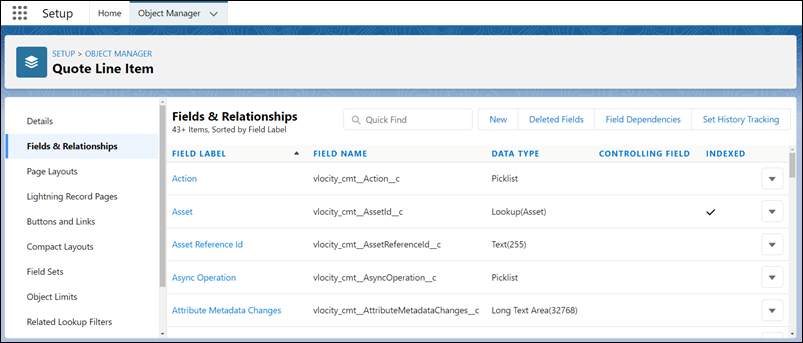


To find a particular object, type the name in the **Quick Find box (1)**.

**Account Balance (2)**is an example of a custom object that doesn’t appear in any ESM interfaces, but that you might want to implement. The object is already set up for you but you need to build the integration processes to implement it.

Click the object label to find out more about the object.

In this example, the Object Manager shows the Fields & Relationships list for the Quote Line Item object in a recent ESM implementation.

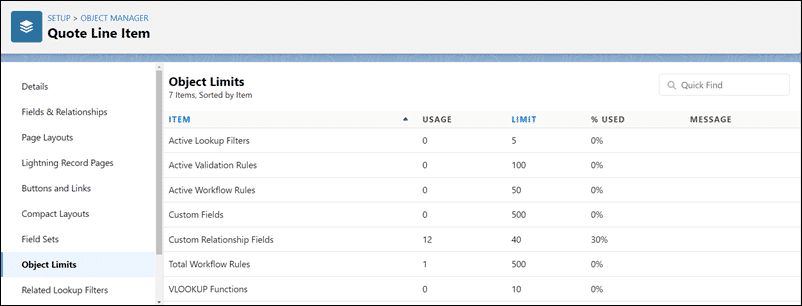


You can see details of:

* The field label
* The data type
* Any field dependencies
* If the field is indexed

Use the menu on the left to see useful information about the Quote Line Item object, including which page layouts feature quote line items, which buttons and links reference this object, and any related lookup filters.

Click **Object Limits** to check that the object is not exceeding the object limits imposed by the Salesforce application.



What are ESM Lightning Web Components?  
Lightning web components are just one of the many tools you can use to customize ESM to suit your organization’s requirements.

ESM’s Lightning web components are open-source, industry-specific components and widgets built on web component standards. They work with all modern browsers and can be distributed with any JavaScript library or framework. You use web components to create new custom, reusable, encapsulated HTML tags for use in your web pages or applications. Think of them as building blocks that go together to create your application.

Customized Salesforce applications before? - then you may have already used Lightning web components, or you may have used Aura Components. If you used Aura Components in the past, you’ll be pleased to know that Lightning web components perform better and are easier to develop than their predecessors.

In this module, you’ll focus on modifying the ESM user interface by creating new Lightning web components to extend those contained in the ESM managed package. However, to see just how easy it is to create and amend any Salesforce Lightning web component, watch the Create a Hello World Lightning Web Component demonstration in Trailhead. The link is included in the Resources for this lesson.

**ESM Lightning Web Components, Salesforce Industries Lightning Web Components, and the SDK**  
ESM Lightning web components supplement the Salesforce Industries Lightning web components by incorporating calls to the ESM software development kit (SDK). The SDK makes calls to ESM APIs, Cart-Based APIs, Digital Commerce cacheable APIs, and Integration Procedures. They are intended for use on the Salesforce platform and follow the same standards as Salesforce's Lightning web components.

When creating a new Lightning web component for ESM, you edit three files:



* The xml file, which contains the metadata for the Lightning web component including the API version and namespace to be used.
* The html file, which contains the template tag. This contains the HTML that defines the structure of your component (such as labels and images).
* The JavaScript file, which contains the code to make your component work, including inputs, data, events, and changes to state.

These files are stored in a folder with the same name as your Lightning web component.

Lightning Web Component Slots  
Each Lightning web component has one or more named “Slots”. These can be added to a component’s HTML template so a parent component can pass markup into the component.

For example, if you wanted to include “Your Name” in the myName slot in your HTML template for your Lightning web component, you would add this code:

<template>

<p>Name: <slot name="myName">Your Name</slot></p>

</template>

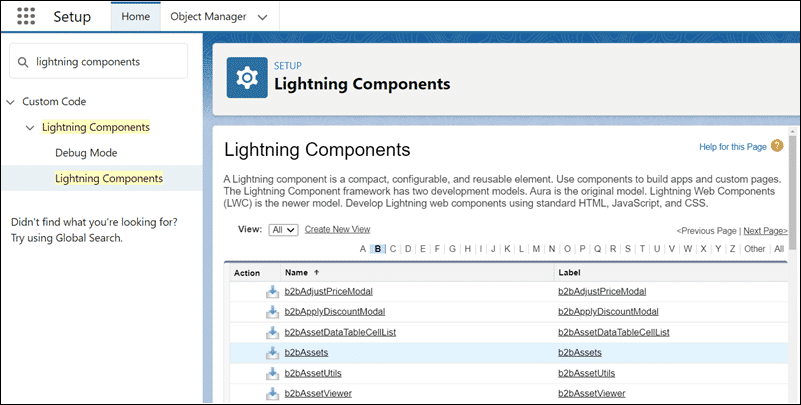
There’s an excellent explanation of using slots, including how to run code on slotchange, in the Developer Guide, which is linked in the Resources section of this lesson.

**What are the ESM-Specific Lightning Web Components?**  
There are three different ways you can identify the Lightning web components used in your ESM org:

* View the Lightning Components in the Salesforce Setup.
* Review the ESM documentation.
* Use the Inspect function on your browser.

Let’s take a look at each.

**Lightning Components in Salesforce Setup**  
View a list of the ESM-specific Lightning web components in your ESM org by going to **Setup**then searching for**Lightning Components**.

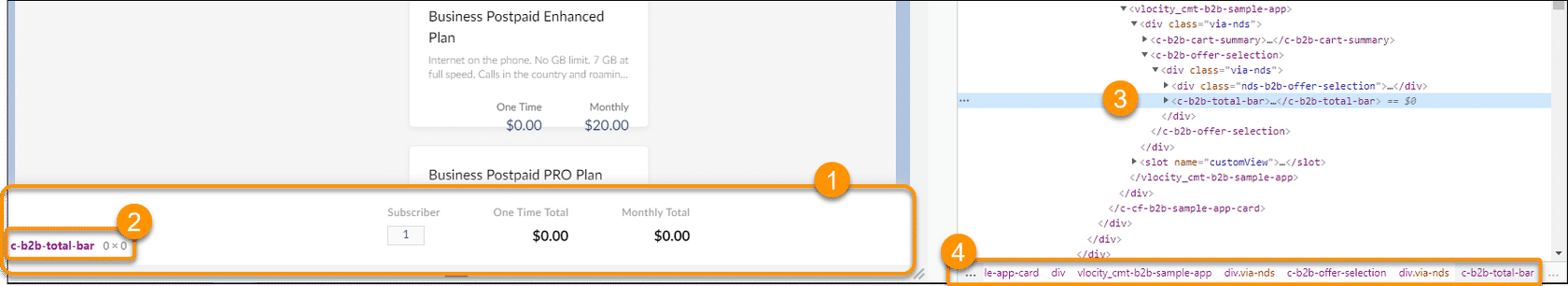


The Lightning web components for ESM mostly begin with b2b, so click the B in the alphabet filter at the top of the Lightning web component list to view them.

Click on the file links within the structure of the Lightning web component to view details of the structure and contents of each Lightning web component. Notice each Lightning web component contains HTML, JavaScript, CSS, and XML files.

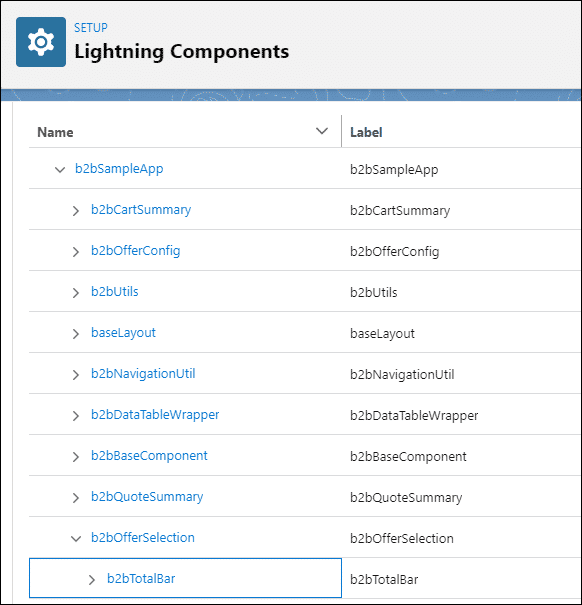
**ESM Documentation**  
There is an extensive description of each ESM Lightning web component, including slots and associated sample GUI images where appropriate, in your user documentation. This is available in the Success Community (and linked in the Resources for this lesson).

**Inspect from the Browser**  
You can also use the **Inspect** function on your browser to locate the details of the Lightning web components and associated slots currently used in your implementation. Here’s an example using the Chrome browser.



* Select the area you want to inspect, and choose **Inspect**from your context menu.
* The name of your Lightning web component will be displayed. Now you know the total bar in the product selection pane is called **c-b2b-total-bar**.
* Click on the Lightning web component in the code to view any slots that have been associated with it.

The path to locate your Lightning web component is shown at the bottom. In this example, you can find the b2bTotalBar Lightning web component in the **b2bSampleApp →b2bOfferSelection**. Let’s check that in the Lightning Components list in Salesforce.



The Sample App  
The b2bSampleApp Lightning web component, which is available in the B2BSampleApp folder, is a main component of the ESM application. Using this Lightning web component, you can add a **customView**slot, to add any component other than those provided in the b2bSampleApp.

Conclusion  
In this lesson you learned different ways to locate and customize ESM Lightning web components. In the next lessons you’re going to try some of these customizations, including:

* Changing the theme of a Lightning web component.
* Creating new Lightning web components to extend ESM functionality.

What is a Design System?  
A design system is a collection of repeatable design patterns and reusable code, referred to as components.

Components include buttons, menus, sounds, animations, visual patterns, and more. In some design systems, they may be fully functional and built on a UI platform. In others, they remain platform independent.

Think of these components as building blocks: they don’t do much on their own, but when you put them together, you can build almost anything.

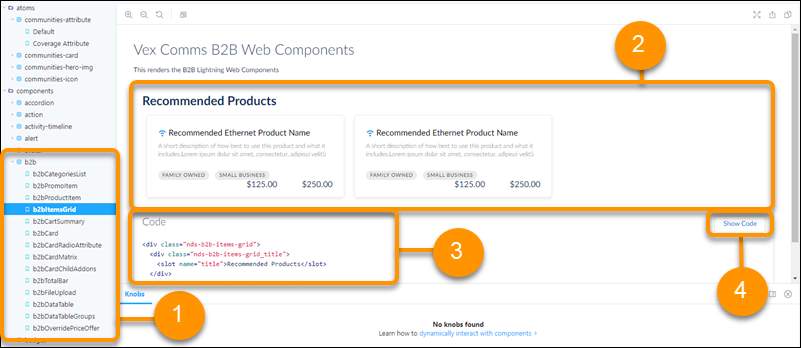
Set ESM’s “Look and Feel” with a Design System  
Although you know that ESM is brilliant already, it would be even more awesome if it matched the style guidelines within your organization, right? This is where the Design System steps in. You can choose to use Salesforce’s Lightning Design System (SLDS), or the Newport Design System (NDS), which extends SLDS specifically for Industries Clouds. In this lesson, you’ll learn how to use NDS - but either design system may be used for ESM.

|  |  |
| --- | --- |
| 📝 | Want to use SLDS instead? Take the *Lightning Design System Development for Designers*module in Trailhead, which is linked in the Resources section of this lesson. |

**What Does NDS Do?**  
At the risk of being acronym-tastic, NDS is used to configure and structure ESM’s html components and CSS framework. NDS rethemes all of ESM’s components in one place, generating a custom, optimized CSS file that can be used in all future pages.

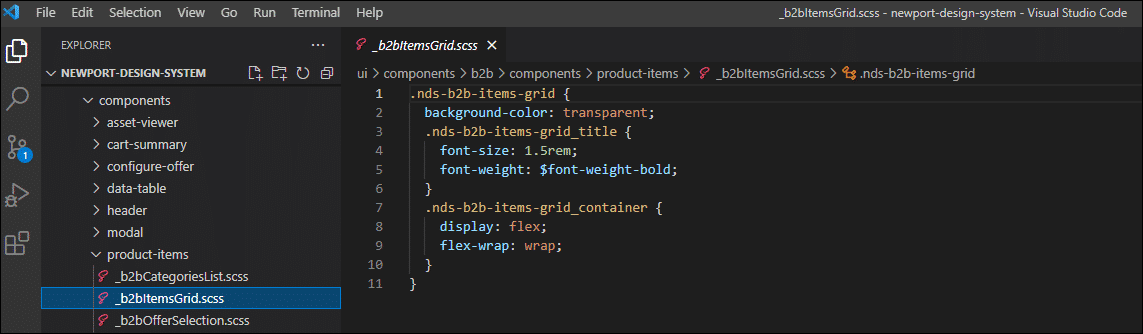
To use NDS, you should be familiar with working in a command line interface (CLI), and using Git repositories. There are links in the Resources if you need additional help with these.

NDS Storybook  
NDS Storybook is a previewer supplied with NDS. Components unique to the ESM user interface can be viewed in the NDS Storybook, within the b2b folder, as shown here.

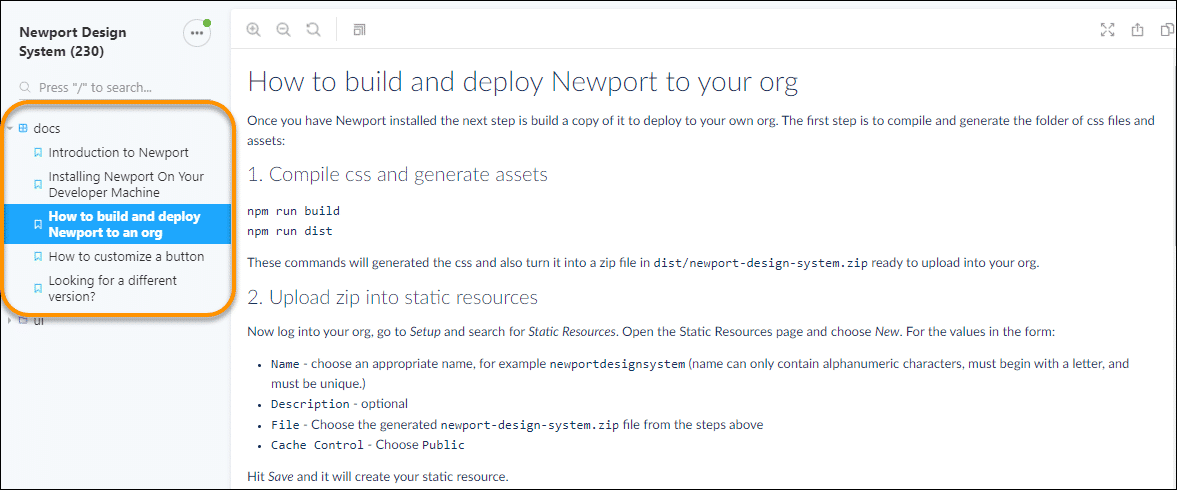


1. On the left is the navigation pane, with a list of items to choose from. Components unique to the ESM user interface are held in the UI/components/b2b folder.
2. Click on a component to view it. You’ll see a preview of the component in the panel on the right. In this example, you’re looking at the items grid. From the previous lesson, you know this is where the product tiles (b2bProductItem) are stored.
3. Below that is the code for the component. Notice the slot for the title.
4. Click **Show Code** to toggle in and out of the code detail for this component.

To amend CSS for this component across the whole of ESM, just change the code identified in the Storybook using your code editor, such as Visual Studio Code.



Once you’re done, compile it, upload it into your org, and configure your custom settings so ESM knows where to look to implement the changes.



There are detailed instructions and demonstrations on completing these tasks in the NDS Documents folder in Storybook, and in the documentation held in the Resources section of this lesson.

Working with Themes in a Managed Package  
ESM Lightning web components are part of the managed package, so you can’t directly edit the HTML or CSS. However, you can inspect the source code using Chrome Dev Tools, and you can preview the components using the NDS Storybook.

To modify the theme or CSS you clone the NDS repository, make the necessary changes using a code editor such as Visual Studio, then upload your changes.

When you change the theme using the CSS class definition, all Lightning web components that use HTML with that CSS class name will change. The exercise in the lab guide associated with this lesson walks you through this process.

Getting Started  
You need to add some custom functionality to ESM to suit your organization. The first step should always be to check the functionality is not already available. There are many buttons, actions, and fields already built in ESM that you can add to your cards through simple configuration.

If you’ve checked and you can’t find what you need, you’ll want to customize your Lightning web components.

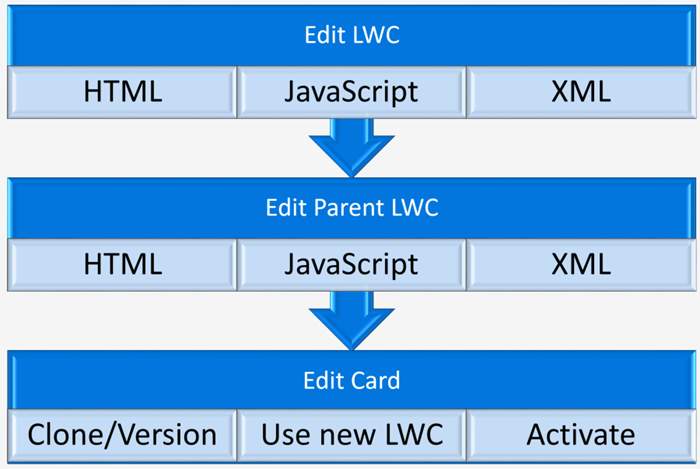
In lesson 1 you learned about customizing Lightning web components using slots. In this lesson you’ll learn how to add your own buttons, actions, and fields to ESM’s Lightning web components with custom code. Note that ESM Lightning web components are part of the managed package. Therefore, you can’t extend them directly. Instead, you create a new Lightning web component, extending the one from the managed package.

When you create your new Lightning web component, there are three files to edit:

* HTML
* Javascript
* Metadata (XML)

These files may be edited using Salesforce’s Lightning web component Designer, or with a code editor such as Visual Studio. You extend the new Lightning web component, replacing the original template or replacing specific slots with your own custom HTML overrides. Once you’re done, you set the**isExposed=true** flag in your new Lightning web component meta.xml file, so that the Lightning web component can be customized further at a later date.

This is the process for customizing your Lightning web components:



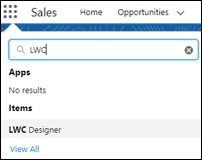
When you customize your Lightning web component, you need to amend each of the three code files. You’ll also need to amend any parent Lightning web components and the cards that use the Lightning web components.

Create and Edit with LWC Designer  
In the examples in this lesson, and in your practice exercises, you’ll use the LWC Designer - but you could instead use your favorite code editor.

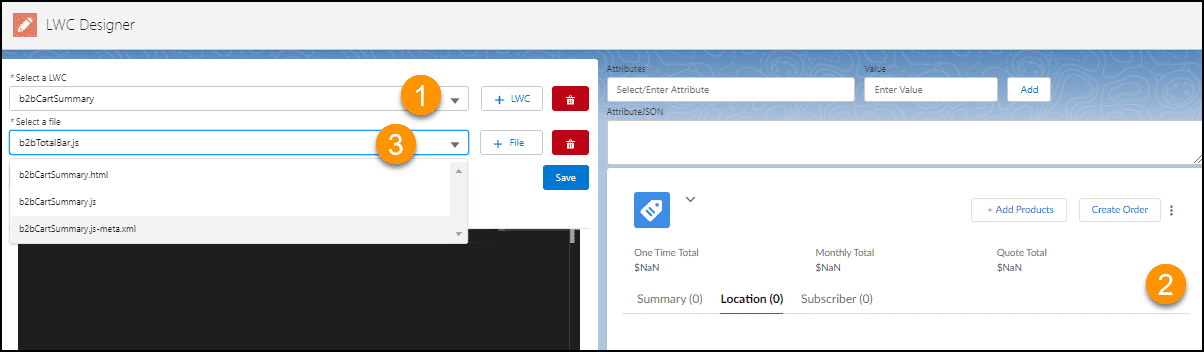
To access the LWC Designer:

In Salesforce, click the **App Launcher,** type *LWC*in the Quick Search box and you’ll find the LWC Designer.

Click **LWC Designer** to start the editor.

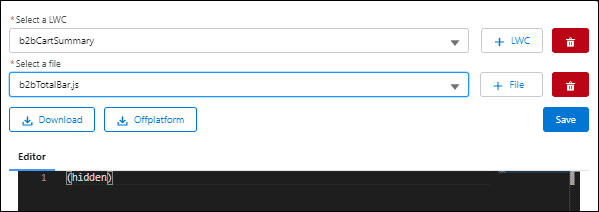


Let’s try editing a Lightning web component, such as the *b2bCartSummary*.



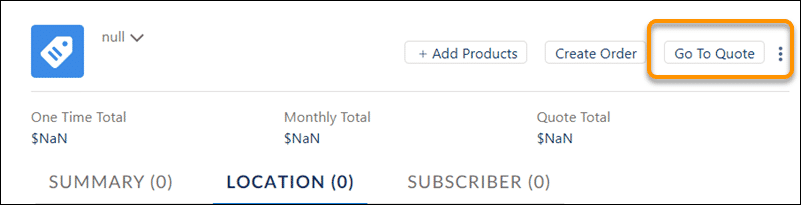
1. In **Select a LWC**, type the name of the Lightning web component to edit. Here the Lightning web component is *b2bCartSummary*.
2. A preview of the Lightning web component shows in the preview pane on the right.
3. From **Select a file**, choose the file in the Lightning web component you want to edit. Here the JavaScript file is selected.

The code for the selected file is shown in the code editor. If the Lightning web component is part of the managed package, you won’t be able to edit it or see the code. Instead, the code will be displayed as (hidden) like this:



If you encounter this situation, you'll need to create a new Lightning web component that extends the selected Lightning web component, as you’re unable to edit the original. Let’s look at how to do that next.

Scenario: Add a New Button in the Cart Summary Pane  
In this scenario you’re going to extend the b2bCartSummary Lightning web component, adding a new “Go to Quote” button to the Cart Summary pane.



There are three steps you’ll need to complete:

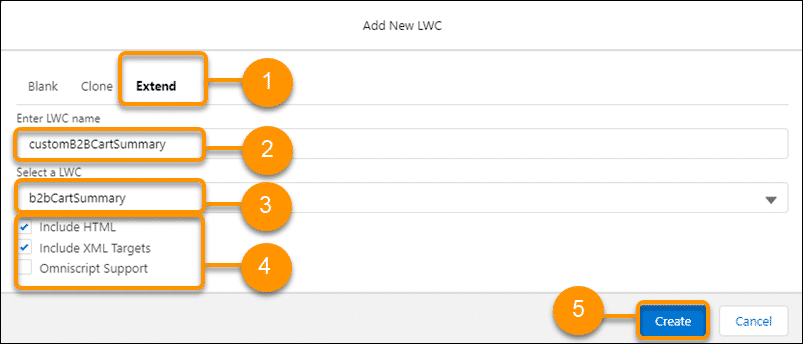
1. **Extend the Lightning web component**  
   The b2bCartSummary Lightning web component is part of the managed package, so can't be amended. Instead, you’ll create a new custom b2bCartSummary, extending the original. You'll then edit the HTML, JavaScript and metadata files for the new Lightning web component.
2. **Amend the parent Lightning web component**  
   The b2bCartSummary is used by the b2bSampleApp Lightning web component as part of the Cart interface. The b2bSampleApp Lightning web component is part of the managed package and can’t be edited. Therefore, you’ll need to create a new, custom b2bSampleApp Lightning web component to include the new custom b2bCartSummary, and amend the associated HTML, JavaScript and XML files.
3. **Update the associated Card**  
   The final step is to ensure the b2bSampleApp card is updated to use information from the new custom b2bSampleApp Lightning web component. As this card is also part of the managed package, you’ll clone the card and configure the cloned card to use the custom Lightning web component.

There are three files you need to code to get your Lightning web component to work. These are:

* The HTML template
* The JavaScript file
* The Metadata XML file

If you’d like to try this example for yourself, copy the code from the appropriate section of the Sample Code at the end this lesson. You can use this code to build the associated files for your Lightning web components.

**Step 1: Create a new Lightning web component extending the managed package Lightning web component**

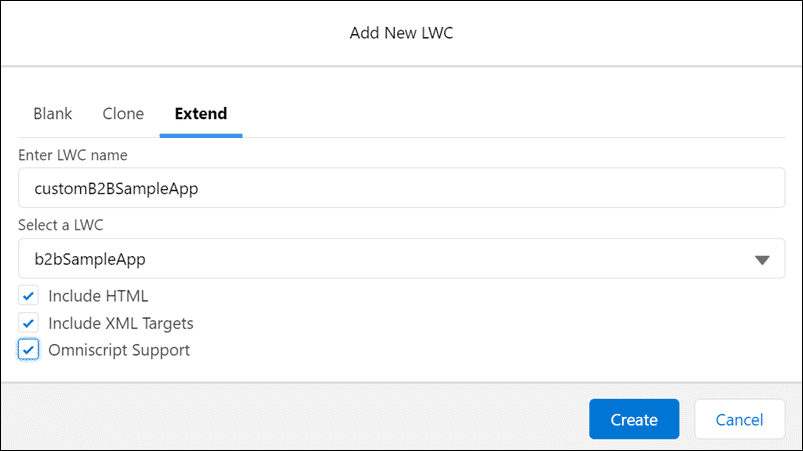


In your ESM playground, use the App Launcher to open the LWC Designer and click the **+LWC** button to create a new Lightning web component.

1. In the **Add New LWC** dialog box, click **Extend**.
2. Enter the name of your new custom Lightning web component. Here we’ve called it*customB2BCartSummary*.
3. Select the Lightning web component you wish to extend your code from. Here we’ve chosen **b2bCartSummary**.
4. Select**Include HTML**and **Include XML Targets**.
5. Click **Create**.

A new custom Lightning web component is created, using elements of code from the existing Lightning web component, and is opened in LWC Designer.

**Step 2: Extend the parent Lightning web component to incorporate your new Lightning web component**



At the top of the LWC Designer, click the **+LWC** button to create a new Lightning web component.

In the **Add New LWC** dialog box, go to the **Extend**tab and add details for the new Lightning web component.

Here we’re creating a new Lightning web component called customB2BSampleApp, which is based on the managed package’s b2bSampleApp. As this Lightning web component is used within OmniStudio elements, we’re including OmniScript Support, along with HTML and XML targets.

Click **Create**. Your new Lightning web component is created and is opened in LWC Designer. Next, repeat the process you completed for the customB2BCartSummary, editing the HTML template, Javascript and metadata XML files to ensure the new Lightning web component uses the child Lightning web component you created in the previous step.

**Step 3: Clone and Update the Associated Card**

If your Lightning web component is used by a Card, you’ll need to update the Card to display the changes. ESM Cards are part of the managed package, and can’t be updated directly, so start by cloning the card. You’ll then update the associated Layout Lightning web component to be the Lightning web component you created earlier before deploying and activating your new card.

**Sample Code**

Sample HTML Code - b2bCartSummary

------------------------------------------------------------------

<template>

<vlocity\_cmt-b2b-cart-summary name-list={nList}>

<!-- existing slot override -->

<div slot="action">

<!--Existing buttons code is needed here if you want to reuse the existing feature provided, along with the new button implementation-->

<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>

<!-- Injecting code for sample custom implementation for new button -->

<div class="nds-m-left\_medium">

<vlocity\_cmt-b2b-button theme="nds" variant="neutral" label="Go To Quote" icon-size="x-small"

onclick={navigateToQuote}>

</vlocity\_cmt-b2b-button>

</div>

</div>

<!-- utility actions code is needed here if you want 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>

</template>

------------------------------------------------------------------

Sample JavaScript- b2bCartSummary

------------------------------------------------------------------

import B2bCartSummary from 'vlocity\_cmt/b2bCartSummary';

export default class customB2BCartSummary extends B2bCartSummary {

connectedCallback(){

super.connectedCallback();

//nList can be any variable, need to store the route Quote property

this.nList = this.route.Quote;

}

navigateToQuote(){

//custom code here

alert('Go To Quote')

}

}

-----------------------------------------------------------------

Metadata XML Sample Code - b2bCartSummary

-----------------------------------------------------------------

<?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>customB2BCartSummary</masterLabel>

<targets>

<target>lightning\_\_RecordPage</target>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_HomePage</target>

</targets>

</LightningComponentBundle>

---------------------------------------------------------------------------------------------------------------------------

Custom HTML code for customB2BSampleApp

------------------------------------------------------------------

<template>

<div class="via-nds">

<div if:true={initApp}>

<c-custom-b2-b-cart-summary></c-custom-b2-b-cart-summary>

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

<vlocity\_cmt-b2b-data-table-wrapper></vlocity\_cmt-b2b-data-table-wrapper>

</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>

-------------------------------------------------------------------

Sample customB2BSampleApp JavaScript

-------------------------------------------------------------------

import B2BSampleApp from "vlocity\_cmt/b2bSampleApp";

import customB2BSampleAppTemplate from "./customB2BSampleApp.html";

/\*\*

\* @class customB2BSampleApp

\* @extends {LightningElement} Extends the b2bSampleApp

\*

\* @classdesc

\* customB2BSampleApp is the component for navigating between components.<br/><br/>

\*/

export default class customB2BSampleApp extends B2BSampleApp {

render(){

return customB2BSampleAppTemplate;

}

}

---------------------------------------------------------------------

Sample customB2BSampleApp metadata

---------------------------------------------------------------------

<?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>customB2BSampleApp</masterLabel>

<targets>

<target>lightning\_\_RecordPage</target>

<target>lightning\_\_AppPage</target>

<target>lightning\_\_HomePage</target>

</targets>

</LightningComponentBundle>

**Recap: What is OmniStudio?**  
OmniStudio is a suite of services, components, and data model objects that combine to create Industry Cloud applications, such as ESM. You’ll use OmniStudio tools to create guided interactions in ESM using data from your Salesforce org and external data sources.

How does ESM use OmniStudio?  
ESM uses OmniStudio to create enterprise-specific guided flows, interfaces, and integrations. Tools include:

**OmniScripts**

OmniScripts contain user-interaction logic to guide the user through enterprise quote and proposal creation.

OmniScripts are created by dragging and dropping the elements you want in the process onto the OmniScript, so generally no coding is involved. The elements you can add to your OmniScript include:

* Actions, such as sending an email.
* Groups of steps or items.
* Functions, formulas, calculations, and messages.
* Input fields and lookups.
* Headings and text blocks.
* Logic branches.

There are several ESM-specific OmniScripts available out-of-the-box for you to extend if required.

Later in this course, you’ll customize remote actions, input fields, and interfaces for the CreateQuote OmniScript.

**OmniStudio Action buttons**

OmniStudio action buttons are added to cards to perform actions such as launching OmniScripts, or taking the user to an external site.

You added actions to slots in ESM when you completed the Lightning web components module of this course.

**DataRaptors**

DataRaptors transfer and transform data between Salesforce and the OmniScripts, Cards, and Integration Procedures tools used in ESM.

DataRaptors read or write Salesforce SObject data, JSON or XML data, or perform **single-step** data structure transformations. For example, you can use DataRaptors to filter, calculate or reformat data from your quote or account. They are not used to read from or write to CSV files, Apex classes, REST APIs, or external objects.

DataRaptors used only by ESM often have names beginning with b2b, such as b2bExpress\_GetLatestWorkingCart, but there are many other DataRaptors used by ESM that you’ll become familiar with when completing the Integration module for this course.

**Integration Procedures**

Integration Procedures bundle server-side data integration operations for efficiency and reuse. For example, you would use an Integration Procedure to call multiple DataRaptors to perform complex data read and transform operations.

There are many Integration Procedures used by ESM, and these will be covered as part of the integration module for this course. Some, but not all, have names beginning with b2bExpress to help you identify them.

**Cards**

Cards display data and launch actions.

In the ESM Lightning Web Components module you cloned and amended the **b2bSampleAppCard**Card.

When designing your Cards and OmniScripts, you can choose between two templates for applying brand guidelines and design specifications:

* Lightning, which references the Salesforce Lightning Design System.
* Newport, which references the Salesforce Industries style guide.

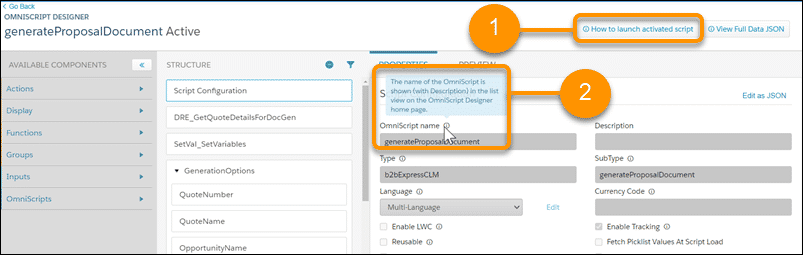
Although we recommend using the Newport Design System, longer term you may want to consider switching to the Salesforce Lightning Design System. The Trailhead course for the Salesforce Lightning Design System is included in the Resources for this lesson.

If you’ve completed the module in this course on Lightning web components, you’ve already used a few OmniStudio tools to customize ESM, including Cards, OmniStudio Actions, and the Newport Design System.

Where Can I Get Support on OmniStudio?  
If you’re just starting out with OmniStudio, complete the OmniStudio Trail in Trailhead, which is linked in the Resources for this lesson. This trail gives you practical experience of all aspects of OmniStudio and provides excellent guidance on best practices.

You’ve also got access to detailed information and guidance about OmniStudio tools in the Salesforce Industries Success Community documentation.

OmniStudio offers in-product help, so you can find contextual information and instructions within your design interface. Here’s an OmniScript example:



You have two options for in-product help and support when creating OmniScripts:

1. Click on the help button to launch full instructions on how to complete the task.
2. Hover your mouse over the information icons to view information about each element.

Finally, there are active support groups in both the Partner Community and the Salesforce Industries Success Community. These groups include Salesforce customers, partners, and employees, who have practical experience in implementing ESM across organizations.

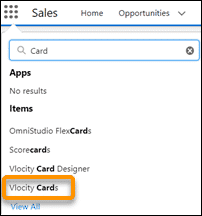
How Does ESM Use Cards?  
ESM uses Cards to build dynamic, engaging, context-sensitive user interfaces. Cards are containers for a related group of information. They display Salesforce object information along with discrete, clickable actions that change according to the context in which they appear, and based on the information they contain.

You create and edit Cards using a declarative tool called the Card Designer, which is accessed from the App Launcher. You add content to states and create conditions for any state to determine what content displays when. As you build your Card, you preview and test your interface with actual data from the Card Designer’s live Preview tab. Once you’re done, save and activate the Card to generate the component to add to your Lightning or Community page.

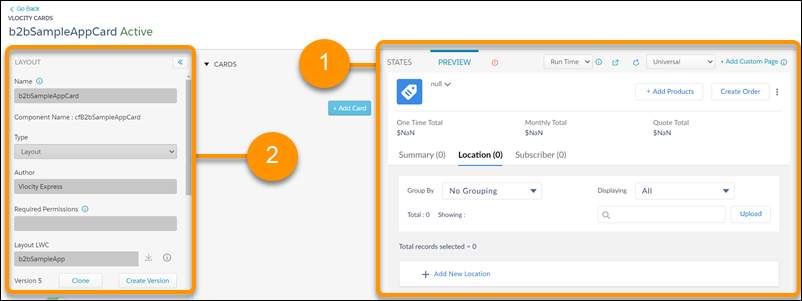
The b2bSampleAppCard  
The b2bSampleAppCard is probably the most important Card in ESM. This Card allows you to conditionally render application components, based on the route and context of the user.

To display changes you’ve made in your ESM user interface, you clone and modify the b2bSampleAppCard. For example, when you completed the Lightning web components module, the final step in making your amendments was to ensure that the b2bSampleAppCard was cloned and amended to display your new custom Lightning web component, then activated.

To access the b2bSampleAppCard, go to the App Launcher, search for *Cards*and select **Vlocity Cards** to view the Cards used by ESM.



Open the Card to take a closer look. Pick the active version of the Card and click on the **PREVIEW**tab to preview the card (1).



Scroll down through the Card layout details (2). The b2bSampleAppCard holds not only the details of associated Lightning web components, but also the session variables used for passing input to ESM’s Lightning web components.

What are ESM’s Session Variables?  
Session variables contain data that is maintained across web pages. ESM uses session variables to pass input from the Cards to Lightning web components for fetching data, or for table column configuration. For example, the b2bCategoriesList passes the catalog category selected by the user so that ESM can return the list of offers in that category.

 Recap: What are OmniScripts?  
OmniScripts are used to guide users through sales and service processes with fast, personalized responses, and seamless integration to enterprise applications and data.

OmniScripts Specific to ESM  
There are seven OmniScripts specific to ESM available out of the box which you may want to customize to suit your business requirements.

**b2bExpress/CreateQuote**

Guides the user through creating a quote by choosing the account, opportunity, and price list.

This OmniScript is triggered when the user clicks **Create Quote**, which is found on the Account page and the Opportunity page.

**b2bExpress/generateProposalDocument**

Guides the user through generating a proposal from a quote.

This OmniScript is triggered when the user clicks a custom **Create Proposal** button in ESM.

**b2bExpress/generateProposalDocumentLWC**

Guides the user through generating a proposal from a quote using LWCs.

This OmniScript is triggered when the user clicks a custom **Create Proposal** button from ESM.

**ESM/QuoteEnrichForAll**

Called when **All**is selected in **Displaying** on the Quote Summary page.

This OmniScript is used to update the billing details, connect date, and phone number for all the selected line items in the quote.

**ESM/QuoteEnrichForInstallation**

Called when **Installation**is selected in **Displaying**on the Quote Summary page.

This OmniScript is used to update the connect date for all the selected line items in the quote.

**ESM/QuoteEnrichForNumber Assignment**

Called when **Number Assignment** is selected in **Displaying**on the Quote Summary page.

This OmniScript is used to update the phone numbers for all the selected line items in the quote.

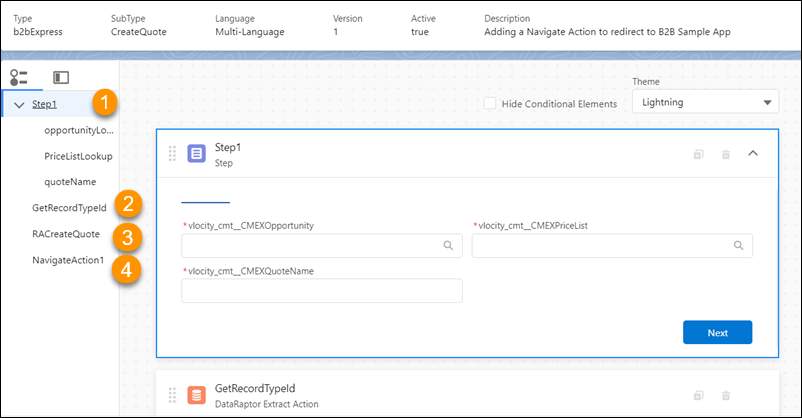
**ESM/QuoteEnrichForPayment**

Called when **Payment**is selected in **Displaying**on the Quote Summary page.

This OmniScript is used to update the billing account associated with all the selected line items in the quote.

You can also create your own OmniScripts to suit your business requirements.

Focus: The CreateQuote OmniScript  
Let’s take a closer look at the CreateQuote OmniScript.

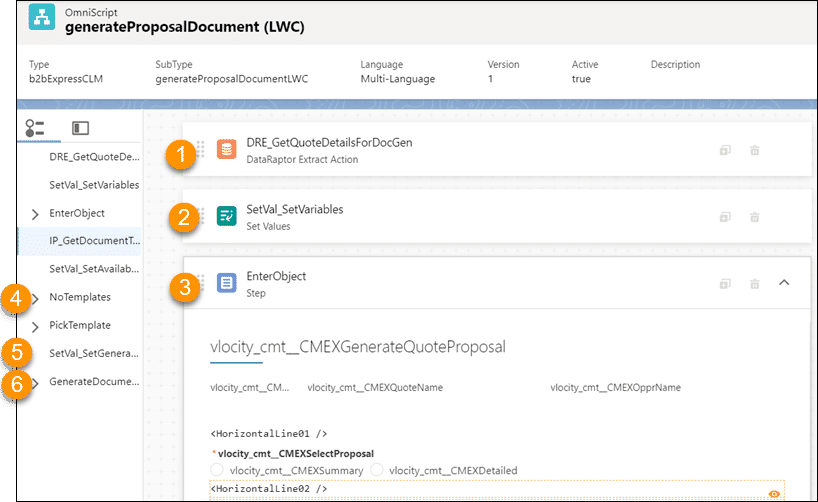


There are four parts to this OmniScript:

1. **Step1** collects information from the user relating to the associated opportunity, the price list to be used, and what they would like the quote to be called. This provides the initial page you see when you click **Create an Enterprise Quote**.
2. The second element is a DataRaptor extract action called **GetRecordTypeID**. This interfaces with a DataRaptor called GetRecordTypeId, which outputs an ID for a new “Quote” object.
3. Next is a remote action called **RACreateQuote**, which uses the **createCart**remote method from the CpqAppHandler class to create a quote. The Account ID is passed as the ContextID. The extra payload key-value pairs associated with this method are:
   1. fields: Id, Name
   2. inputFields:(an array consisting of opportunityid, Name, RecordTypeId, and pricelistname)
   3. methodName:createCart
   4. objectType:Quote
   5. subaction:createQuote
4. The final step is a navigate action to take you to the newly created quote.

Examples of process customizations performed using the CreateQuote OmniScript include automating the creation of opportunities as part of quote creation, and adding extra information to the quote, such as available budget, when it’s created.

Focus: The generateProposalDocument(LWC) OmniScript  
The OmniScript to generate a proposal for the current quote is triggered when the user selects “Create Proposal” from the quote action menu, or clicks on the custom Create Proposal action button. As you can see, it is slightly more complex than the CreateQuote OmniScript!



1. The first element in the OmniScript is a **DataRaptor Extract** action, which uses the GetQuoteDetailsDocGen DataRaptor to get the quote data, where ContextId=QuoteId.
2. The **SetVal\_SetVariables** Set Values element sets the TemplateCategory to be used by CLM to “Proposal”, and the Template Type to a Word.docx template.
3. The user is then asked to select the type of proposal they would like (Summary or Detailed), and the level of aggregation for the proposal (by Members or by Groups). This information is used by an integration procedure action called IP\_GetDocumentTemplates to fetch the appropriate template for the proposal.
4. If the template option selected by the user is not available, the **NoTemplates** step is shown, and the user may return to select another option. If there is more than one template applicable for the options selected by the user, the **PickTemplate**step provides the user with a dropdown list of templates to choose from.
5. A Set Values element called **SetVal\_SetGenerationOptions**sets all the options for document generation, such as the ContextId, file format, and selected template.
6. The final step, called **GenerateDocument**, generates the proposal and displays it to the user.

How do I Customize, Extend, or Create New OmniScripts?  
A detailed lesson on OmniScripts is already provided in the OmniStudio trail in Trailhead, so we won’t be covering it here. However, you can practice ESM-specific amendments by altering the CreateQuote OmniScript in the practice exercise below.

What do I Need to Integrate? - and How?  
ESM provides an out-of-the-box solution for large quotes and orders, but think of this as a starting point for your sales teams and customize it to suit your business. You’ve already learned how to customize the look and feel of ESM, and the guided processes, but what about data and integration with your other systems?

ESM integrates with Salesforce’s Customer 360 Cloud, which provides a solution for your marketing, sales, commerce, service, and IT departments. However, you may have other systems you need to integrate with ESM to support the quote and order process. These systems may include back-office systems, such as billing, asset, field services, and outage management.

The good news? ESM gives you out-of-the-box integrations and processes, and tools to build your own custom integrations. You use these tools to quickly create integration and configuration. Even better, add customizations and integrations incrementally as and when your ESM solution needs them.

In the previous lessons, you learned about customizing ESM using declarative tools such as OmniScripts, Cards, and Layouts. Combined with LWCs, these tools give you faster user experience configuration and connected process development. One of the great features of these tools is that they allow customers to skip from one channel to another without losing their input. For example, your customer can start an order on the self-service website, and then call the contact center to finish the order, with all the information they provided earlier accessible in all channels, at every point.

In this module you’ll learn about implementing the out-of-the-box processes and integrations provided with ESM. You’ll also extend ESM’s integration, to pass data into and out of the OmniScripts, Cards, and LWCs you learned about earlier, using DataRaptors, Integration Procedures, and APIs.

Process Flows and Integration Available Out of the Box  
Before you create customizations, ensure that your requirements aren’t already met by out-of-the-box ESM capabilities. You should also check the ESM roadmap for upcoming features that may support your business needs. Here are some existing process flows and integrations included out of the box as part of the ESM managed package. The out-of-the-box integrations are covered in detail in the next lesson, but let’s take a quick look now at what’s available when you first install ESM.

**Customer-Related Processes**

These include creating and amending accounts, importing and amending subscribers and locations, changing plans, and move in/move out guided processes. Customizable ESM-specific guided processes were discussed in detail in the OmniScripts module of this course.

**Pricing Calculation Procedures**

There are two ESM-specific types of out-of-the-box customization related to pricing: frame agreements and calculation procedures and matrices.

To implement frame agreement pricing, an additional pricing step is added to the pricing plan steps in ESM to retrieve the best price for the offer in a related frame agreement. You can learn more about this in the Contract Lifecycle Management documentation linked in the Resources for this lesson.

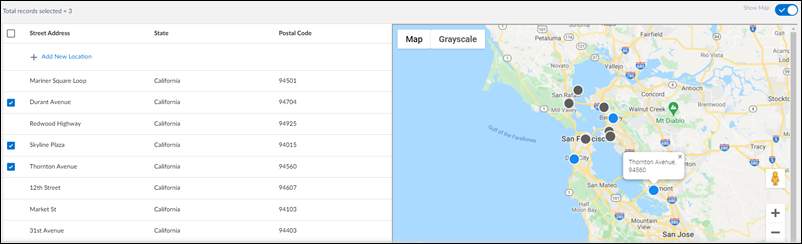
ESM product models make use of calculation procedures and calculation matrices to implement attribute-based pricing. This is a standard feature of Industries CPQ, and is explained in detail in the Industries CPQ Advanced Pricing module, and in the attribute-based pricing documentation, linked in the resources for this lesson. To implement attribute-based pricing, you create a pricing step in the default pricing plan to call a calculation procedure and calculation matrix. The calculation matrix holds information about the products, associated attribute values, and pricing. The ESM calculation matrices available out of the box are:

* **AttributePricingMatrix**, which works with the AttributePricingProcedure calculation procedure to determine pricing for a product based on attributes. Attributes specified depend on the nature of the product, and include details such as the contract term, handset storage capacity, number of security licenses required, and the number of static IP addresses required.
* **B2BQuantityBasedDiscounts** matrix works with the B2BQuantityBasedDiscountsProcedure calculation procedure to apply product-specific volume-based discounts.
* **L2WAN\_AttributePricingMatrix** works with the L2WAN\_AttributePricingProcedure to price WAN products. The matrix uses the EVC endpoint bandwidth and EVC endpoint class of service to determine the price of the EVC endpoint.
* **RangeAttributePricingMatrix** is a sample matrix you can use with the RangeAttributePricingProcedure calculation procedure to create your own range-based pricing. A completed example of this matrix, called VolumeRangeAttributePricingMatrix, is included in your training playground.  
  SourceTargetAttributePricingMatrix is a sample matrix you can use with the SourceTargetAttributePricingProcedure, to specify the price of a target product based on attributes specified on a source product.

At runtime, the PricingPlanService implementation passes the Cart line items, including product information and their attributes and fields, to the calculation service and retrieves the correct price. The price returned overrides the base price in the shared catalog. Additional adjustments and calculations can be applied before the price is displayed in the ESM Cart.

**Google Maps Integration**

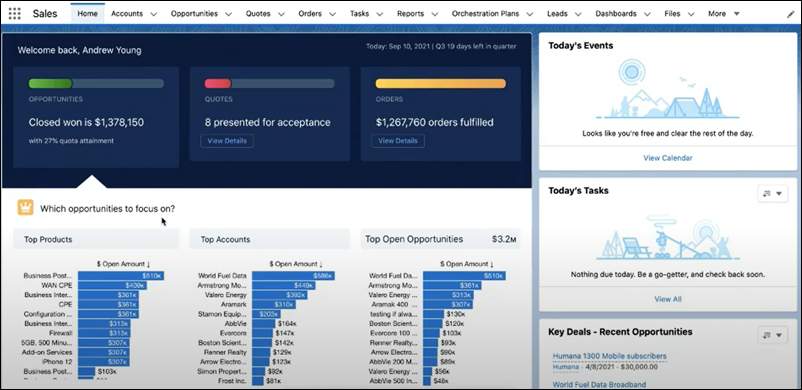
ESM uses Google Maps APIs to find and select locations on a map. To enable this functionality, generate the required Google Maps API key in the Google Maps platform, and then add this key to your org. The key is used to validate Google Maps API calls. There is a guided exercise to help you implement this integration in the next lesson.



**Tableau CRM Integration**

ESM integrates out of the box with Salesforce’s analytics application, Tableau CRM, to provide insights on your sales data and help your sales team prioritize their workloads.

Tableau CRM is a separate license that may be purchased by your organization. There is a guided exercise to help you implement this integration in the next lesson.



Integration Tools  
Before continuing with this course, we recommend you complete the OmniStudio Trail on Trailhead, which includes lessons on working with custom integration tools, including DataRaptors, integration procedures, and the SDK. In summary, you’d use each of these, but for different purposes.

* DataRaptors read, transform, or write data. They typically supply Salesforce data to OmniScripts, Integration Procedures, and Cards, but can read external data.
* Integration Procedures execute multiple actions in a single server call. They can be called from an OmniScript, API, or Apex method and can be a data source for an ESM Card.

But where are the APIs? Don’t worry - they’re there! It’s just best that you don’t use them on their own. Instead, Salesforce Industries recommends you call the ESM APIs through the ESM SDK. This approach improves usability, and you can ensure that errors are detected before API calls are made, so the ESM data structure rules are followed.

What's Available?  
There are two out-of-the-box integrations provided in ESM that are extremely helpful for your sales team. The first is integration with a Google Maps API, so your sales team can select locations interactively from a map when creating quotes and orders. The other is integration with Tableau CRM, which collects and displays opportunity, quote, order, and account analytics within the ESM interface.

In this lesson, you will learn how to gain access to each integration, and how to configure them for your business, before creating your own integrations using the practice exercises.

Click **Start**on each process flow to find out more.

**ESM Integration with Google Maps**

ESM has out-of-the-box integration with Google Maps, which makes it easy for users to select and group subscribers and locations. This can really speed up the quote process, as you’ll see in this demonstration.

To enable Google Maps integration, you’ll need your Google Maps API key, a 39-character hexadecimal code.

**Get Your Google Maps API Key**

Google Maps APIs are charged per use, so you should ask your IT Department to supply you with a Google Maps API Key to allow you to use your company’s Google API license for the map integration.

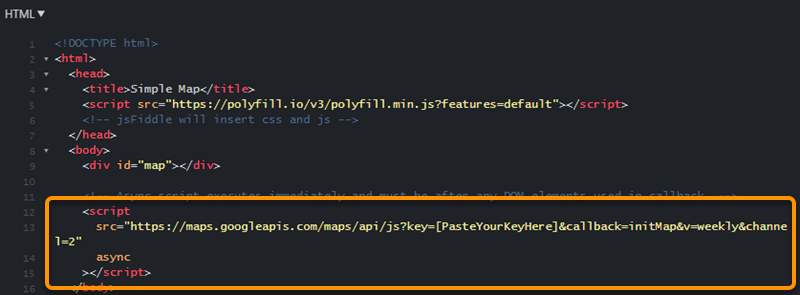
If you don’t have a Google Maps API key, and want to set one up for your own personal use, you can follow the steps described on the Google Maps Platform site, linked in the Resources for this lesson.

|  |  |
| --- | --- |
| 📝 | You will be charged for API use. If you are following these steps for training purposes only, and use a personal account, we recommend you remove your APIs once you are done. |

In summary, here are the steps:

1. Set up a Google account, or log into your business or personal Google account if you already have one.
2. Set up a billing account for your Google account. You’ll need a credit card for this. Remember, you’ll be charged for anyone who uses your API, so if you’re using a personal card it would be best if you remove it once you’ve tried out the integration exercises.
3. Set up a project.
4. Enable these Google Maps APIs:
   1. Geocoding API
   2. Maps JavaScript API
5. Create the API Key. If you click on **Credentials**in the Navigator on the left of the Google Maps Platform page, you’ll see all the API keys available to you. Copy the Google Maps key somewhere safe that you can find later, as you will be testing it, then adding it to ESM next.

**Test Your Google Maps API Key**

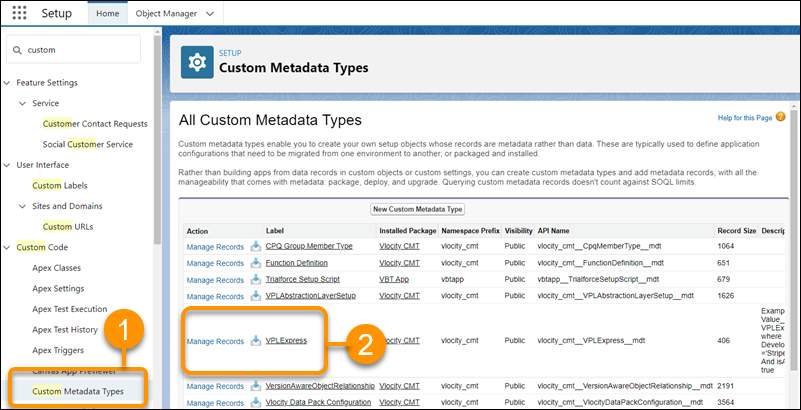


Go to the **Simple Map** sample on the Google Maps Platform documentation. There is a link to this page in the resources for this lesson. The Simple Map sample shows you how the map works and how you set up your site to use it.

1. Under Try Sample, click JSFiddle.
2. Paste your key into the appropriate section of the HTML code. We’ve highlighted it here and tagged it with [PasteYourKeyHere] so you can see where it goes.
3. Click ⊳ Run at the top of the screen to run the API with your API key. You should see a functioning map in the lower right of the page.

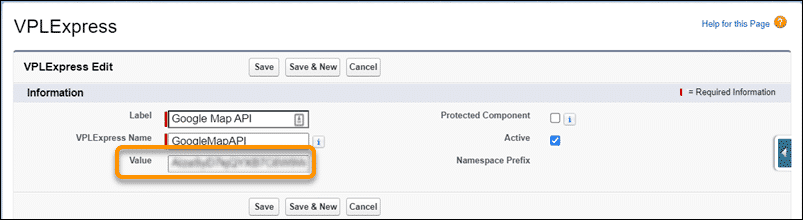
Now you’re ready to put your API key into ESM.

**Open Salesforce Custom Metadata Types**



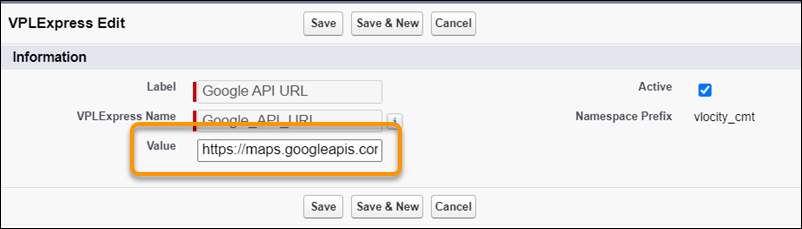
1. In Salesforce, go to **Settings**and click **Setup**.
2. In **Quick Find**, type *custom*and click **Custom Metadata Types** (1).  
   Find VPLExpress in the list of Custom Metadata Types (2) and click Manage Records.

**Plug Your Google Maps API Key into ESM**



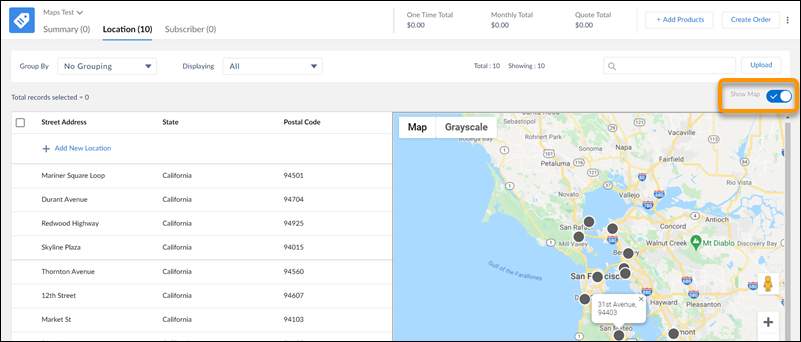
1. Click **Edit**to the left of the **Google Map API**.
2. Paste your API key into the **Value**field and save.
3. Repeat this process for the **Google API Public Key** custom metadata type, entering your Google Maps API key into the value.

**Add the Google API URL**



1. Click **Edit**to the left of the **Google API URL**.
2. Type the link to the map site in the **Value**field:  
   https://maps.googleapis.com/maps/api/js
3. Click **Save**to save your changes.

**Test Your Integration**



Open and configure an enterprise quote that has at least one location allocated.

Select **Show Map.**

A Google Map indicating the locations associated with the quote will be displayed.

**Troubleshooting the Google Maps Integration**

If your map doesn’t work when you switch it on, first check that you have locations with legitimate addresses in your quote. Once this is done, try these troubleshooting steps.

* Check you have an API key in your Google API Public Key. You add this the same as you did the Google Map API key. If you don’t have a general Google key, use your Google Maps API key.
* Check you have the correct URL for Google APIs in the Google API URL value.
* Check there are no spaces in your Google Map API URL value.
* Check you have enabled both the Geocoding API and the Maps JavaScript API in the Google Platform.
* Check the API key is created and not restricted.

Recap: What are DataRaptors?  
DataRaptors are mapping tools that you use to read or write Salesforce sObject data or to perform single-step data transformations. There are four types of DataRaptor:

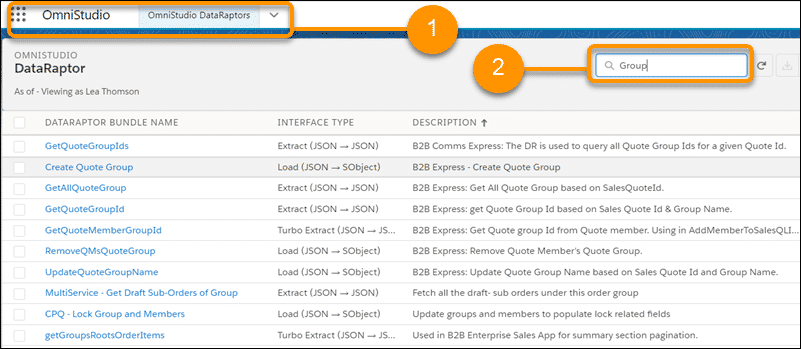
* Turbo Extract DataRaptors read data from a single Salesforce object type, with support for fields from related objects. You can then select the fields to include. Formulas and complex field mappings aren’t supported.
* Extract DataRaptors read data from Salesforce objects and output JSON or XML with complex field mappings. Formulas are supported.
* Transform DataRaptors perform intermediate data transformations without reading from or writing to Salesforce. Formulas are supported.
* Load DataRaptors update Salesforce data from JSON or XML input. Formulas are supported.

A Reminder: DataRaptor Best Practices  
DataRaptors are great - but to get the most out of them try to follow these best practices:

* DataRaptors should extract or load the data needed for just one operation.
* Use relationship notation (queries) where possible to pull data from other sObjects.
* Try to keep the sObjects to fewer than four.
* Filtering and sorting should be restricted to indexed fields only, such as the Id and Name fields.
* Use caching to store frequently accessed, infrequently updated data.

ESM-Specific DataRaptors  
To find out more about DataRaptors used by ESM, and those that have been built specifically for ESM, click on the **App Launcher** and select **OmniStudio**, then select **OmniStudio DataRaptors** from the tab dropdown (1).

You can see the name of the DataRaptor, the type, and a brief description of its functionality.



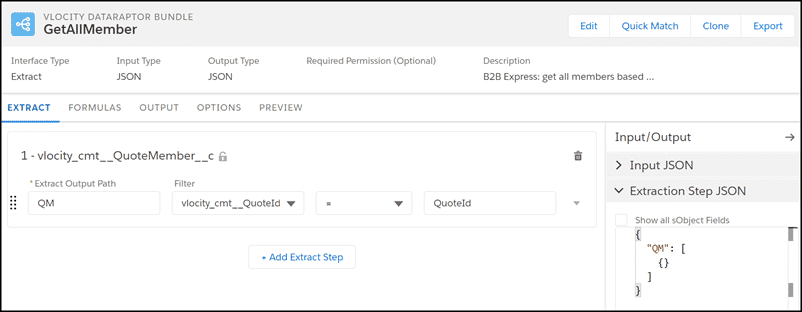
You can filter your list, or you can quickly search for what you need by typing part of the DataRaptor name in the search box (2). Click on the DataRaptor name to open it and view more detail if required.

Let’s look at two DataRaptors you’ll probably want to extend as part of your customization process.

Focus: The GetAllMember DataRaptor  
The GetAllMember DataRaptor is an Extract type DataRaptor, which fetches all quote members’ details for a specified quote ID.

Open it up from your OmniStudio DataRaptor list and take a look.

Click on the **Extract**tab to see where the DataRaptor is looking for the data, and how it is filtered.

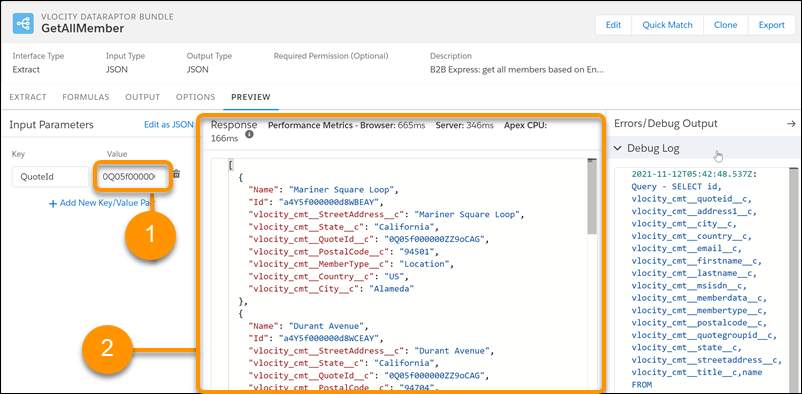


Here you can see it is accessing a custom ESM object called **vlocity\_cmt\_\_QuoteMember**.

This object is a record containing fields for both locations and subscribers. You can view detailed information about the vlocity\_cmt\_\_QuoteMember object, including all the associated fields and relationships, in Salesforce’s Object Manager.

There are no formulas or transformations in the GetAllMember DataRaptor.

Go to the **PREVIEW**tab in the DataRaptor to test the output.



* Copy the unique quote ID from an existing quote, and paste it into the **QuoteID** Value for **Input Parameters.**
* Click **Execute**to view the response.

The quote members associated with your specified quote are shown in the output pane. In this example, the quote members are locations, including Mariner Square Loop and Durant Avenue.

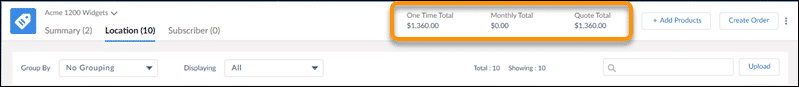
Why is this DataRaptor important? If you want to customize your quote members, for example, to include extra fields, you will add the fields to your Quote Member object, your Cards, and your associated OmniScripts and LWCs. You’ll then extend this DataRaptor to extract the new fields from your object to add to your Card.

You would also amend the corresponding DataRaptor Load, probably with an Integration Procedure, to add and amend the new fields for your quote members.

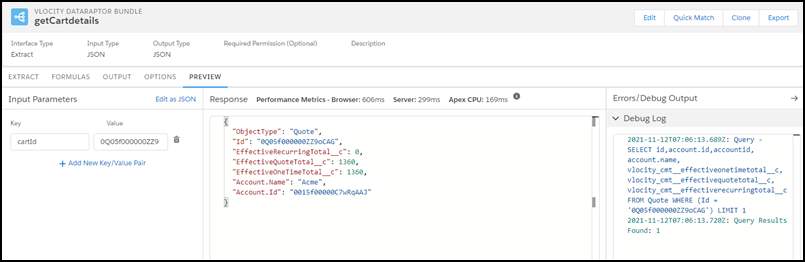
Focus: The getCartdetails DataRaptor  
You use this DataRaptor to fetch the following information from the Cart object, using the current Cart Id as the filter:

* Account Id (Account.Id)
* Account Name (Account.Name)
* One-Time Charges Total (EffectiveOneTimeTotal\_\_c)
* Recurring Charges Total (EffectiveRecurringTotal\_\_c)
* Quote Total (EffectiveQuoteTotal\_\_c)
* Cart Id (Id)

This information is then displayed in the header of the quote.



Here’s an example response based on an existing cartID.



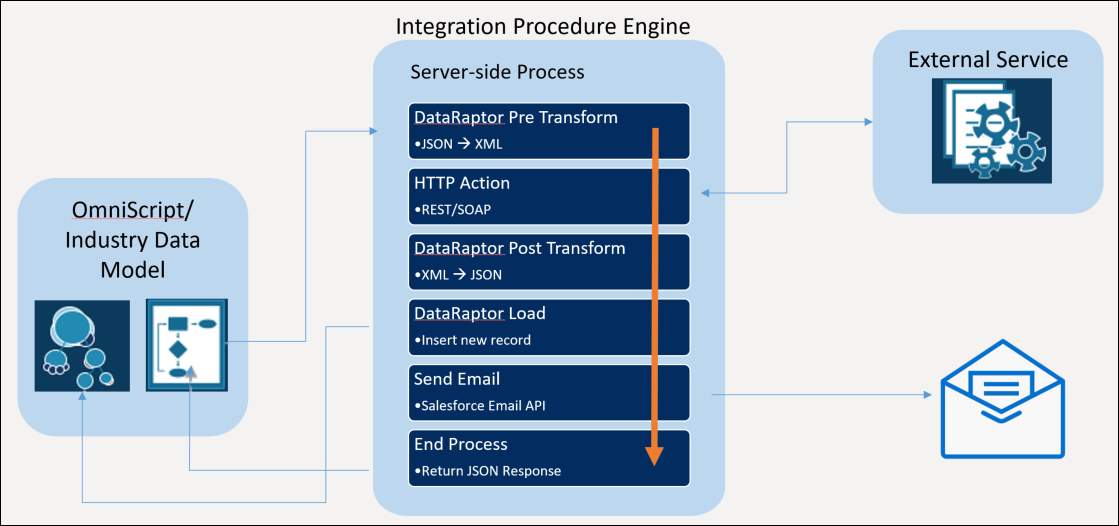
Here the quote accessed is for Acme, with a recurring charge total of $0, a one-time total of $1,360 and a total quote charge of $1,360.

You would extend this DataRaptor if, for example, you want to display a profit margin for the quote or the total cost and the charges in the quote header.

Recap: DataRaptors, APIs, and Integration Procedures  
You’ve learned that DataRaptors, APIs, and Integration Procedures work together to integrate ESM with other applications, but which should be used where?

DataRaptors read or write Salesforce sObject data or perform single-step data structure transformations.

In contrast, Integration Procedures (VIPs) interact with many types of data, including REST APIs and Apex classes, and process it in multiple steps. Here’s how OmniScripts, VIPs, DataRaptors, and APIs work together.



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. VIPs may use one or more DataRaptors to complete their tasks.

In summary, VIPs are best to use when you need to access and transform data from third-party sources and no user interaction is required, and when you want to move the workload from the client to the server.

VIPs can do some things that OmniScripts can't, the most important of which is list processing with Loop Blocks and List Actions. VIPs can perform more data processing steps than DataRaptors can, and they're more flexible than Calculation Procedures.

Integration Procedures Specific to ESM  
In the current release, there are more than 30 ESM-specific VIPs available out of the box. You can read more about each of these VIPs, including the required inputs, outputs, and how each is invoked, using the Integration Procedures link in the Resources for this lesson.

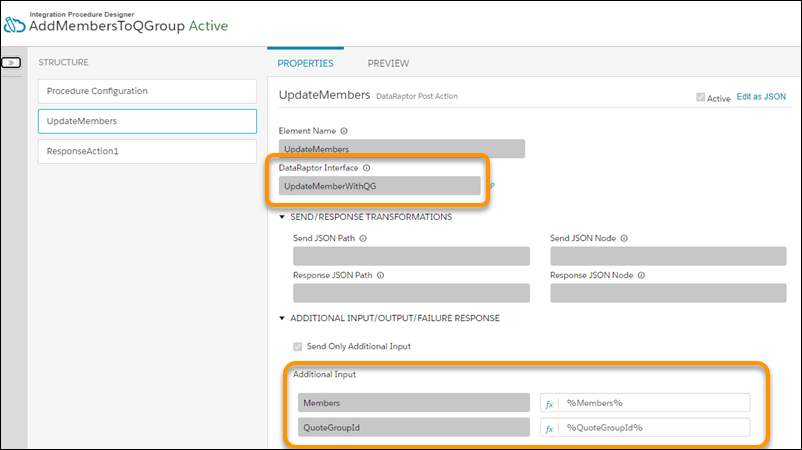
Notice that many of the VIPs are called by the b2bDataTable Lightning web component. This is a Lightning web component specific to ESM which renders data uploaded for locations or subscribers, grouping, filters, searches, sorting, and pagination. Let’s look at the impacts of this in a bit more detail.

ADDMEMBERSTOQGROUP

When you add a member to a quote group in a quote, this involves the b2bDataTable Lightning web component, which calls the AddMem\_ToQG Integration Procedure.

The AddMem\_ToQG Integration Procedure passes the member Id, and the selected quote group Id to a DataRaptor Load called UpdateMemberWithQG, which adds a new quote member with the associated details to the vlocity\_cmt\_\_QuoteMember\_\_c object.

Once this is done, the Integration Procedure provides a response action to confirm the quote member has been added.



SUBMITSUBORDER

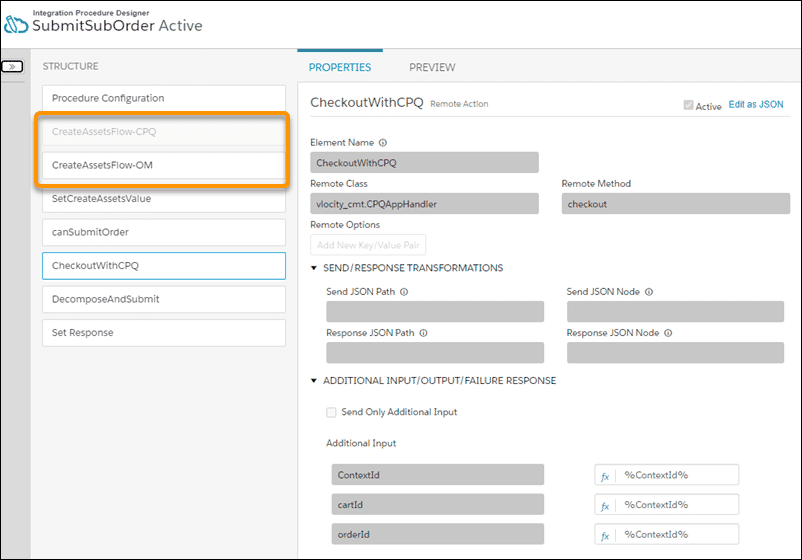
**SubmitSubOrder**is a multi-step Integration Procedure used to submit an order.

It is activated from a procedure called **B2BmexSubmitSubOrderProcName**, accessed in the CPQ Configuration setup.

If you want to add extra validation steps into your order submission process, you would edit this Integration Procedure.

The key steps involved in the SubmitSubOrder are:

* **CreateAssetsFlow**: here you select how your orders are submitted. For example, you can enable either the CPQ or OM order flow by enabling or disabling the CreateAssetFlow-CPQ. In this example, the CPQ flow is disabled, so the CreateAssetsFlow is managed by Order Management. It is possible to have both flows enabled.
* **SetCreateAssetsValue** - this uses a function to decide how to create the assets based on the CreateAssetFlow.
* **CanSubmitOrder**: this passes the CartId as the ContextId to a remote method called canSubmitOrder to determine whether or not the order is valid for submission.
* **CheckoutwithCPQ**: this uses a formula to determine whether or not the CreateAssetsFlow is handled by CPQ, then if it is true, runs the appropriate methods from the CPQAppHandler to create the assets.
* **DecomposeAndSubmit**: this uses a formula to determine whether or not the CreateAssetsFlow is handled by OM, then if true triggers the OM Decomposition Controller.
* **SetResponse**: this uses output from OM or CPQ to provide a feedback message to the user on whether or not their sub-order can be submitted.



ESM APIs  
You use the ESM APIs to create and manage enterprise quotes by calling corresponding Integration Procedures.

It’s best practice to use the ESM SDK and Lightning web components to call these APIs and modify the UI. The SDK and Lightning web components improve usability and reduce the amount of work you have to do on your UIs, by hiding the complex API semantics inside the SDK interface.

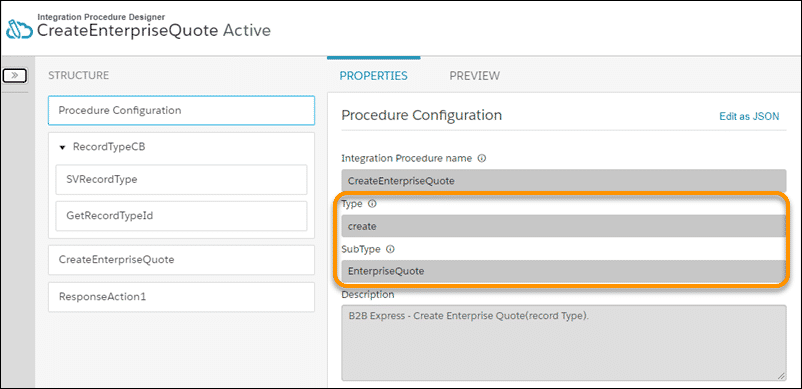
There are more than 30 APIs specifically designed for ESM. You can read about them in detail in the documentation, which is linked in the Resources section of this lesson. For each API, the documentation describes:

* The function of the API, including its HTTP method.
* The associated Integration Procedure.
* The URI used to call the API.
* An example request.
* An example response.

How to Post Data to an Integration Procedure  
To post JSON data to an ESM Integration Procedure, issue a POST call using a URL formatted like this:

/services/apexrest/vlocity\_cmt/v1/integrationprocedure/{type}\_{subtype}/

Specify the type and subtype of the Integration Procedure, which can be found in its procedure configuration.



To pass input data in a POST call, you can specify a JSON request body.

In this example POST, the CreateEnterpriseQuote Integration Procedure creates an enterprise quote. It requires an account ID, an opportunity ID, and the name of a price list. It returns the ID of the newly created enterprise quote.

POST

/services/apexrest/vlocity\_cmt/v1/integrationprocedure/create\_EnterpriseQuote/

Example POST JSON Data

{

"AccountId": "0015w00002DWyJg",

"OpportunityId": "0065w000022LuEJ",

"PriceListName": "2020 Business"

}

Example Result

{

"EnterpriseQuoteId": "0Q05w000001xmPjCAI"

}