CPQ Cloud

Integrating Oracle Commerce Cloud Service and CPQ Cloud Service

ORACLE IMPLEMENTATION GUIDE

ORACLE

Table of Contents

Introduction

Self-service users in Commerce Cloud can configure complex products for purchase in Commerce Cloud using the CPQ Cloud configurator. They can also request a CPQ Cloud quote, thereby initiating a CPQ Cloud Transaction a sales specialist can modify, reconfigure, or discount. Once finalized in CPQ Cloud, the quote returns to Commerce Cloud for acceptance and ordering by the self-service user. For additional information, refer to Appendix A: Configurator Flow and Appendix B: Request for Quote Flow.

NOTES:

- The integration of Commerce Cloud with CPQ Cloud uses the Oracle Integration Cloud Service (OIC) to provide pre-built integrations for the two user flows.

Purpose

The purpose of this implementation guide is to provide the steps that administrators must complete in CPQ Cloud, OIC, and Commerce Cloud to prepare for an Oracle Commerce Cloud Service and CPQ Cloud Service integration.

Audience

This implementation guide is for administrators who are setting up and configuring the integration. The guide assumes administrators have prior Commerce Cloud, CPQ Cloud, and OIC administration experience.

Prerequisites

The following is a list of integration prerequisites:

- A Commerce Cloud 18C-MP or later site setup as described in this implementation guide.
- A CPQ Cloud 18C or later Base Ref App site set up as described in this implementation guide. The integration between Commerce Cloud and CPQ Cloud adds attributes to the Base Ref App site that correspond to required Commerce Cloud order data.
- A synchronized product catalog to ensure that products in the Commerce Cloud catalog map to corresponding items in the CPQ Cloud catalog.
- Oracle Integration Cloud Service (OIC) 18.3.5 or later.

NOTE: For information about how to obtain any of the above prerequisites, contact an Oracle sales representative.

Set Up CPQ Cloud

This section contains the general, configuration, and Commerce steps administrators must complete in CPQ Cloud.

General Set Up

Administrators must complete the following general set up procedures:

- Enable Guest Access to CPQ Cloud
- Add Template Dependencies to File Manger
- Make CPQ Cloud Stylesheet Edits
- Synchronize CPQ Cloud Parts with Commerce Cloud SKUs

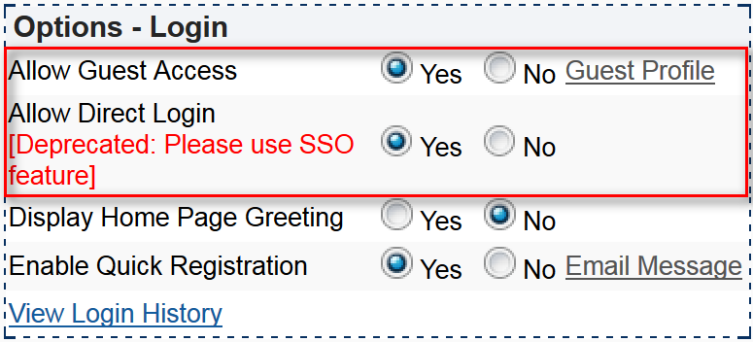
Enable Guest Access to CPQ Cloud

Administrators can allow multiple self-service users in Commerce Cloud to access a CPQ Cloud site as a guest user from an iFrame displaying within Commerce Cloud. When Commerce Cloud punches in to CPQ Cloud for configuring items, the system uses sessions for unregistered users (i.e. guest users). When self-service users access a CPQ site, their session parameters pass from Commerce Cloud to CPQ Cloud. This provides a seamless user experience and eliminates the need for Commerce Cloud self-service users to enter login credentials when entering a CPQ Cloud site from Commerce Cloud.

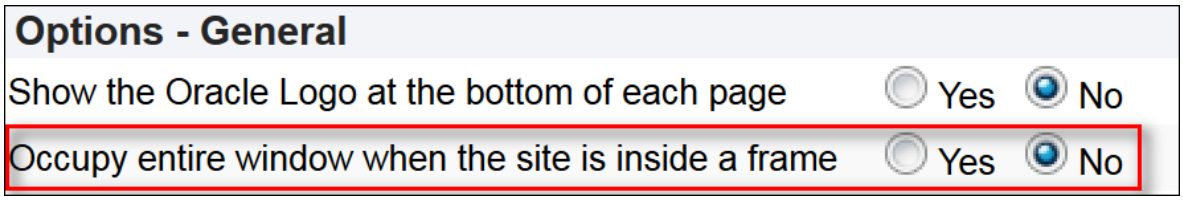
Session parameters include currency, language, and locale preferences such as number format, units, and date format. For example: If a Commerce Cloud self-service user's language preference is set to German, the text in the CPQ Cloud interface displays in German when the user accesses CPQ Cloud. The user's currency and locale preferences are also passed from Commerce Cloud and display in CPQ Cloud.

To enable guest access to CPQ Cloud:

1. Open the Admin Home page.
2. Under **General**, select **General Site Options**.
   The **Options – General** page opens.
3. Under **Options – Login**, set **Allow Guest Access** to **Yes**.
   This setting allows Commerce Cloud to punch in to CPQ Cloud.
4. If multi-currency support from Commerce Cloud is required, set **Allow Direct Login [Deprecated: Please use SSO feature]** to **Yes.**



5. Under **Options – General**, set **Occupy entire window when the site is inside a frame** to **No**. This setting improves usability when punching in to CPQ Cloud from Commerce Cloud.



Add Template Dependencies to File Manager

The “Add to Cart” action sends items to a Commerce Cloud cart via an **Add to Cart** button, which displays on the Commerce Cloud integrated CPQ Cloud site following configuration. Use the information provided in this section to add payload template files to File Manager. If Commerce Cloud requires additional information from CPQ Cloud during the “Add to Cart” action, administrators can add the information by creating configurable attributes and modifying the payload templates. Administrators can then export the configurable attributes as key-value pairs from CPQ Cloud to Commerce Cloud.

Payload template files (i.e. Recommended_Items_Payload-Cloud.txt and AddToCartPayload-Cloud.txt) form the payload structure for sending a configured item to the Commerce Cloud shopping cart. The template files support the “Add to Cart” action and include configuration information such as config id, quantity, and BOM items. BML reads the template files and replaces the values in brackets, such as {{bomitems}}, with dynamic values.

Complete the following steps to add the payload template files to File Manager:

1. Open the Admin Home page.
2. Navigate to **Utilities** > **File Manager**.
   File Manager opens.
3. Create a new folder named **CommerceCloud**.
4. Under **Add Files**, click **Browse.**
   The **Choose File to Upload** dialog opens.
5. Navigate to the **Recommended_Items_Payload-Cloud.txt** file and click **Open**.
6. Click **Add File**.
   The **Recommended_Items_Payload-Cloud.txt** file displays in **File Manager**.

7. Complete steps 1-6 for **AddToCartPayload-Cloud.txt**.

Shown below is the content of each of the payload template files.

Recommended_Items_Payload-Cloud.txt

```
{ "quantity": "{{quantity}}", "catalogRefId": "{{part}}", "price":
"{{price}}", "recurringCharge": { "amount":"{{recurringPrice}}",
"frequency":"{{pricePeriod}}", "duration":"{{duration}}" } }
```

AddToCartPayload-Cloud.txt

```
{ "messageType": "Configuration_Details", "quantity": "1", "catalogRefId":
"{{model}}", "amount": "{{totalPrice}}", "price": "{{basePrice}}",
"currencyCode": "{{currency}}", "configurationId": "{{ConfigId}}",
"childItems": [{{ChildItems}}], "bomItems": [{{BomItems}}]}
```

Make CPQ Cloud Stylesheet Edits

Oracle recommends administrators hide CPQ Cloud navigation options outside the scope of the integration from Commerce Cloud self-service users.

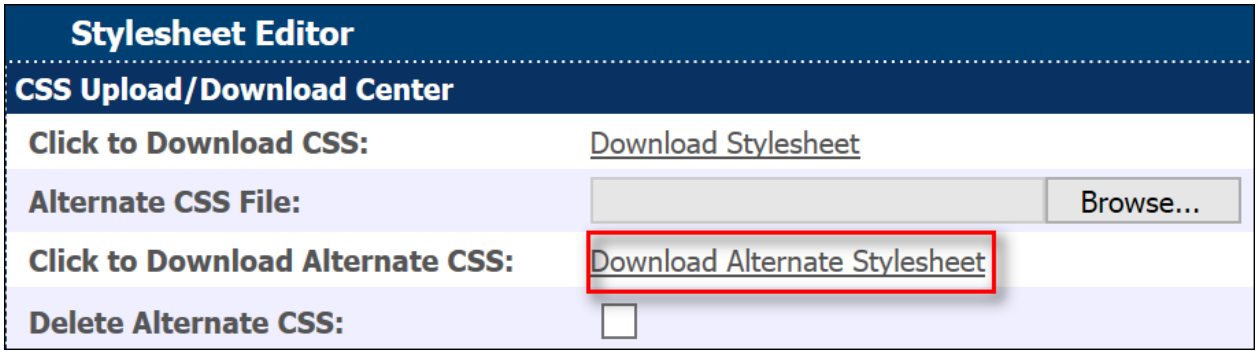
Hide the CPQ Cloud Home Button

By hiding the CPQ Cloud Home button, the CPQ Cloud configurator opens whenever users access CPQ Cloud. Users cannot navigate away from the original model that opens in the configurator, which prevents them from configuring a different model or adding a different model to Commerce Cloud.

To hide the CPQ Cloud Home button:

1. Open the Admin Home page.
2. Under **General**, select **Navigation Menus**.
   The **Navigation Menus** page opens.
3. Click **Edit Stylesheet**.
   The **Stylesheet Editor** opens.

4. Select **Download Alternate Stylesheet**.



5. When the alternate CSS file opens, update the CSS to include the following CSS snippet to hide the **Home** button within the iFrame.

```
.nav-links>a img[title="Home"]{
    display: none;
}
```

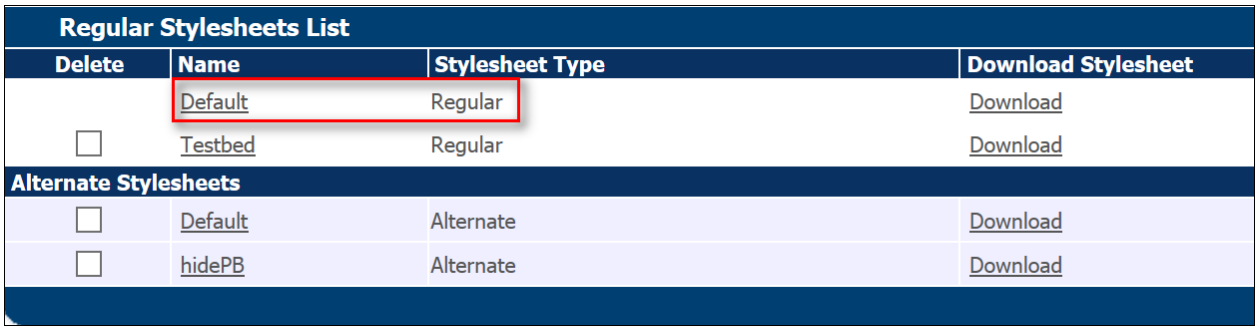
IMPORTANT: If the **Home** button shows both a label and an icon, administrators cannot hide the label using only CSS. From the Admin Home page, navigate to **Style and Templates** > **Navigation Menus** > **Subheader** > **Home** > **Edit**. Choose **Icon** for **Display**. The **Home** button is then hidden with the CSS change.

Hide Price Books

CPQ Cloud uses Price Books as a way to associate parts with a price. Oracle recommends hiding Price Book information from users.

To hide Price Books:

1. Open the Admin Home page.
2. Under **Products**, select **Catalog Definition.**
   The **Supported Products** page opens.
3. From the **Navigation** drop-down menu, select **Stylesheets**.
4. Click **List**.
   The **Regular Stylesheets List** page opens.
5. Download the **Default Regular Stylesheet**.



6. Copy the contents of the **Default Regular Stylesheet**.
7. Create a new stylesheet with a name indicative of the stylesheet's purpose.
   For example: Hide Price Books
8. Paste the contents of the **Default Regular Stylesheet** into the new stylesheet and add the following CSS:

```
.pricebook-container {
          display: none;
  }
```

9. Save the stylesheet.
10. On the **Regular Stylesheets List** page, click **Add Alternate**.
    The **Configuration Stylesheet Editor** opens.
11. Click **Browse**.
12. Use the **File Upload** dialog to locate and select the new stylesheet.

13. Click **Open**.
    The stylesheet displays in the **Regular Stylesheets List** page under the list of **Alternate Stylesheets**.

Synchronize CPQ Cloud Parts with Commerce Cloud SKUs

In Commerce Cloud, SKUs represent a purchasable instance of a product on a Commerce Cloud storefront. Administrators must synchronize CPQ Cloud parts with Commerce Cloud SKUs to ensure the pricing information associated with a part is the same in both CPQ Cloud and Commerce Cloud.

To synchronize CPQ Cloud parts with Commerce Cloud SKUs:

1. Open the Admin Home page.
2. Under **Products**, select **Parts**.
   The **Parts Search for Admin** page opens.
2. Add new parts in CPQ Cloud with part numbers that match SKUs in Commerce Cloud.
3. Add Part Custom fields for recurring charge price type, frequency, duration, and cost.

NOTES:

- The client-side BML sample included in the *Configure Client-Side Integration, Add To Cart Button, and JSON Response* section of this implementation guide assumes part custom fields 4, 5, 6, and 8 represent recurring period, cost, duration, and type respectively. In order to use other part custom fields, the Add to Cart BML and OIC mappings will have to be adjusted accordingly.
- If a non-configurable SKU is later added to Commerce Cloud and intended for use by the Oracle Commerce Cloud Service and CPQ Cloud Service integration, repeat the above procedure to add the corresponding part in CPQ Cloud.
- In addition to CPQ parts, configurable models must also have a corresponding SKU in Commerce Cloud. The SKU number in Commerce Cloud should match the model's label and variable name.

Configuration Set Up

This section contains the Configuration set up procedures that administrators must complete in CPQ Cloud.

Configure Client-Side Integration, Add To Cart Button, and JSON Payload Response

Administrators must configure a “Client-side” integration to add the **Add to Cart** button on a Commerce Cloud site. The “Client-side” integration enables the sharing of data between CPQ Cloud and Commerce Cloud.

To configure a “Client-side” integration:

1. Open the Admin Home page.
2. Under **Products**, click **Catalog Definition**.
   The **Supported Products** page opens. **Product Families** displays by default in the **Navigation** drop-down menu.
3. Click **List**.
   The **Supported Product Families** page opens.
4. From the **Navigation** drop-down menu, select **Integrations**.
5. Click **List**.
   The **Edit Integration** page opens.
6. Use the **Edit Integration** page to create a “Client-side” integration using the following settings:
   - Name: Add To Cart
   - Integration Type: Client-side
   - Hide in Reconfiguration: No
   - Action: Define Advanced Function
7. Click **Define Function** and use the sample BML from Appendix D: Add to Cart BML to add the **Add to Cart** button to the Commerce Cloud site.
8. For the **End-Point URL**, select the **Simple** option.
   Enter the URL of the Commerce Cloud site to integrate with CPQ Cloud. The value entered should include the basic URL or Commerce Cloud’s storefront and administration pages. Administrators can add multiple Commerce Cloud sites for a single integration by listing each site delimited by the pipe delimiter (|) character.

   For example:
   http://cc-store.oracle.com|http://cc-admin.oracle.com|http://second-store.oracle.com|http://second-admin.oracle.com
9. Click **Apply**.

NOTES:

- The “Add To Cart” BML references File Manager locations. Administrators must modify the BML on a per site basis.

Configure CPQ Models Corresponding to Products in Commerce Cloud

Administrators must create CPQ Cloud models corresponding to SKUs in Commerce Cloud.

To configure models corresponding to products in Commerce Cloud:

1. Open the Admin Home page.
2. Under **Products**, select **Catalog Definition**.
   The **Supported Products** page opens with **Product Families** displaying by default in the **Navigation** drop-down menu.
3. Click **List**.
   The **Supported Product Families** page opens with **Product Lines** displaying by default in the **Navigation** drop-down menu.
4. Click **List**.
   The **Product Line Administration List** page opens with **Models** displaying by default in the **Navigation** drop-down menu.
5. Click **List**.
   The **Model Administration List** page opens.
6. Click **Add**.
7. Use the **Model Administration** page to create a new model with both the variable name and label matching the configurable root SKU in Commerce Cloud.
8. Create a pricing rule on the model with a price matching the root SKU in Commerce Cloud.

Configure Child Line Items Corresponding to SKUs in Commerce Cloud

For information about setting up BOM Mapping items for a model, refer to the CPQ Cloud Administrator Online Help.

NOTES:

- Quantity for the root BOM should use a configurable integer attribute in BOM Attribute Mapping. Otherwise, incorrect quantities may be populated during reconfigure.
- Support for configurable items with BOM mapped items is limited to child parts for CPQ Cloud 2018C.
- A **Disable BOM-Mapping Rules During Updates** setting is available on the **Configuration Settings** page. Since the Oracle Commerce Cloud Service – Oracle CPQ Cloud Service integration does not support this setting, administrators should set it to **No**. Failure to do so may result in unintended access to company associated parts.

Create Configurable Attributes

Configurable attributes define the characteristics of product families. CPQ Cloud uses configurable attributes in search flows, Configuration flows, and every type of Configuration rule.

To create configurable attributes:

1. While administrators can create the following configurable attributes at any level, Oracle recommends creating the attributes at the Product Family level.

Label	Variable Name	Attribute Type	Additional Settings
Currency Code	currencyCode	Text Field	
CC Site ID	cC_SiteId_t	Text Field	
Quantity	quantity	Integer	Required, Default = 1, Positive Number Validation

2. Create a recommendation rule configured as follows:

Condition	Apply Rule To	Action Type	Action Attribute	Values to Set	Set Type
Always True	Configuration	Standard	currencyCode	Edit Function: `return _BM_USER_CURRENCY;`	Forced Set

3. Create any additional attributes that suit your organization's needs and place them on the Configuration flow layout.

NOTES:

- Administrators must place "currencyCode", "cC_SiteId_t", and "quantity" on the layout, but they do not need to display them.
- For information about configurable attributes and the steps to create them, refer to the CPQ Cloud Administration Help.

4. Create a hiding rule configured as follows:

Condition	Action Attribute

Advanced:
`if (_transaction_id == "-1") {`
`  return  true;`
`}`
`  return false;`
	quantity

Commerce Set Up

This section contains the Commerce set up steps that administrators must complete in CPQ Cloud:

NOTE: Request for Quote and Sync Quote flows do not currently support Asset/Subscription based orders.

Create Commerce Attributes at the Transaction Level

Administrators must create the Commerce attributes shown in the following table at the Transaction level and can adjust the attribute labels, as desired.

NOTE: An asterisk (*) next to the attribute label indicates the attribute should already exist as part of the Base Reference Application.

Attribute Label	Variable Name	Attribute Type	Additional Settings
CC Order Id	cC_OrderId_t	Text Field	
Discount Info	cC_DiscountInfo_t	Text Field	
Requestor Note	cC_RequesterNote_t	Text Area	
Request Date	cC_RequestDate_t	Date	Default Value: System Variable: Current Date
Customer*	customer_t	Additional Address Set	
Reject Explanation*	rejectExplanation_t	Text Area	
Rejection Date	cC_RejectionDate_t	Date	
Provider Note	cC_ProviderNote_t	Text Field	
Expiration Date*	priceExpirationDate_t	Date	
CC External Id	cC_ExternalId_t	Text Field	
CC External Order Price	cC_ExternalOrderPrice_t	Currency	Default Value: Use the formula provided in the *Apply Formulas* section.
CC External Order Price Quantity	cC_ExternalOrderPriceQuantity_t	Integer	
CC Expiration Date	cC_ExpirationDate_t	Date	
CC Agent Id	cC_AgentId_t	Text Field	
CC Subtotal	cC_Subtotal_t	Currency	

Attribute Label	Variable Name	Attribute Type	Additional Settings
CC Order Discount	cC_OrderDiscount_t	Float	Auto Update: Yes Default Value: Enter a non-blank default value to ensure the value sent to Commerce Cloud during Sync Quote (i.e. externalOrderPrice) is populated.
CC Order Discount Type	cC_OrderDiscountType_t	Menu	Auto Update: Yes Menu Options: Percent Off, Amount Off, Price Override Default Value: Enter a non-blank default value to ensure the value sent to Commerce Cloud during Sync Quote (i.e. externalOrderPrice) is populated.
CC_LineItem_Data	cC_LineItem_Data_t	Text Area	
CC Total Net Price	cC_TotalNetPrice_t	Currency	Auto Update: Yes Document View: Hide Default Value: Use the formula provided in the *Apply Formulas* section.
Order Discount Total	cC_OrderDiscountTotal_t	Currency	Auto Update: Yes Document View: Hide Default Value: Use the formula provided in the *Apply Formulas* section.
Total (Net)*	totalOneTimeNetAmount_t	Currency	Default Value: Use the formula provided in the *Apply Formulas* section.

Attribute Label	Variable Name	Attribute Type	Additional Settings
Total Discount*	totalOneTimeDiscount_t	Currency	**Default Value:** Use the formula provided in the *Apply Formulas* section.
CC Order Total	cC_Order_Total_t	Currency	
CC Organization Id	cC_OrgId_t	Text Field	
CC Site Id	cC_SiteId_t	Text Field	
CC Site name	cC_SiteName_t	Text Field	
Request Date	cC_RequestDate_t	Date	
Ship To Attributes*	shipTo_t	Additional Address Set	
Invoice To Attributes*	invoiceTo_t	Additional Address Set	

Modify the Existing "Status" Transaction Level Attribute

The Status ("status_t") attribute is an existing Transaction-level attribute that should already exist on Base Ref App environments. Administrators must modify this attribute as described below.

- Add the following options:
  - Rejected [REJECTED]
  - Synced [SYNCED]
- Under **Modify**, set the attribute to "**Use Specified Value**" for the following actions:
  - Create Order: ORDERED
  - Customer Rejection: REJECTED
  - Save: CREATED
  - Sync Quote: SYNCED
  - Cancel Transaction: CANCELED

Create Attributes at the Commerce Line Level and Add Them to the Commerce Layout

Create the Commerce attributes shown below at the Commerce line level. Once created, add the attributes to the Commerce layout.

Label	Variable Name	Attribute Type	Additional Settings
Commerce Item Id	cC_CommerceItemId_l	Text Field	
Product Id	cC_ProductId_l	Text Field	

Catalog Ref Id	cC_CatalogRefId_l	Text Field	Default Value: Function
			`if(_model_variable_name <> ""){`
			`return _model_variable_name;`
			`}`
			`return _part_number;`

External Price	cC_ExternalPrice_l	Currency	
External Price Quantity	cC_ExternalPriceQuantity_l	Integer	
CC Net Price	cC_NetPrice_l	Currency	
Quantity*	requestedQuantity_l	Currency	
Price (List)*	listPrice_l	Currency	Default Value: Use the formula provided in the *Apply Formulas* section.
	oRCL_ABO_ActionCode_l	Single Select Menu	This menu attribute comes from the ABO installation package and is a requirement for the Sync Quote action.

NOTE: An asterisk (*) is used in the above table to indicate attributes that should already exist on Base Ref App environments.

Apply Formulas

The following Commerce attributes should already exist on Base Ref App environments. Apply the listed formulas to the attributes.

Variable Name	Formula
cC_ExternalOrderPrice_t	`if((cC_OrderDiscountType_t = "amountOff"), (cC_TotalNetPrice_t - cC_OrderDiscount_t), if((cC_OrderDiscountType_t = "percentOff"), (cC_TotalNetPrice_t - (cC_TotalNetPrice_t * (cC_OrderDiscount_t / 100))), if((cC_OrderDiscountType_t = "priceOverride"), cC_OrderDiscount_t,cC_TotalNetPrice_t)))`
totalOneTimeNetAmount_t*	`cC_ExternalOrderPrice_t`
totalOneTimeDiscount_t*	`sumIf( ( priceType_l NOT= "Recurring" ), discountAmount_l) + cC_OrderDiscountTotal_t`
cC_OrderDiscountTotal_t	`if( ( cC_OrderDiscountType_t = "amountOff" ), cC_OrderDiscount_t,if( ( cC_OrderDiscountType_t = "percentOff" ), ( cC_ExternalOrderPrice_t - ( cC_OrderDiscount_t / 100 ) ), if( ( cC_OrderDiscountType_t = "priceOverride" ), ( cC_ExternalOrderPrice_t - cC_OrderDiscount_t ), 0 )))`
cC_TotalNetPrice_t	`sumIf( ( priceType_l NOT= "Recurring" ), netAmount_l)`
listPrice_l*	`if( ( _model_base_price NOT= 0 ), _model_base_price,_price_list_price_each)`

Note: An asterisk (*) next to the variable name indicates that a formula for the attribute already exists on Base Ref App environments. Administrators must update the existing formulas as opposed to creating new formulas.

Set Up Commerce Actions

Complete the following steps to set up Commerce actions.

1. Create the following Commerce action at the Transaction level.

Label	Variable Name	Action Type	Integration	Advanced Modify (Before Formulas)
Sync Quote	cC_syncQuote	Modify	CPQ-OCCS Sync Quote	**Transaction Attribute**: CC_LineItem_Data
				Transaction Line Attributes: _document_number _model_variable_name cC_ProductId_l cC_CommerceItemId_l
				BML: Refer to Appendix E: SyncQuote BML

2. Place the Sync Quote action on the Commerce layout.

3. Set the quote level actions “cleanSave_t" and " _remove_transactionLine" to define the following attributes based on their formula definitions:
    - Quote Level Attributes:
        - Total Contract Value
        - Total Discount Per Month
        - Total (List) Per Month
        - Total (Net) Per Month
        - Total Discount
        - Total (List)
        - Total (Net)
        - Annual Contract Value
        - Transaction Total
        - Total Contract Discount
        - Annual Contract Discount
        - CC External Order Price
    - Line Level Attributes
        - Actual Amount
        - Annual Value
        - Contract Value
        - Amount (List)
        - Amount (Net)
        - Price (Net)
        - Quantity
4. Set the line level action “save_l" to define the following line level attributes based on their formula definitions:
    - Actual Amount
    - Annual Value
    - Contract Value
    - Amount (List)

- Amount (Net)
- Price (Net)
- Quantity

NOTES:

- The “Save” action is already setup to use formulas for a majority of these attributes in the Base Ref Application.
- The Request for Quote and Sync Quote flows do not support the “Copy Line Items” action. The action is not accessible for Commerce Cloud integrated Transactions.

(Optional) Create Commerce Validation Rule

Administrators have the option of creating a Commerce validation rule that blocks users from editing the quantity of child items.

1. Open the Admin Home page.
2. Under **Commerce and Documents**, click **Process Definition**.
   The **Processes** page opens with **Documents** displaying by default in the **Navigation** drop-down menu.
3. Click **List** next to the *Oracle Quote to Order* Commerce process.
   The **Document List** page opens.
4. At the Transaction Line level, select **Rules** from the **Navigation** drop-down menu.
5. Click **List**.
6. From the **Add** menu, select **Validation**.
   The **Validation: New Rule** page opens.
7. In the **Name** field, enter a name for the validation rule.
8. Click in the **Variable Name** field to auto-populate the field.
9. For the **Condition Type**, select **Advanced**.
10. Click **Define Function**.
    The **Select Attributes** dialog opens.
11. Select the attributes shown in the following tables.

System Variable Name	Type	Description
_system_current_document_number	String	Current Document Number

Variable Name for (Transaction Line)	Type	Description
_model_variable_name	String	Model Variable Name
_price_quantity	Integer	Quantity

12. Click **Next**.
13. Enter the following BML:

```
 oldvalue = getoldvalue("_price_quantity",
atoi(_system_current_document_number));


 if((_model_variable_name == "") AND (_price_quantity <> atoi(oldvalue))) {


 return true;


 }
 return false;
```

14. Click **Save and Close**.
15. On the **Validation: New Rule** page, select **Advanced** as the **Action Type**.
16. Click **Define Function**.
    The **Select Attributes** dialog opens.
17. Select the **Variable Name for (Transaction Line)** tab.
18. Select the "_price_quantity" attribute.
19. Click **Next**.
20. Enter the following BML.

```
 attributeDict = dict("dict<string>");


    // inner dictionary for attr2
    attr2ActionDict = dict("string");
```

```
      // assembling the constraint action
      put(attr2ActionDict, BM_CM_RULES_MESSAGE, "Please re-configure the item to
change quantity of sub-item");


      // put the inner dictionary into the outer dictionary
      put(attributeDict, "_price_quantity", attr2ActionDict);



  // return the outer dictionary
  return attributeDict;
```

21. Click **Save and Close**.

Set Up Steps

Administrators must use CPQ Cloud to create a Synced step as well as step transitions.

1. Create a new “Synced” step.
2. Create a step transition for the "Sync Quote" action to move from the "In Progress" step to the "Synced" step.
3. Create a step transition for the “Save" action to move from the "Synced" step to the "In Progress" step.
4. Create a step transition for the "Customer Rejection" action to move from the "Synced" step to the "Rejected by Customer" step.
5. Create a step transition for the “Create Order” action to move from the “Synced” step to the “Ordered” step.
6. Create a step transition for the “Cancel Transaction” action to move from the “Synced” step to the “Canceled” step.
7. Hide the "Sync Quote" action from the following steps:
    - Fulfilled
    - Canceled
    - Rejected By Customer
8. Hide all Modify actions from the “Synced” step EXCEPT the following:
    - Save
    - Customer Rejection
    - Create Order
    - Cancel Transaction

NOTES:

- Make sure all of the attributes used in the Request for Quote flow have read/write access at the Start step.
- For instructions on how to create Commerce attributes, actions, and step transitions, refer to the CPQ Cloud Administration Help.

Modify Process Manager View

Administrators must complete the following procedure to modify a process manager view.

1. Add a data column to the "Order Id" Commerce process.
2. Map the data column to the "Order Id" quote level attribute.
3. Add a Process Manager column using the Order Id data column.

Set Up Subscription Ordering in CPQ

For information about setting up Subscription or Asset based orders, refer to the ABO implementation guide and the CPQ Cloud Administrator Online Help.

Set Up OIC Integrations

Administrators must import an OIC Integration Package to an OIC environment that connects Commerce Cloud and CPQ Cloud through a common configuration. The OIC Integration Package creates web service connections that allow users to adjust order and quote details in CPQ Cloud, approve or reject changes in Commerce Cloud, and complete or cancel orders in Commerce Cloud. This section contains the steps an administrator must complete to set up and activate the OIC integrations.

Download the Integration Packages

Administrators must complete the following procedure to download the OIC Integration Package.

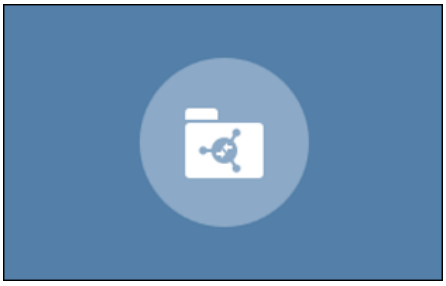
1. Go to the Integrating Oracle Commerce Cloud and CPQ Cloud article on My Oracle Support.
2. If you want to implement the integration between Commerce Cloud and the CPQ Cloud Configurator, download OCCS-CPQ_CONFIGURATION_INTEGRATION_X.X.par to a location where it is accessible from OIC.
3. If you want to implement the integration between Commerce Cloud and CPQ Cloud Quoting, download OCCS-CPQ_QUOTE_INTEGRATION_X.X.par to a location that is accessible from OIC.
4. If you want to enable Asset Based Ordering (ABO) through the integration between Commerce Cloud and CPQ Cloud, download OCCS_CPQ_ASSET_INTEGRATION_X.X.par to a location that is accessible from OIC.

Import the Integration Package

Import the OIC Integration Package into OIC to create an integration between Commerce Cloud and CPQ Cloud in OIC.

To import the OIC Integration Package:

1. Log in to OIC as an admin user.
2. Click the **Packages** icon.



3. Click the **Import** button.
4. Click **Browse**.
5. Select the integration package archive (PAR) file you want to import.
6. Click **Import**. The package is added to the Packages list.

The OCCS-CPQ_QUOTE_INTEGRATION package includes the following three integration flows: *OCCS-CPQ Create Quote*, *OCCS-CPQ Update Quote*, and *OCCS-CPQ Sync Quote*.

- The *OCCS-CPQ Create Quote* integration sends quote request information to CPQ Cloud.
- The *OCCS-CPQ Sync Quote* integration allows CPQ Cloud to send information to Commerce Cloud at the end of the quoting process and synchronize this information in Commerce Cloud. This ensures that the order information in Commerce Cloud matches the related order information in CPQ Cloud.
- The *OCCS-CPQ Update Quote* integration sends information to CPQ Cloud related to accepting, rejecting, or re-requesting a quote.

The OCCS-CPQ_CONFIGURATION_INTEGRATION package includes the *OCCS-CPQ Get Configurations* integration flow. This integration is required for the configuration flow and is available to import into OIC. The name of the target connection for this integration is “Oracle CPQ”. The target connection identifier is “Oracle_CPQ”, and the target connection description is “Oracle CPQ ICS Adapter Connection”.

The OCCS-CPQ_ASSET_INTEGRATION package includes two integration flows: OCCS-CPQ Get Assets and OCCS-CPQ Asset Actions. This integration is required for Asset Based ordering.

- The *OCCS-CPQ Get Assets* integration returns information about assets and services associated with the shopper’s account(s)
- The *OCCS-CPQ Asset Actions* integration enables Commerce Cloud to modify, renew, and terminate actions on assets and services associated with the shopper’s account(s).

NOTE: Importing and setting up the OIC Integration Package is a prerequisite to completing the “Sync Quote” action in CPQ Cloud. After all CPQ Cloud setup procedures are completed, regenerate the OCCS-CPQ Create Quote integration to ensure it accurately reflects the current state of the *Oracle Quote to Order* process.

Configure CPQ Cloud Connections

Administrators must configure connections from the integrations referenced in the previous section to CPQ Cloud. The following CPQ Cloud connections are part of the integrations: Oracle CPQ Cloud, Oracle Commerce Cloud, Oracle CPQ getConfigurations, Oracle CPQ Quote, CPQ Get Assets, and CPQ Asset Actions. Each connection corresponds to different SOAP or REST APIs for CPQ Cloud web services. Setting a connection to use the wrong API will cause the integrations to fail.

To configure the CPQ Cloud connections:

1. Log in to OIC as an admin user.
2. Click the **Connections** icon.

3. Click the **Oracle CPQ Cloud** connection.
4. Click **Configure Connectivity**.
5. Add the WSDL or REST metadata URL for the CPQ Cloud getConfigurations API.

NOTE: The CPQ Transaction and Asset connections are REST based and use the REST Catalog URL. The getConfigurations and Sync Quote connections are SOAP based and use the WSDL URL.

6. Click **OK**.
7. Click **Configure Security**. The CPQ Cloud connection uses the Basic security policy, so you must enter the login details for your CPQ Cloud account.
8. Click **OK**.
9. Click **Test** to test the connection.
10. Click **Save**.
    The CPQ Cloud connection is now configured for the integration. Repeat steps 1-10 for each of the remaining CPQ Cloud connections.

Generate Security Token for Commerce Cloud connections

An administrator must generate a security token to support the Commerce Cloud REST web service APIs used to access Commerce Cloud data.

1. Log in to Commerce Cloud.
2. Click the **Menu** icon.
3. Select **Settings** from the menu.
4. Click **Web APIs** from the sidebar menu.
5. Click **Registered Applications** from the **Web APIs** panel.
6. Click **Register Application**.
7. Enter a name for the integration. Since you are registering OIC, choose a meaningful name that reflects the integration.
8. Click **Save**. The Application ID and Application Key are automatically generated. The application displays on the **Registered Applications** page.
9. Click the name of the application you created.
10. Select **Click to reveal** to display the application key.

NOTE: Administrators need the application key when configuring the Commerce Cloud connection in OIC. Copy the registration key, so it's available when you complete the *"Configure the Commerce Cloud Connection"* procedure.

Configure the Commerce Cloud Connection

An administrator must complete the following steps to configure the connection from the OIC integrations to Commerce Cloud.

1. Log in to OIC as an admin user.
2. Click the **Connections** icon.
3. Click the **Oracle Commerce Cloud** connection.
4. Click **Configure Connectivity**.
5. Enter the Connection base URL, which is derived using the below structure, where <siteURL> is the base URL of the Oracle Commerce Cloud site that integrates with OIC.

   ```
   Connection base URL: https://<hostname>:<port>/ccadmin/v1
   ```

6. Click **Configure Security**. The Commerce Cloud connection uses the OAuth security policy, so you must enter the security token for the connection. The security token was generated in the *Generate Security Token* section.
7. Click **OK**.
8. Click **Test**.
9. Click **Save**.
   Your Commerce Cloud connection is now configured for the integration.

Activate the OIC Integrations

Once the CPQ Cloud, Commerce Cloud, Oracle CPQ Quote, Oracle CPQ Configure, and Oracle CPQ getConfigurations connections are configured, the administrator must activate these integrations.

To activate the OIC integrations:

1. Log in to OIC as an admin user.
2. Click the **Integrations** icon to display the **Integrations List**.
3. Click **Activate** for the integration you want to activate.
4. Decide whether you want to switch on detailed tracing, which collects information about messages processed by the integration flow. Administrators may find detailed tracing helpful when troubleshooting issues with the integration flow, but it may impact performance.

   To switch on detailed tracing, select the **Enable detailed tracing** check box.

NOTE: Once an integration flow is active, administrators must deactivate and then reactivate the flow to switch detailed tracing on or off.

5. Click **Activate**.

Create Sync Quote Action in CPQ Cloud

Create the following Commerce action at the Commerce quote level:
Label(Sync Quote), Variable Name(syncQuote), Action Type(Modify)

NOTE: syncQuote uses the OIC integration for the "CPQ-OCCS Sync Quote" service.

Set Up OIC Integration on CPQ Cloud Site

Administrators must set up the OIC integration on the CPQ Cloud site by completing the following steps:

1. Click **Admin** to go to the Admin Home page.
2. Navigate to **Integration Platform** > **Integration Center**.
   The **Integration Center** opens.
3. From the **Type** drop-down menu, select **Integration Cloud Service**.
4. In the **Name** field, enter Sync Quote integration.
   The Variable Name field will auto-populate.
5. In the **Discovery URL** field, enter the OIC domain.
6. In the **Username** field, enter a valid username.
7. In the **Password** field, enter a valid password.
8. Click **Create Integration**.

Set Sync Quote Action to Run Advanced Modify

Complete the following steps to set the Sync Quote action to run Advanced Modify:

1. Open the Admin Home page.
2. Navigate to **Process and Documents** > **Process Definition**.
   The **Processes** page opens with **Documents** displaying by default in the **Navigation** drop-down menu.
3. Click **List**.
   The **Document List** page opens.
4. From the **Navigation** drop-down menu, select **Actions**.
5. Click **List**.
   The **Action List** page opens.
6. Click the syncQuote link.
   The **Admin Action** page opens.
7. Under the General Tab > Advanced Modify - Before Formulas > select **Define Advanced Modify - Before Formulas**.
8. Click **Define Function**.
9. Select the attributes shown in the following tables:

Variable Name for (Transaction)	Type	Description
cC_LineItem_Data	String	CC_LineItem_Data

Variable Name for (Transaction Line)	Type	Description
transactionLine	Collection of Sub Documents	
_document_number	String	Document Number
_model_variable_name	String	Model Variable Name
cC_ProductId_I	String	Product Id
cC_CommerceItemId_I	String	Commerce Item Id

10. Insert the following BML in BML Editor:

```
str = "";


for each in transactionLine{
    if (each._model_variable_name <> ""){
        lineItem_array = split(cC_LineItem_Data, "|");
        for lineItem in lineItem_array {
            row = split(lineItem, "~");
            if(row[0] == each._document_number){
                str = str + each._document_number + "~ cC_CommerceItemId_l ~"
+ row[1]+"|";
                str = str + each._document_number + "~ cC_ProductId_l ~" +
row[2]+"|";
            }
        }
    }
}
     return str;
```

11. Update and click **Save**.
12. Navigate to the **Integration** tab and move **Sync quote** above **Modify Functions**.
13. Update and click **Save**.
14. Place the “syncQuote” action on the layout.

Configure Commerce Cloud Webhooks

The REST API generated by activating the OIC integration can be configured as a webhook in Commerce Cloud Administration.

- **Request Quote**: This webhook is triggered when a request or a re-request for a quote is submitted by a Commerce Cloud self-service user. The webhook pushes notifications using the *OCCS-CPQ Create Quote* integration flow.
- **Update Quote**: This webhook is triggered when a response to a requested quote is accepted or rejected or the quote order is cancelled by the Commerce Cloud self-service user. This webhook pushes notifications using the *OCCS-CPQ Update Quote* integration flow.
- **External Price Validation**: This webhook is triggered at check out when the order contains one or more items configured by CPQ Cloud. The webhook validates the configuration and the price provided for configured items.
- **Contact Accounts Retrieval**: This webhook returns a list of service account IDs for the shopper.
- **Services Retrieval**: This webhook returns information about a service or asset associated with the shopper and uses the *OCCS-CPQ Get Assets* integration flow. This webhook calls the Contact Accounts Retrieval webhook, so that webhook must also be configured for the Services Retrieval webhook to function correctly.
- **Service Actions**: This webhook is used to perform modify, renew, and terminate actions on a service or asset and uses the *OCCS-CPQ Asset Actions* integration flow. This webhook calls the Contact Accounts Retrieval webhook, so that webhook must also be configured for the Services Retrieval webhook to function correctly.

NOTE: Administrators must configure the *Production* and *Preview* versions of the webhooks to ensure they work in all environments. The *Production* webhooks send information from the live Commerce Cloud store to the production environments of your live systems. The *Preview* webhooks send information from the preview environment to the test or sandbox environments of external systems.

To configure Request Quote, Update Quote, External Price Validation, Services Retrieval, or Service Actions webhooks:

1. Log in to OIC as an admin user.
2. Click the **Integrations** icon.
3. Click the **Integration Details** icon to display information about the integration flow.
    - If configuring the **Request Quote** webhook, display information for the *OCCS-CPQ Create Quote* integration flow.
    - If configuring the **Update Quote** webhook, display information for the *OCCS-CPQ Update Quote* integration flow.
    - If configuring the **External Price Validation** webhook, display information for the *OCCS-CPQ GetConfigurations* integration flow.

- If configuring the **Services Retrieval** webhook, display information for the *OCCS-CPQ Get Assets* integration flow.
- If configuring the **Service Actions** webhook, display information for the *OCCS-CPQ Asset Actions* integration flow.

4. Copy the Endpoint URL for the integration.
5. Log in to Commerce Cloud.
6. Click on the **Menu** icon.
7. Select **Settings** from the menu.
8. Select **Web APIs** from the sidebar menu.
9. Click the webhook you want to configure.
10. Paste the Endpoint URL that was copied into the URL field for the webhook.
11. Remove the “metadata” text from the end of the URL.
12. Enter your OIC user name and password.
13. Click **Save**.
    The webhook is now configured and is triggered each time the relevant event occurs, which in turn triggers the relevant integration flow.

Understand the Services Action SSE

Service actions to perform modify and renew actions on a service or asset are performed using server side extensions, one set for Storefront and one for Agent to perform.

See the section Configure the Commerce Cloud Server Side Extensions for information on these actions.

NOTE: For information about Oracle Commerce Cloud webhooks, refer to the “Configure Webhooks” chapter of the *Using Oracle Commerce Cloud Service* document.

Configure the Commerce Cloud Server-Side Extensions

Commerce Cloud includes some server-side extensions (SSEs) that you can configure to perform specific functions relating to asset-based orders.

Before you install any of the server side extensions, you must make sure your custom Node.js server is associated with your Oracle Commerce Cloud environment. You can check this using the GET `/ccadmin/v1/extensionServers` endpoint in the Admin API:

```
GET /ccadmin/v1/extensionServers  HTTP/1.1
Content-Type: application/json
Authorization: Bearer <access_token>
```

The response lists the extension servers associated with your Commerce Cloud environment. If there are no extension servers associated with your environment, use the POST /ccadmin/v1/extensionServers endpoint to associate your extension server. For example:

```
POST /ccadmin/v1/extensionServers  HTTP/1.1
Content-Type: application/json
Authorization: Bearer <access_token>

{
    "url" : "http://myextserver.example.com:14000",
    "type" : "custom",
    "active" : true
}
```

You can configure the SSEs directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE that you want (for example, Order Qualification).

For more complete information on server-side extensions and how to develop them for use with Oracle Commerce Cloud, refer to Develop server-side extensions section in Extending Oracle Commerce Cloud.

Configure the Credit Check SSE

The Credit Check SSE is used to connect to a third-party credit check system so that you can perform a credit check on the logged-in shopper.

Understand the Check Credit endpoint

The Check Credit endpoint is triggered whenever a credit check is requested by Commerce Cloud. You can configure the Credit Check Model SSE directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE. The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

The inputs for this endpoint are:

- Amount information
- Recurring amount frequency
- Recurring amount duration
- Recurring amount
- Contact information
    - First Name
    - Last Name
    - Email Address
    - Telephone Number
- Address information
    - Address line 1
    - Address line 2
    - City
    - State
    - Country
    - Postal code

The return for this endpoint is either a TRUE or FALSE value depending on whether the shopper passed the credit check or not.

Understand the Create Contact endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud.

The input for this endpoint is the User Token for the logged-in shopper.

The return for this endpoint is the new External Contact ID created for the shopper.

Understand the Query Accounts endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud and when they go to checkout for an order that contains service items.

The input for this endpoint is the User Token for the logged-in shopper.

The returns for this endpoint are the accounts, roles, addresses, and business profiles associated with the shopper.

Understand the Query Contacts endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud.

The input for this endpoint is the User Token for the logged-in shopper.

The return for this endpoint is the External Contact ID for the shopper.

Understand the Update Accounts endpoint

This endpoint is triggered when a shopper saves an account address.

The inputs for this endpoint are:

- User Token for the logged-in shopper.
- The Account ID of the account to which the billing profile is linked.
- The new address as provided by the shopper.

The returns for this endpoint are the accounts, roles, addresses, and business profiles associated with the shopper.

Configure the Customer Account Model SSE

This SSE is used to return information about the customer account model for a registered shopper or to update the customer account model when required.

You can configure the Customer Account Model SSE directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE. The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Create Accounts endpoint

This endpoint is triggered if the Query Accounts endpoint does not return any accounts for the shopper.

The inputs for this endpoint are:

- User Token for the logged-in shopper.
- Account Type
- Account Name
- Primary Contact
- Billing Profile(s)
- Address(es)
- Contact ID(s)
- Contact Role(s)

The returns for this endpoint are the accounts, roles, addresses, and business profiles now associated with the shopper.

Understand the Create Contact endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud.

The input for this endpoint is the User Token for the logged-in shopper.

The return for this endpoint is the new External Contact ID created for the shopper.

Understand the Query Accounts endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud and when they go to Checkout for an order that contains service items.

The input for this endpoint is the User Token for the logged-in shopper.

The returns for this endpoint are the accounts, roles, addresses, and business profiles associated with the shopper.

Understand the Query Contacts endpoint

This endpoint is triggered when a shopper logs in to Commerce Cloud.

The input for this endpoint is the User Token for the logged-in shopper.

The return for this endpoint is the External Contact ID for the shopper.

Understand the Update Accounts endpoint

This endpoint is triggered when a shopper saves an account address.

The inputs for this endpoint are:

- User Token for the logged-in shopper.
- The Account ID of the account to which the billing profile is linked.
- The new address as provided by the shopper.

The returns for this endpoint are the accounts, roles, addresses, and business profiles associated with the shopper.

Understand the Update Accounts endpoint

This endpoint is triggered when a shopper saves an account address.

The inputs for this endpoint are:

- User Token for the logged-in shopper.
- The Account ID of the account to which the billing profile is linked.
- The new address as provided by the shopper.

The returns for this endpoint are the accounts, roles, addresses, and business profiles associated with the shopper.

Configure the Order Qualification SSE

This SSE is used to perform any final checks on an order before payment is authorized and the order is submitted to downstream systems for processing and fulfillment.

You can configure the Order Qualification SSE directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE. The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Order Qualification endpoint

This endpoint is triggered by the Order Validation webhook when any order containing a configured item is submitted.

The input for this endpoint is the order containing the configured item.

The return for this endpoint is either a TRUE or FALSE value depending on whether the order passed the validation check or not. If the value is FALSE the return also includes information about which item(s) in the order failed validation.

Configure the Order Qualification Pipeline SSE

This SSE is used to ensure that an order is valid. It checks that the customer account, service account, billing account, and billing profile information for each configured item in an order is valid.

You can configure the Order Qualification SSE directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE. The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Order Qualification endpoint

This endpoint is triggered when a shopper goes to checkout for an order that contains configured items.

The inputs for this endpoint are:

- Contact record for the shopper
- Order containing configured items.

The return for this endpoint is either a TRUE or FALSE value depending on whether the order passed the validation check or not. If the value is FALSE the return also includes information about which item(s) in the order failed validation.

Configure the Service Actions SSE

Service actions to perform modify and renew actions on a service or asset are performed using server side extensions, one set for Storefront and one for Agent to perform.

You can configure the Service Actions SSE directly from Commerce Cloud Administration. From the menu, click Settings then Web APIs. Select the name of the SSE. The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Service Actions endpoints

The Server Side Extension Endpoints for Service Actions are Modify and Renew.

These endpoints are triggered when a user performs an operation on an asset.

The inputs for these endpoints are:

- Logged in User Token.
- Asset ID, the unique ID for the asset for this operation. This may be a root, branch or leaf asset.

The returns for this endpoint are BOM (Bill of Materials) and Error.

Enable Integrations in Commerce Cloud

You must complete the procedures in this section to enable the CPQ Cloud Configurator integration, the CPQ Cloud Request For Quote integration, and the Asset Based Ordering (ABO) integration in Commerce Cloud. For additional information about these integrations, refer to Appendix A: Configurator Flow and Appendix B: Request for Quote Flow.

Enable CPQ Cloud Configuration Integration

Complete the following steps to enable the CPQ Cloud Configuration integration:

1. Log in to Commerce Cloud.
2. Click on the **Menu** icon.
3. Select **Settings** from the menu.
4. Select **Oracle Integrations** from the sidebar menu.
5. Select **CPQ Configuration** from the drop-down menu.
6. Select the **Enable Integration** check box.
7. Enter the Configuration URL using the following structure:
   `https://<cpq_domain>/commerce/new_equipment/products/model_configs.jsp`
8. Enter the Reconfiguration URL using the following structure:
   `https://<cpq_domain>/commerce/new_equipment/products/external_reconfig.jsp`

NOTE: Enter the Configuration URL and the Reconfiguration URL for both the Production and Preview environments.

9. Click **Save**.

   If you are using a multisite environment you must follow these instructions for each site that uses the CPQ Configuration integration.

Identify Configurable Products in the Product Catalog

Before a Commerce Cloud self-service user can use the CPQ Cloud configurator to configure complex products for purchase in Commerce Cloud, an administrator must identify the products as configurable in the product catalog. Before doing so, it is important to have a synchronized product catalog to ensure that products in the Commerce Cloud catalog map to corresponding items in the CPQ Cloud catalog.

To identify a product as configurable:

1. Log in to Commerce Cloud.
2. Click on the **Menu** icon.
3. Select **Catalog** from the menu.
4. Select the product you wish to identify as configurable.

5. Click on the **SKUs** tab of the product detail pop-up frame.
6. Select the SKU you wish to identify as configurable.
7. Check the **Configurable** checkbox. This displays three further fields you must complete.
8. Enter the **Model** information. This should match the Model information of a configurable product in the CPQ Cloud catalog.
9. Enter the **Product Line** information. This should match the Product Line information of a configurable product in the CPQ Cloud catalog.
10. Enter the **Product Family** information. This should match the Product Family information of a configurable product in the CPQ Cloud catalog.
11. Click **Save**. This returns you to the SKU frame where the SKU you updated should be marked with an asterisk to identify it as a configurable SKU.

NOTE: Administrators can also perform the above setup steps in bulk by using the SKU import program. From the **Catalog** tab in Commerce Cloud, click **Manage Catalog** and select **Import**. In the **Import** dialog, click **Browse** and locate the CSV file to import. Click **Upload File,** click **Validate,** and then click **Import.**

Add Customize Button to the Product Details Widget

Administrators must add a Customize button to the Product Details widget, so the button is visible to Commerce Cloud self-service users from the Product Details page for a customizable product.

To add a Customize button to the Product Details widget:

1. Log in to Commerce Cloud.
2. Click on the **Menu** icon.
3. Select **Design** from the menu.
4. Select **Product Layout** from the layout list.
5. Delete the **Product Details** widget from the layout.
6. Place a new product details widget on the layout.
7. Click the **Settings** icon for the new **Product Details** widget.
8. From the **Element Library**, place a **Customize** button on the new **Product Details** widget.
9. Publish the changes.

Enable CPQ Cloud Quoting Integration

Complete the following steps to enable the CPQ Cloud quoting integration:

1. Log in to Commerce Cloud.
2. Click on the **Menu** icon.
3. Select **Settings** from the menu.
4. Select **Oracle Integrations** from the sidebar menu.

5. Select **CPQ Quoting** from the drop-down menu.
6. Select the **Enable Integration** check box.

If you are using a multisite environment you must follow these instructions for each site that uses the CPQ Quoting integration.

Add Quote Button to Checkout and Order Details Pages

To make the CPQ Cloud quoting capability available to Commerce Cloud self-service users, administrators must add the **Request Quote** widget to the **Checkout** layout and the **Quote Details** widget to the **Order Details** layout.

The **Request Quote** widget adds a **Quote Notes** text box and a **Request Quote** button to the **Checkout** layout.

The **Quote Details** widget adds a **Quote Notes** text box populated with any notes associated with the order to the **Order Detail** layout. The widget also adds a **Reject Quote**, **Request Re-Quote**, and **Accept Quote** buttons to the to the **Order Detail** layout.

The **Quote Details** and **Request Quote** widgets do not display on the layouts by default. The administrator must first make the widgets available and then place them on the **Checkout** and **Order Detail** pages.

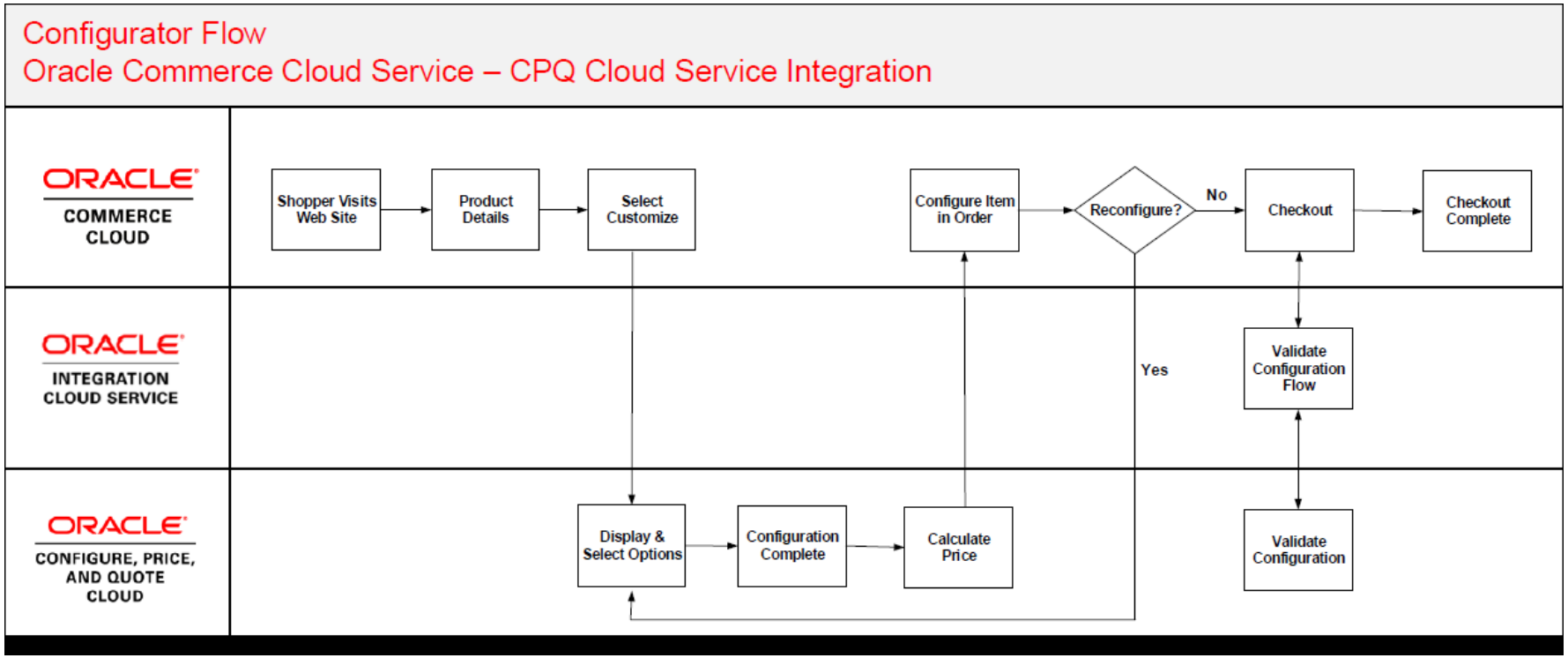
To add quote buttons to the **Checkout** and **Order Details** pages:

1. Log in to Commerce Cloud.
2. Click the **Menu** icon.
3. Select **Design** from the menu.
4. Select the **Components** tab on the **Design** page.
5. Click **Show Hidden**.
6. Click the **Show** icon for the **Quote Details Widget** and the **Request Quote Widget**.
7. Within the **Design** page, select the **Layouts** tab.
8. From the layout list, select **Checkout Layout**.
9. Drag and drop the **Request Quote** widget from the **Components** menu to the desired location on the **Checkout** layout.
10. From the layout list, select **Order Details**.
11. Drag and drop the **Quote Details** widget from the **Components** menu to the desired location on the **Order Details** layout.
12. Publish the changes.

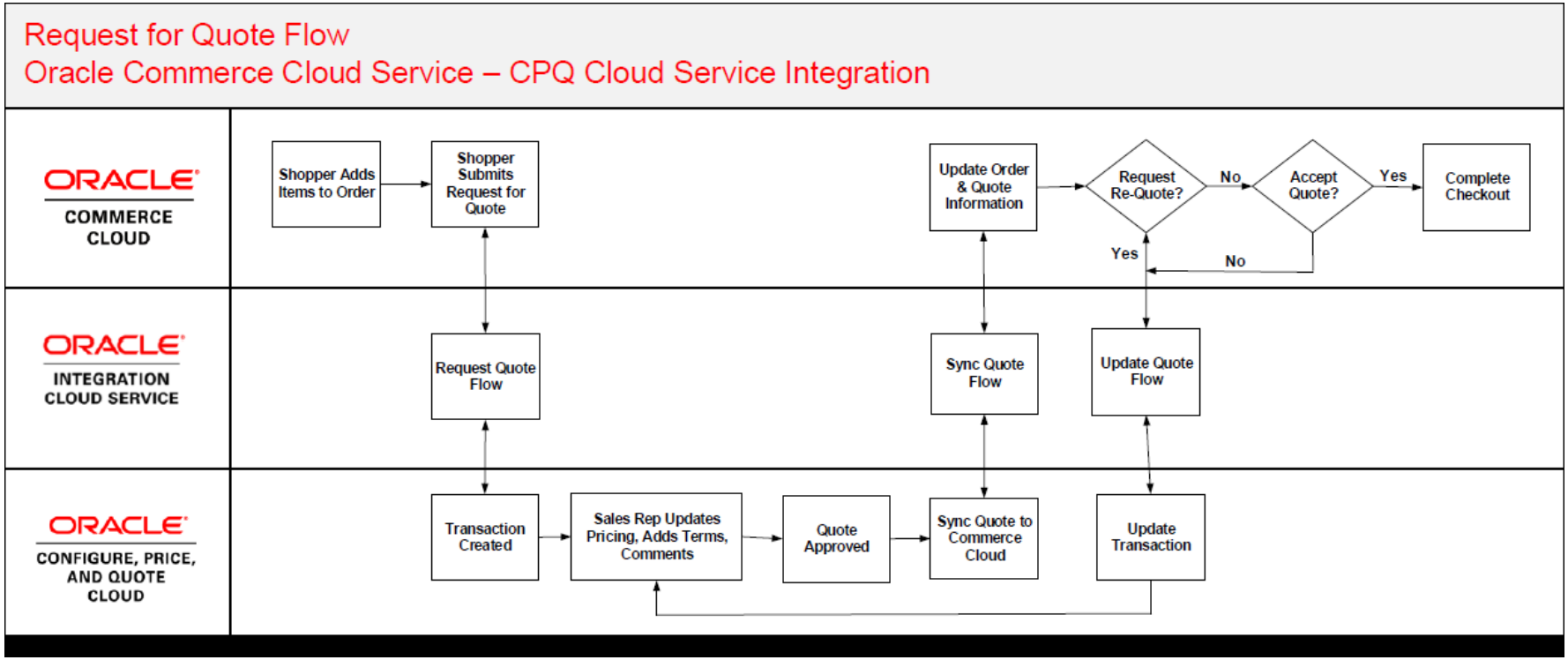
Enable Asset Based Ordering

To enable Asset Based Ordering, you must make sure that you have set up the right integration webhooks and/or SSEs mentioned in the Configure the Commerce Cloud Webhooks and Configure the Server Side Extensions sections of this document.

Appendix A: Configurator Flow



Appendix B: Request for Quote Flow



Appendix C: OIC Integration Mappings

Importing and setting up the OIC package is a prerequisite to completing the Sync Quote action in CPQ Cloud. After all CPQ Cloud setup is completed, regenerate the OIC integration flows to ensure they accurately reflect the current state of the *Oracle Quote to Order* Commerce process.

Note: Mappings in bold indicate complex, conditional mappings. Mappings in italics indicate the mappings are a static text value instead of a source attribute.

Integration Flow	Target Variable Name	Mapping	Comments
OCCS-CPQ Create Quote > New_Transaction			
	cC_RequesterNote_t	requesterNote	
	cC_OrgId_t	organizationId	
	cC_OrderId_t	id	
	cC_SiteId_t	siteId	
	cC_RequesterNote_t	requesterNote	
	currencyCode	currencyCode	
	_customer_t_address	**shippingGroups >address1**	
	_customer_t_state	**shippingGroups > state**	
	_customer_t_address_2	**shippingGroups > address2**	
	_customer_t_company_name	shippingGroups > companyName	
	_customer_t_country	**shippingGroups > country**	
	_customer_t_city	**shippingGroups > city**	
	_customer_t_zip	**shippingGroups > postalCode**	
	_customer_t_phone	**shippingGroups > phoneNumber**	

Integration Flow	Target Variable Name	Mapping	Comments
	_customer_t_email	email	
	_customer_t_last_name	lastName	
	_customer_t_first_name	firstName	
	items	*commerceItems*	
	_price_book_var_name	*_default_price_book*	
	_configuration_id	configuratorId	
	cC_CommerceItemId_l	id	
	_part_number	catalogRefId	
	cC_CatalogRefId_l	catalogRefId	
	_price_quantity	*quantity*	
	cC_ProductId_l	productId	
	cC_NetPrice_l > value	**priceInfo > amount > quantity**	
	cC_NetPrice_l > currency	priceInfo > currencyCode	
	_modify_action	*cleanSave_t*	
OCCS-CPQ Create Quote > Update_Quote			
	id	id	
	externalId	bs_id	
OCCS-CPQ Create Quote > Re-			

Integration Flow	Target Variable Name	Mapping	Comments
Request_Quote			
	cC_RequesterNote_t	requesterNote	
	id	externalId	
OCCS-CPQ Sync Quote			
	id	cC_OrderId_t	
	providerNote	cC_ProviderNote_t	
	agentId	cC_AgentId_t	
	externalId	id	
	expirationDate	**cC_ExpirationDate_t**	
	externalPrice	totalOneTimeNetAmount_t	
line-item			
	productId	cC_ProductId_l	
	catalogRefId	cC_CatalogRefId_l	
	configuratorId	_configuration_id	
	externalPrice	netPrice_l	
	externalPriceQuantity	*-1*	
	id	cC_CommerceItemId_l	
	actionCode	oRCL_ABO_ActionCode_l	

Integration Flow	Target Variable Name	Mapping	Comments
	quantity	**requestedQuantity_l**	
	externalData	configattrinfo	xsl manipulations to feed config attributes as an array of maps. Format: <externalData> <name></name> <values> <name></name> <variableName></variableName> <label>Id</label> <displayValue></displayValue> <value></value> </values> </externalData> < externalData> <name></name> <values> <name></name> <variableName></variableName> <label></label> <displayValue></displayValue>

Integration Flow	Target Variable Name	Mapping	Comments
			<value></value> </values> </ externalData>
OCCS-CPQ Update Quote > Accept Quote			
	id	externalId	
	cC_AgentId_t	agentId	
OCCS-CPQ Update Quote > Reject Quote			
	id	externalId	
	cC_AgentId_t	agentId	
	cC_RejectionDate_t	date	
	rejectExplanation_t	note	
OCCS-CPQ Update Quote > Cancel Quote			
	id	externalId	
	cC_AgentId_t	agentId	
	cC_RejectionDate_t	date	
	rejectExplanation_t	note	
OCCS-CPQ Get Configurations			
	locale	locale	
	currency	currencyCode	
	configurationId	**configuratorId**	

Integration Flow	Target Variable Name	Mapping	Comments
	price	*true*	
	spare	*true*	
	bomMapping	*true*	
OCCS-CPQ Get Assets			
	limit	limit	
	offset	offset	
	q	for-each(id), for-each(id), for-each(recordId), "{$and:[{$or:[", "{id:{$eq: "", recordId, ""}}", ""}},", "]}", ",", "{$and:[", "{$or:[", "{customer:{$eq:"", id, ""}}", ""}},", "]}", ",", "{$or:[", "{serviceAccount:{$eq:"", id, ""}}", ""}},", "]}", "]}"	
	expand	*descendantAssets*	
OCCS-CPQ Asset Actions (for all flows)			
	id	recordId	
	sourceIdentifier	sourceIdentifier	
	transactionDate	transactionDate	
	transactionId	transactionId	
OCCS-CPQ Asset Actions (CpqModifyAsset flow)			

Integration Flow	Target Variable Name	Mapping	Comments
	productLine	product_line	
	configContextKey	configContextKey	
	configuratorUrl	configuratorURL	
	bomKey	bomkey	
	segment	segment	
	model	model	
OCCS-CPQ Asset Actions (CpqRenewAsset, CpqTerminateAsset, CpqSuspendAsset, CpqResumeAsset flows)			
	configId	lineId	
	serviceAccountId	serviceAccount	
	deactivationDate	endDate	
	amount	amount	
	quantity	quantity	
	parentServiceId	parentId	
	externalRecurringCharge	field5	Corresponds to part custom field 5 in CPQ
	externalData	attributes	
	billingAccountId	billingAccount	
	externalRecurringChargeFrequency	field4	Corresponds to part custom field 4 in CPQ
	childItems	for-each(children), for-each(partNumber)	

Integration Flow	Target Variable Name	Mapping	Comments
	catalogRefId	partNumber	
	configuratorId	lineId	
	externalRecurringDuration	field6	Corresponds to part custom field 6 in CPQ
	externalPrice	_price_unit_price_each	
	assetId	id	
	actionCode	oRCL_ABO_ActionCode_l	
	serviceId	id	
	activationDate	startDate	

Appendix D: Add to Cart BML

```
// Rec Item Properties
part = String[1];
quantity = String[1];
price = String[1];
selected = String[1];
sparepaths = String[1];
sparepaths[0] = "/configuration/configureResponse/spare/rule/item/part";
sparepaths[1] = "/configuration/configureResponse/spare/rule/item/quantity";
sparepaths[2] = "/configuration/configureResponse/spare/rule/item/price";
sparepaths[3] = "/configuration/configureResponse/spare/rule/item/selected";

// BOM Item Properties
bomItem = String[1];
bomItem[0] = "/configuration/configureResponse/bomItem";

// Model/Price Properties
models = string[1];
configIdSearch = string[1];
currpath = String[1];
totalPrices = string[1];
bomTotals = string[1];
models[0] = "/configuration/configureResponse/item/model";
configIdSearch[0] = "/configuration/configureResponse/item/@configurationId";
currpath[0] =
"/configuration/configureResponse/attributes/attribute[@_variableName='currenc
yCode']/value";
totalPrices[0] = "/configuration/configureResponse/price/totalPrice";
bomTotals[0] = "/configuration/configureResponse/price/bomPrice";
priceTotal = 0.0;
baseModelPrice = 0.0;
```

```
recurringSubtotal = 0.0;

// Extract data from configXML
outputModel = readxmlsingle(configXML, models);
outputConfigIds = readxmlsingle(configXML, configIdSearch);
currXML = readxmlsingle(configXML, currpath);
currency = get(currXML, currpath[0]);
outputPrices = readxmlsingle(configXML, totalPrices);
bomPrices = readxmlsingle(configXML, bomTotals);
output1 = readxmlmultiple(configXML, sparepaths);
bomItemXMLDict = readxmlsingle(configXML, bomItem);
bomItemString = get(bomItemXMLDict,
"/configuration/configureResponse/bomItem");

payloadTemplate = urldatabyget("https://cpq-
046.us.oracle.com/bmfsweb/slc10xgj/image/CommerceCloud/AddToCartPayload-
Cloud.txt", "", "");
model1 = "";
totalPrice1 = "";

// Get Model data
for model in models {
  model1 = get(outputModel, model);
}

// Get Price data
for totalPrice in totalPrices {
  totalPrice1 = get(outputPrices, totalPrice);
  totalPrice0 = replace(totalPrice1, ",", "");
  if (isnumber(substring(totalPrice0, 1))) {
    totalPrice2 = getcurrencyvalue(totalPrice1, currency);
    priceTotal = priceTotal + totalPrice2;
  }
```

```
}
baseModelPrice = priceTotal;

// Add BOM total price
if (containskey(bomPrices, bomTotals[0])) {
  for bomPrice in bomTotals {
    bomTotal = get(bomPrices, bomPrice);
    bomTotalReplace = replace(bomTotal, ",", "");
    if (isnumber(substring(bomTotalReplace, 1))) {
      bomTotalPrice = getcurrencyvalue(bomTotal, currency);
      priceTotal = bomTotalPrice + priceTotal;
    }
  }
}

// Get ConfigID
configId = "";
for id in configIdSearch {
  configId = get(outputConfigIds, id);
}

// Get Recommended Items
for sparepath in sparepaths {
  if (find(sparepath, "part") <  > -1) {
    part = get(output1, sparepath);
  }
  elif(find(sparepath, "quantity") <  > -1) {
    quantity = get(output1, sparepath);
  }
  elif(find(sparepath, "price") <  > -1) {
    price = get(output1, sparepath);
  }
```

```
  elif(find(sparepath, "selected") <  > -1) {
    selected = get(output1, sparepath);
  }
}

// Format Rec Items payload
recItemList = "";
if (isnull(part)) {
  print("No Recommended Items");
} else {
  recItems = sizeofarray(part);
  recItemsInt = integer[recItems];

  i = 0;
  for recItem in recItemsInt {
    if (selected[i] == "true") {
      //recurring price from parts BMQL
      part_num = part[i];
      partCustomFields = bmql("SELECT part_number, custom_field5,
custom_field4, custom_field6, custom_field8 FROM _parts WHERE part_number =
$part_num");
      recItemPayloadTemplate = urldatabyget("https://cpq-
046.us.oracle.com/bmfsweb/slc10xgj/image/CommerceCloud/Recommended_Items_Paylo
ad-Cloud.txt", "", "");
      recItemPayloadTemplate = replace(recItemPayloadTemplate, "{{quantity}}",
quantity[i]);
      recItemPayloadTemplate = replace(recItemPayloadTemplate, "{{part}}",
part[i]);

      for each in partCustomFields {
        if (get(each, "custom_field8") == "Recurring") {
          recItemPayloadTemplate = replace(recItemPayloadTemplate,
"{{pricePeriod}}", get(each, "custom_field4"));
```

```
          recItemPayloadTemplate = replace(recItemPayloadTemplate,
"{{recurringPrice}}", get(each, "custom_field5"));
          recItemPayloadTemplate = replace(recItemPayloadTemplate,
"{{duration}}", get(each, "custom_field6"));
          //recurringSubtotal = recurringSubtotal + get(each,
"custom_field5");
        } else {
          childPayloadJson = json(recItemPayloadTemplate);
          jsonremove(childPayloadJson, "recurringCharge");
          recItemPayloadTemplate = jsontostr(childPayloadJson);
        }
      }

      //remove region specific formatting for price
      sPrice0 = substring(price[i], 1);
      sPrice0 = replace(sPrice0, ",", "");

      if (isnumber(sPrice0)) {
        priceTotal = priceTotal + atof(sPrice0);
        recItemPayloadTemplate = replace(recItemPayloadTemplate, "{{price}}",
sPrice0);
      } else {
        recItemPayloadTemplate = replace(recItemPayloadTemplate, "{{price}}",
"0");
      }
      if (recItemList == "") {
        recItemList = recItemPayloadTemplate;
      } else {
        recItemList = recItemList + "," + recItemPayloadTemplate;
      }
    }
    i = i + 1;
  }
```

```
}

// Get the BOM Items
if (isnull(bomItemString)) {
  print "No BOM Items";
  bomItemString = "";
  payloadTemplate = replace(payloadTemplate, "{{BomItems}}", bomItemString);
} else {
  // Get part numbers for each BOM item, convert to string array for bmql
  bomJson = json(bomItemString);

  // Remove extraneous BOM fields (may have to revert if CC was expecting to
use them)
  jsonpathremove(bomJson, "$..variableName");
  jsonpathremove(bomJson, "$..definition");
  jsonpathremove(bomJson, "$..category");

  // Replacing all 0 prices with actual number 0
  bomPriceArray = jsonpathgetmultiple(bomJson, "$.._price_unit_price_each");
  replace_lookup = boolean[];
  bomPricesString = jsonarraytostr(bomPriceArray);
  bomPricesString = replace(replace(replace(bomPricesString, "\"", ""), "[",
""), "]", "");
  bomPricesStringArray = split(bomPricesString, ",");

  i = 0;
  for each in bomPricesStringArray {
    append(replace_lookup, isnumber(each));
    i = i + 1;
  }

  i = 0;
  for each in replace_lookup {
```

```
    if (i == 0 and each == false) {
      jsonpathset(bomJson, "$.fields._price_unit_price_each", "0");
    }
    elif(each == false) {
      str = "$.children[" + string(i - 1) + "].fields._price_unit_price_each";
      jsonpathset(bomJson, str, "0");
    }

    i = i + 1;
  }

  bomItemString = jsontostr(bomJson);
  bomPartsArray = jsonpathgetmultiple(bomJson, "$..partNumber");
  bomPartsString = jsonarraytostr(bomPartsArray);
  bomPartsString = replace(replace(replace(bomPartsString, "\"", ""), "[",
""), "]", "");
  bomPartsStringArray = split(bomPartsString, ",");
  bomParts = bmql("SELECT part_number, custom_field5, custom_field4,
custom_field6, custom_field8 FROM _parts WHERE part_number IN
$bomPartsStringArray");

  // Get path for each part, add recurringCharge to them all
  for each in bomParts {
    partField = "\"partNumber\":\"" + get(each, "part_number") + "\",";
    recurringTemplate = "\"recurringCharge\":{
\"amount\":,\"frequency\":,\"duration\":},";

    if (get(each, "custom_field8") == "Recurring") {
      recurringTemplate = replace(recurringTemplate, "frequency\":",
"frequency\":\"" + get(each, "custom_field4") + "\"");
      recurringTemplate = replace(recurringTemplate, "amount\":",
"amount\":\"" + get(each, "custom_field5") + "\"");
      recurringTemplate = replace(recurringTemplate, "duration\":",
"duration\":\"" + get(each, "custom_field6") + "\"");
```

```
    } else {
      recurringTemplate = "";
    }
    bomItemString = replace(bomItemString, partField, partField +
recurringTemplate);
  }

  // Unflatten
  bomItemString = replace(bomItemString, "\"partNumber\":",
"\"catalogRefId\":");
  bomItemString = replace(bomItemString, "On Request", "0"); // This may only
fix English users
  bomJson = convertbomtohier(json(bomItemString));
  payloadTemplate = replace(payloadTemplate, "{{BomItems}}",
jsontostr(bomJson));
}

// Format main template with subcomponents and properties
payloadTemplate = replace(payloadTemplate, "{{commerceItemId}}", "");
payloadTemplate = replace(payloadTemplate, "{{ConfigId}}", configId);
payloadTemplate = replace(payloadTemplate, "{{model}}", model1);
payloadTemplate = replace(payloadTemplate, "{{totalPrice}}",
string(priceTotal));
payloadTemplate = replace(payloadTemplate, "{{basePrice}}",
string(baseModelPrice));
payloadTemplate = replace(payloadTemplate, "{{currency}}", currency);
payloadTemplate = replace(payloadTemplate, "{{ChildItems}}", recItemList);
return payloadTemplate;
```

Appendix E: SyncQuote BML

```
str = "";

for each in transactionLine{
    if (each._model_variable_name <> ""){
        lineItem_array = split(cC_LineItem_Data_t, "|");
        for lineItem in lineItem_array {
            row = split(lineItem, "~");
            if(row[0] == each._document_number){
                str = str + each._document_number + "~cC_CommerceItemId_l~" +
row[1]+"|";
                str = str + each._document_number + "~cC_ProductId_l~" +
row[2]+"|";
            }
        }
    }
}

return str;
```

Oracle is committed to developing practices and products that help protect the environment

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Integrated Cloud Applications & Platform Services