Using Oracle CPQ Features with Oracle CX Commerce

F37074-01
January 2021

Using Oracle CPQ Features with Oracle CX Commerce,

F37074-01

Contents

1 Introduction

Many important Oracle CPQ features are available via an integration solution between Oracle CPQ andOracle CX Commerce.

This document is intended to provide the instructions on how to use Oracle CPQ features with Oracle CX Commerce - via an integration supported by the two solutions.

Oracle CX Commerce is an eCommerce solution designed specifically to run in the Oracle Cloud. The service provides you with a range of powerful tools to build a flexible, feature-rich storefront for your shoppers.

Activities you can perform with Oracle CX Commerce include the following:

- Customize the design and layout of your storefront pages and preview your changes
- Display your store content in different languages
- Create or import catalog items
- Manage inventory
- Offer promotions
- Manage shopper accounts
- Allow shoppers to set up wish lists
- View reports about your store
- Test the visual elements of your store to determine which design shoppers prefer
- Develop custom features for your store through the Oracle CX Commerce web services API

Oracle CPQ is the only cloud solution to support the complete quote-to-cash process - from shopper inquiry to order fulfillment. It guides users to optimal product options and configurations from simple to complex, automatically applying discounts and relevant up-sell and cross-sell opportunities.

Integrating these solutions brings together the capabilities of Oracle CX Commerce and Oracle CPQ to provide a unified solution that enables businesses to offer shoppers a method of interacting meaningfully with the business during the purchasing process, and to provide agents with the means to be flexible with shoppers, improving their contact experience and maximizing shopper satisfaction.

Objective

By integrating Oracle CX Commerce and Oracle CPQ, you increase the number of supported available commerce shopper features.

The integration of Oracle CX Commerce and Oracle CPQ targets support for the following shopper commerce activity:

- **Product configuration**: The shopper or agent can configure any product that has been identified as configurable in the product catalog.
- **Shopper quote request**: The shopper can request a quote for an order.
- **Agent quote request**: An agent dealing with a shopper contact can request a quote for a discount on behalf of the shopper.
- **Asset Based Ordering** - Asset based ordering (ABO) allows you to sell tangible assets or subscription services delivered over a period of time; for example mobile phone call and data plans, television and broadband packages, cloud storage service, music streaming service, etc.

This document provides instructions on how to set up an integration between Oracle CX Commerce and Oracle CPQ so that relevant Commerce information is automatically passed to Oracle CPQ, ensuring that the decision process has all the required information and increasing the speed at which a reply is delivered to the shopper or agent.

This document describes the setup tasks that must be performed in Oracle CX Commerce and Oracle Integration Cloud in order to use this integration flow. There are additional setup tasks that must be performed in Oracle CPQ so that the integration works as expected. Full information about these tasks that must be performed in Oracle CPQ can be found in the Integrating Oracle CX Commerce with Oracle CPQ article on My Oracle Support.

Chapter 2 – Configuring the Integration: provides technical instructions on the following topics:

- How to download the Oracle Integration Cloud Integration Flows.
- How to configure the Oracle Integration Cloud Integration Flows.
- How to setup the connection to Oracle CPQ.
- How to setup the connection to Oracle CX Commerce.
- How to configure the webhooks to trigger the integration flows.
- How to configure the SSEs (Side-Server Extension) necessary for the integration flows.

Chapter 3 – Using the Integration Functionality: provides instructions on how to use the functionality supported by this integration.

Audience

You must follow product-provided documentation to set up and configure the integration between Oracle CX Commerce and Oracle CPQ systems.

This document is written for Oracle CX Commerce and Oracle CPQ administrators who need to set up and configure the integration between these two systems.

Readers of this document should have experience with Oracle CX Commerce, Oracle CPQ and Oracle Integration Cloud (OIC) administration. This document does not provide instructions on configuring aspects other than the integration for Oracle CX Commerce and Oracle CPQ.

Prerequisites

In order to configure and use the Oracle CX Commerce/Oracle CPQ integration, there are specific software, account, and data prerequisites that must be met.

For the purposes of this document, it is assumed that you already have:

- An Oracle CX Commerce account and access to the Oracle CX Commerce 19.1 or later with necessary SSEs enabled (see sections that follow).
- An Oracle CPQ account and access to Oracle CPQ 19.1 or later.
- An Oracle Integration Cloud account and access to Oracle Integration Cloud Service 18.4.5 or later.
- A synchronized product catalog to ensure that products in the Commerce catalog map to corresponding items in the Oracle CPQ catalog.
- Pricing Base pricing data which is synchronized from the primary PIM (Product Information Management)/ERP (Enterprise Resource Planning system to both Oracle CX Commerce and Oracle CPQ.
- Profiles Shopper/Account data which is synchronized from the primary CRM (Customer Relationship Management) system to both Oracle CX Commerce and Oracle CPQ.
- An extension server to support any required Serve-Side Extensions for the integration.

If you do not have one or more of these, please contact an Oracle sales representative for information on how to acquire one: http://www.oracle.com/us/corporate/contact/index.html.

Additional Resources

Addition information about Oracle CX Commerce can be found through the Oracle Help Center page for Oracle CX Commerce.

If you require further information regarding Oracle CX Commerce, you can access the latest product documentation and training videos through the Oracle Help Center page for Oracle CX Commerce.

If you require further information regarding Oracle CPQ, you can access the latest product documentation through the forOracle Help Center page Oracle CPQ.

The documentation mentioned contains links to blogs, developer communities, and Support. (Please note that some of these resources require an account for access.)

2
Configure the Integration

Several stages are required to configure this integration.

Five stages are required to configure the integration between Oracle CPQ and Commerce. Each stage is covered in this chapter.

Configure the Integration Package

In order to use this integration, you must first download the integration package(s) and then import the package(s) into Oracle Integration Cloud.

This section provides detail about where the integration package(s) can be downloaded and how to import the integration package.

Importing the integration package in Oracle Integration Cloud (OIC) creates connections between Oracle CX Commerce and Oracle CPQ in OIC. It also creates an integration between Commerce and Oracle CPQ with some default mappings in place.

Download the integration package

Follow these steps to download the integration package:

1. Go to the Integrating Oracle CX Commerce and Oracle CPQ with Oracle CPQ article on My Oracle Support.
2. If you want to implement the integration between Commerce and the Oracle CPQ Configurator, download `OCCS-CPQ_CONFIGURATION_INTEGRATION_X.X.par` to a location where it is accessible from OIC.
   Note: `_X.X.par` refers to the most recent version of all downloadable files described.
3. If you want to implement the integration between Commerce and Oracle CPQ 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 and Oracle CPQ, download the following packages to a location that is accessible from OIC:
   - `OCCS_CPQ_ASSET_INTEGRATION_X.X.par`
   - `OCCS_CPQ_GETCONFIGBOM_X.X.par`
   - `OCCS_CPQ_CONFIGURATION_INTEGRATION_X.X.par`
   - `OCC_CPQ_Get_Asset_Upgrade_Options_X.X.par`

Import the integration package(s)

Import the OIC Integration Package into OIC to create an integration between Commerce and Oracle CPQ through OIC.

To import the OIC Integration Package:

1. Log on to OIC as an admin user.
2. Click the **Packages** icon.
3. Click the **Import** button.
4. Click **Browse** to open a navigation pane.
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_CONFIGURATION_INTEGRATION` package includes the OCCS-CPQ Get Configurations integration flow. The GetConfigurations integration flow is used for the following Asset Based Ordering operations:

- Modify
- Upgrade
- Renew
- Resume

This integration is required for the configuration flow. 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_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 Oracle CPQ.
- The OCCS-CPQ Update Quote integration sends information to Oracle CPQ related to accepting, rejecting, or re-requesting a quote.
- The OCCS-CPQ Sync Quote integration allows Oracle CPQ to send information to Commerce at the end of the quoting process and synchronize this information in Commerce. This ensures that the order information in Commerce matches the related order information in Oracle CPQ.

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 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.”

Note: The OCCS-CPQ Get Assets integration returns information about assets and services associated with the shopper’s account(s).

The `OCCS_CPQ_GETCONFIGBOM` package contains the following OIC integration flow which is also used in Asset Based ordering:

- `GetConfigBom` - This flow is invoked for the following Asset Based Ordering operation flows:
  - Suspend
  - Terminate

`GetConfigBom` calls are required to be made for each `configuratorID` of these filtered items to retrieve a saved Configuration BOM Instance of the item on Oracle CPQ.

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.”

Configure the Oracle CX Commerce Connection

For the integration to run successful, you need to configure the connection from the integrations imported to OIC to Commerce.

You must complete the following steps to configure the connection from the OIC integrations to Commerce.

1. Log on to OIC as an admin user.
2. Click the **Connections** icon.
3. Click the Oracle CX Commerce connection.
4. Click the **Configure Connectivity** button.
5. Enter the Connection base URL. The Connection base URL is derived using the following structure where <siteURL> is the base URL and port number of the Oracle CX Commerce site that integrates with OIC. For example:

   ```
   Connection base URL: https://<siteURL>/ccadmin/v1
   ```

6. Click the **Configure Security** button.
7. The Oracle CX Commerce connection uses the OAuth security policy, so you must enter a Security token for the connection. This token is generated in Oracle CX Commerce. Instructions on generating the token can be found in the next Generate a Security Token section of this document.
8. Click **OK**.
9. Click **Test** to test that the connection is working.
10. Click **Save**.

Your Oracle CX Commerce connection is now configured for the integration.

Generate a Security Token

This integration uses the Oracle CX Commerce REST web services APIs to access Oracle CX Commerce data. You must register the integration within Oracle CX Commerce and generate a security token in order for the integration to be granted access to the data.

Follow these instructions in order to generate a security token:

1. Log onto Oracle CX Commerce.
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 the **Register Application** button.

7. Enter a name for the integration. The application you are registering is OIC, so you should choose a meaningful name that reflects this.
8. Click **Save**. The Application ID and Application Key are automatically generated and the application is added to the Registered Applications page.
9. Click on the name of the application you created.
10. Click on **Click to reveal** to display the application key. You can copy the application key to use as the security token for the Oracle CX Commerce connection.

For more information on managing an application within Oracle CX Commerce, please refer to Register applications.

Activate the Integrations

Once your integrations are configured, you must activate them using the OIC admin user interface.

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

Follow these instructions to activate the OIC integrations:

1. Log on to OIC as an admin user.
2. Click on the **Integrations** icon to display the Integrations list.
3. Click on the **Activate** button for the integration you wish 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 it and activate it again to switch detailed tracing on or off.
5. Click **Activate**.

Configure the Commerce Webhooks

You must configure webhooks in Commerce Administration in order to support the REST API generated by the activation of the OIC integration.

The REST API generated by activating the OIC integration can be configured as a Webhook in Commerce Administration. These webhooks include the following:

- **Request Quote**: This webhook is triggered when a request or re-request for a quote is submitted by a Commerce self-service user. This 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, rejected, or the quote is canceled by a Commerce self-service user. This webhook pushes notifications using the OCCS-CPQ Update Quote integration flow.

- **External Price Validation**: This webhook is triggered at checkout when the order contains one or more items configured by Oracle CPQ. This webhook should point to the SSE app URL configured later. The webhook validates the configuration and price provided for the configured items. It also includes the commerce item ID data in the request payload and updates the external price information of the commerce items. Finally, it invokes a re-pricing operation at order checkout.
- **Contact Accounts Retrieval**: This webhook has been deprecated. The corresponding SSE endpoints are invoked from the widget. It returns a list of service account IDs for the shopper. Formerly, this webhook called the Contact Accounts Retrieval webhook, so that webhook also had to be configured for the Services Retrieval webhook to function correctly.
- **Services Retrieval**: This webhook has been deprecated. The corresponding SSE endpoints are invoked from the corresponding widget. Formerly, this webhook returned information about a service or asset associated with the shopper and used the OCCS-CPQ Get Assets integration flow. This webhook called the Contact Accounts Retrieval webhook, so that webhook also had to be configured for the Services Retrieval webhook to function correctly.

You must configure the Production and Preview version of these webhooks to ensure that they work in all environments. The Production webhooks send information from your live store to production environments of your live systems, while preview webhooks send information from your preview environment to the test or sandbox environments of your external systems.

Follow these instructions to configure the Request Quote, Update Quote, External Price Validation, Services Retrieval, and Services webhooks:

1. Log on to OIC as an admin user.
2. Click on the **Integrations** icon.
3. Click on the **Integration Details** icon to display information about the integration flow.
    - If you are configuring the Request Quote webhook, you should display information for the OCCS-CPQ Create Quote integration flow.
    - If you are configuring the Update Quote webhook, you should display information for the OCCS-CPQ Update Quote integration flow.
    - If you are configuring the External Price Validation webhook, you should display information for the OCCS-CPQ External Pricing integration flow. For this webhook, you to configure the SSE app endpoint.
    - If you are configuring the Services Retrieval webhook, you should display information for the OCCS-CPQ Get Assets integration flow. This OIC flows requires the Services SSE to be set up and invoked from there.
    - If you are configuring the Services webhook, you should display information for the OCCS-CPQ Asset Actions integration flow. This OIC flows requires the Services SSE to be set up and invoked from there.
4. Copy the Endpoint URL for the integration.
5. Log into Commerce.
6. Click the **Menu** icon.
7. Select **Settings** from the menu.
8. Select **Web APIs** from the sidebar menu.

9. Click the webhook you wish to configure.
10. Paste the Endpoint URL you copied into the URL field for the webhook.
11. Remove the “metadata” text from the end of the URL.
12. Enter the user name and Password for your OIC account.
13. Click the **Save** button.

The webhook is now configured and is triggered each time the relevant event occurs, which in turn triggers the relevant integration flow.

Note: It is not possible to edit webhooks differently for different sites. Updating webhooks applies changes regardless of the site selected.

For more information on Oracle CX Commerce webhooks, please refer to Configure webhooks.

Understand the Services SSE

Modify, renew, terminate, suspend, and resume actions performed on a service or asset are done using the Services server side extensions, one set for Storefront and one for Agent. Get Assets and Get Asset details are also performed using the endpoints in the Services SSE.

See the topic Configure the Server Side Extensions for information.

Configure the Server Side Extensions

To perform specific functions relating to asset-based orders, you need to install and configure the related Commerce server-side extensions (SSEs).

Available Commerce server-side extensions (SSEs) can be installed and configured to perform specific functions relating to asset-based orders.

For more complete information on server-side extensions and how to develop them for use with Commerce, refer to Develop server-side extensions in *Extending Oracle CX Commerce* found in the Commerce Help Library.

The next sections in this topic explain the purpose and configuration of each available SSE as well as provide information on the inputs required for their respective endpoints. Finally the last section of this topic, **Understand the general procedure for installing and configuring the integration SSEs** , provides general instruction on downloading, installing, and configuring the available SSEs.

Note: Address information is something used extensively in Commerce transactions. For all procedures and SSEs that require address information for endpoint inputs, in addition to using Commerce's default address formats, you can also use the REST API to create multi-country custom address formats. This lets you create country-specific address formats to ensure that your address formats align with the requirements of any external service that you might use. This means that addresses appearing in profiles, accounts, registration requests, order addresses and more can be customized. For more complete information on creating custom addresses and understanding how to use custom address formatting, refer to the following:

- Customize Address Formats using the API in *Extending Oracle CX Commerce*
- Work with address types in *Extending Oracle CX Commerce*
- Account Details in *Using Oracle CX Commerce*

- Work with account addresses in *Using Oracle CX Commerce*
- Work with account registration requests in *Using Oracle CX Commerce*

Configure the Credit Check SSE

Since Commerce does not provide a pre-built integration with any particular credit checking system, 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.

Note: This SSE is optional and can be used if you want a credit check to be done as part of an order submit task.

You can configure the available SSEs, **CheckCredit-store.zip** and **CheckCredit-agent.zip**, by first downloading the SSE packages.

Note: As written, this SSE generates outbound calls to an external credit checking system. This means that the Credit Check SSE calls out to an external system to perform the credit check. In order to use this SSE to connect to the external checking of your choice, you must modify the SSE code to provide the specific calls needed to connect to the correct credit checking system.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Check Credit endpoint

The Check Credit endpoint is triggered whenever a credit check is requested by Commerce.

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.

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. In detail, this SSE is meant to get account details from CDM masters like OEC Communications and is required in Telco kind of installations

You can configure the available SSEs, **CustomerAccountModel-store.zip** and **CustomerAccountModel-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

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.

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 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.

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 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.

It also validates that for any item in the order which is based on a SKU where the configurable property is TRUE and the `assetable` property is TRUE the quantity must be 1 and, if not, return an error indicating that this item can only be purchased one at a time. This check is done by looking to see if the root item has an `assetKey` value. For more information, see the Use Asset Based Ordering section of this guide.

You can configure the available SSEs, **OrderQualification-store.zip** and **OrderQualification-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring integration SSEs** section at the end of this topic.

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 Qualification 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 enables an order qualification step in the purchasing process that can be invoked via the Order Qualification webhook. The extension can be configured to execute custom order qualification processes such as checking whether the shopper is eligible to purchase the items in the cart. It contains a pre-built algorithm to validate that the Customer, Billing, and Service accounts as well as the Billing Profile assigned to the items in the cart are valid for the

logged in shopper. It also contains a module to check if the cancel in-flight is allowed for a given order.

You can configure the available SSEs, **OrderQualificationPipeline-store.zip** and **OrderQualificationPipeline-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Order Qualification Pipeline 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 Order Validation Pipeline SSE

This SSE enables an order qualification step in the purchasing process that can be invoked via the Order Validation webhook. The extension can be configured to execute any final checks particular to the purchasing model before the order payment is authorized and the order is submitted to the downstream systems for fulfillment and provisioning.

You can configure the available SSEs, **OrderValidationPipeline-store.zip** and **OrderValidationPipeline-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

Configure the Services SSE

The Services SSE is used to perform modify, renew, terminate, suspend, and resume actions on a service or asset - one SSE for Storefront and one for Agent. The SSE also contains a module to check if the cancel in-flight feature is allowed for a given order and is also used to retrieve the assets and asset details

You can configure the available SSEs, **Services-store.zip** and **Services-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

The subsection(s) that follows describe the relevant endpoint(s) for this SSE.

Understand the Services SSE endpoints

The Server Side Extension Endpoints for the Services SSE are the following:

- Modify
- Renew
- Terminate
- Suspend
- Resume

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

The inputs for these endpoints are:

- Logged in User Token
- `AssetKey`, 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.

Configure the Configuration Validation SSE

The Configuration Validation SSE plays an important role in Asset Based Ordering and validating asset configuration. This specific SSE performs a configuration validation between items in a shopper's cart and the items captured in response to configuration validation end points. For more complete information on Asset Based Ordering, refer to the Using the Integration Functionality section of this document.

To use this SSE, you should first have the External Pricing webhook set to `/ccstorex/custom/v1/validateCPQConfigurations`. This is done on the Settings page of the Administration user interface.

You should also have the following endpoints configured:

- `GET_CONFIGBOM_URI`
- `GET_CONFIG_URI`

The `GET_CONFIGBOM_URI` URL gets triggered for the Suspend and Terminate Services. The `GET_CONFIG_URI` URL gets triggered for the Renew, Modify, and Resume Services. The SSE does validation between items in cart and items captured in the response of these two end points

You can configure the available SSEs, **Services-store.zip** and **Services-agent.zip**, by first downloading the SSE package.

To complete installing and configuring the SSE, refer to the **Understand the general procedure for installing and configuring the integration SSEs** section at the end of this topic.

Understand the general procedure for installing and configuring the integration SSEs

To use this integration, you need to install and configure the integration server-side extensions (SSEs). The SSE code logic allows communication between Commerce and Oracle CPQ - via Oracle Integration Cloud as part of the data flow. The Commerce and Oracle CPQ integration functionality/communication is provided through the configuration of these server-side extensions.

In addition to providing REST APIs and webhooks for integrating with external systems (as well as widgets for extending your storefront), Commerce also includes support for

developing server-side extensions written in JavaScript. For more information, refer to Working with Commerce Server-Side Extensions

The general installation and configuration procedure for the integration SSEs uses the following steps:

- Before you configure and install the integration server-side extensions, first make sure your custom **Node.js** server is associated with your Commerce environment.
- Download the integration server-side extension (SSE) files locally, so that you can install and configure them. Select and remember the desired location where you want the SSE .ZIP file(s) to be downloaded. See Integrating Oracle CX Commerce and Oracle CPQ (Doc ID 2214316.1) on the My Oracle Support site for more information on the required integration SSE .ZIP files and for the links that let you download these files.
- After downloading the required files, you need to install them. Use the `POST /ccadmin/v1/serverExtensions` endpoint to do this. Specify the Content-Type as `multipart/form-data` and include a reference to the file in the body of the request. For example, your request header might look like the following:

  ```
  POST /ccadmin/v1/serverExtensions  HTTP/1.1
  Content-Type: multipart/form-data
  Authorization: Bearer <access_token>
  ```

- The request body should consist of the `<YOUR_SSE_NAME>.zip` file, uploaded as `multipart/form data`. The response to the request should look similar to this:

  ```
  {
      "result": {
          "unzipped": false,
          "failedImages": 0,
          "allImagesFailed": false,
          "failedImagesReasons": {},
          "modifiedImages": 0,
          "newImages": 1,
          "assignedImages": 0
      },
      "success": true,
      "links": [
          {
              "rel": "self",
              "href": "http://myserver.example.com:7002/ccadmin/v1/
  serverExtensions"
          }
      ],
      "token": "d63c663af7f15_cd3d"
  }
  ```

- Make changes to each server-side extension's `config.json` file by providing the correct URLs to complete the SSE configuration portion of that integration. The typical steps used for working with the SSE code and making changes to the `config.json` file include the following:
  - Obtain and download the correct SSE .ZIP file.
  - Extract the SSE .ZIP file.

- Edit and save the `config.json` file.
- Zip the files using the original .ZIP file name as the original.

The following example shows some configuration information (in **bold**) that must be added to the `config.json` file for both the store and agent models of the SSE:

```
"hostname": "yourhostname.example.com",
"port": "7003",
"timeout": 50000,
"username_env_var": "YOUR_USERNAME",
"password_env_var": "YOUR_PASSWORD",
"QUERY_CONTACTS": "/ic/api/integration/v1/flows/rest/
OCC_OEC_GET_PROFILE_SSE/1.0/contacts",
"CREATE_CONTACT": "/ic/api/integration/v1/flows/rest/
OCC_OEC_CONTACT_CREATE_SSE/1.0/contacts",
"QUERY_ACCOUNTS": "/ic/api/integration/v1/flows/rest/
OCC_OEC_GET_ACCNT_DETLS_PROF_SSE/1.0/accounts",
"CREATE_ACCOUNTS": "/ic/api/integration/v1/flows/rest/
OCC_OEC_ACOUNT_CREATE_SSE/1.0/contacts/{currentContactId}/accounts",
"UPDATE_ACCOUNT": "/ic/api/integration/v1/flows/rest/
OCC_OEC_ACCOUNT_UPDATE_SSE/1.0/contacts/{currentContactId}/accounts/
{accountId}"
```

All of the example endpoint URLs (paths) specified in the example, starting from the "`QUERY_CONTACTS`" to the "`UPDATE_ACCOUNT`" keys, are coming from Oracle Integration Cloud and are necessary for a successful integration activation between Commerce and Oracle CPQ. The paths that you would use when editing your `config.json` files would be the ones specific to your SSE endpoints. The ones shown here are for example purposes only. Refer to each specific SSE section in this topic to obtain the correct SSE and endpoint information.

- Upload the modified SSE .ZIP file. To upload the file, click **Settings** then **Extensions**. On the Extensions page, click **Installed** and then **Upload Extension**. Select the location and name of the ZIP file.

Understand the environment variables supported by the integration SSEs

When communicating with Commerce via its REST APIs, you need to authenticate your requests using confidential information. The need to authenticate is not just limited to Commerce as many 3rd party services require the same. It is recommended that you do not store confidential information in extension files but that you use environment variables to maintain value confidentiality. In the previous example `config.json` file, environment variables are used to make username and password information confidential. Commerce SSEs include the `nconf` package which provides a hierarchical `node.js` configuration with files, environment variables, command-line arguments, and atomic object merging. Use the hierarchy provided by `nconf` to manage your configuration values and maintain different values for different environments used in your integration. You can also use environment variables to pass through API information you want protected. Refer to "REST API authentication" in the Commerce REST API documentation for more info on how to authenticate Commerce API calls.

The specific environment variables supported by the integration SSEs are the following:

Table 2-1 Integration SSE environment variables

SSE name	Supported variable name	Description
CustomerAccountModel-Store	CRM_USERNAME	Specifies the basic authentication username for the accounts integration. In this case, for Oracle Integration Cloud (OIC) which integrates OEC Comms.
	CRM_PASSWORD	Specifies the basic authentication password for accounts integration. In this case, for OIC which integrates OEC Comms.
CustomerAccountModel-Agent	CRM_USERNAME	Specifies the basic authentication username. In this case, for Oracle Integration Cloud (OIC) which integrates OEC Comms.
	CRM_PASSWORD	Specifies the basic authentication password for accounts integration. In this case, for OIC which integrates OEC Comms.
Services-Store	OIC_USERNAME	Specifies the basic authentication username for the accounts integration and Oracle CPQ integration that proxies via OIC.
	OIC_PASSWORD	Specifies the basic authentication password for the accounts integration and Oracle CPQ integration that proxies via OIC.
	CPQ_USERNAME	Specifies the basic authentication username for requests that go directly to Oracle CPQ.
	CPQ_PASSWORD	Specifies the basic authentication password for requests that go directly to Oracle CPQ.
Services - Agent	OIC_USERNAME	Specifies the basic authentication username for the accounts integration and Oracle CPQ integration that proxies via OIC.
	OIC_PASSWORD	Specifies the basic authentication password for the accounts integration and Oracle CPQ integration that proxies via OIC.

Table 2-1 (Cont.) Integration SSE environment variables

SSE name	Supported variable name	Description
	CPQ_USERNAME	Specifies the basic authentication username for requests that go directly to Oracle CPQ.
	CPQ_PASSWORD	Specifies the basic authentication password for requests that go directly to Oracle CPQ.
Order Qualification Pipeline	ORDER_QUALIFICATION_PIPELINE_USERNAME	Specifies the basic authentication username for securing the `/v1/orderQualification` route.
	ORDER_QUALIFICATION_PIPELINE_PASSWORD	Specifies the basic authentication password for securing the `/v1/orderQualification` route.
	VALIDATION_USERNAME	Specifies the basic authentication username for accessing the `/v1/crm/accounts` route in the Customer Account Model to retrieve accounts data.
	VALIDATION_PASSWORD	Specifies the basic authentication password for accessing the `/v1/crm/accounts` route in the Customer Account Model to retrieve accounts data.
	OIC_USER_NAME	Specifies the basic authentication username for accessing the services integration that integrates with Oracle CPQ to retrieve asset/services data.
	OIC_PASSWORD	Specifies the basic authentication password for accessing the services integration that integrates with Oracle CPQ to retrieve asset/services data.
Order Validation Pipeline	ORDER_VALIDATION_PIPELINE_USERNAME	Specifies the basic authentication username for securing the `/v1/orderValidation` route.
	ORDER_VALIDATION_PIPELINE_PASSWORD	Specifies the basic authentication password for the `/v1/orderValidation` route
cpq-configurator-app-store	CPQ_USERNAME	Specifies the basic authentication username for requests that go directly to Oracle CPQ.

Table 2-1 (Cont.) Integration SSE environment variables

SSE name	Supported variable name	Description
	CPQ_PASSWORD	Specifies the basic authentication password for requests that go directly to Oracle CPQ.
cpq-configurator-app-agent	CPQ_USERNAME	Specifies the basic authentication username for requests that go directly to Oracle CPQ.
	CPQ_PASSWORD	Specifies the basic authentication password for requests that go directly to Oracle CPQ.

Note: Commerce provides the `admin` endpoint that can be used to set an environment variable on the Commerce server. For additional information on each SSE's supported environment variables, a `README.TXT` file is provided along with the `config.json` file that has additional usage information.

Enable the Integrations

You need to enable the Oracle CPQ Configurator integration, the Oracle CPQ Request For Quote integration, and the Asset Based Ordering (ABO) integration in Commerce for the complete integration to work successfully.

You must complete the procedures in this section to enable the Oracle CPQ Configurator integration, the Oracle CPQ Request For Quote integration, and the Asset Based Ordering (ABO) integration in Commerce.

For additional information, refer to Appendix A: Configurator Flow and Appendix B: Request for Quote Flow.

Enable Oracle CPQ Configuration Integration

Follow these steps to enable the Oracle CPQ Configuration Integration within Oracle CX Commerce:

1. Log on to Commerce.
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 dropdown menu.
6. Check the **Enable Integration** checkbox.
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. You must enter these values for your production and preview environments.

   ```
   https://<cpq_domain>/commerce/new_equipment/products/
   external_reconfig.jsp
   ```

9. Enter the Modification URL using the following structure. You must enter these values for your production and preview environments.

   ```
   https://<cpq_domain>/commerce/new_equipment/products/
   model_configs.jsp
   ```

10. Click the **Save** button.

If you are using multiple sites you must follow these instructions for each site that you operate that uses the Oracle CPQ Configuration integration.

Identify Configurable Products in the Product Catalog

Before a Commerce self-service user can use the Oracle CPQ configurator to configure complex products for purchase in Commerce, you 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 catalog map to corresponding items in the Oracle CPQ catalog.

To identify a product as configurable:

1. Log in to Commerce.
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 Oracle CPQ catalog.
9. Enter the Product Line information. This should match the Product Line information of a configurable product in the Oracle CPQ catalog.
10. Enter the Product Family information. This should match the Product Family information of a configurable product in the Oracle CPQ 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 page in Commerce, 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 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.
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 Oracle CPQ Quoting Integration

Follow these steps to enable the Oracle CPQ Quoting Integration within Oracle CX Commerce:

1. Log on to Commerce.
2. Click the **Menu** icon.
3. Select **Settings** from the menu.
4. Select **Oracle Integrations** from the sidebar menu.
5. Select **CPQ Quoting** from the dropdown box.
6. Check the **Enable Integration** checkbox.

Add Quote Buttons to the Checkout and Order Details Pages

To make the Oracle CPQ quoting capability available to Commerce self-service users, you 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. You 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.

2. Click the **Menu** icon.
3. Select **Design** from the menu.
4. Select the **Components** tab on the Design page.
5. Click the **Show Hidden** button.
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 Webhooks and Configure the Server Side Extensions topics of this document.

3
Use the Integration Functionality

Oracle CPQ provides greater pricing and price quoting features for Oracle CX Commerce when the two are used together in an integration.

This chapter provides the instructions on how to use this functionality in Oracle CX Commerce that is supported by the integration with Oracle CPQ.

Configure an item

Items marked as configurable in your catalog can be configured either by an agent via the Commerce Agent Console or by a shopper via the Commerce Storefront.

Items that have been marked as configurable in your catalog may be configured either by an agent via the Commerce Agent Console, or by a shopper via the Commerce Storefront. This section provides instructions for both methods of configuring an item.

Configure an Item by an Agent

These instructions detail how an agent can configure an item via the Agent Console.

1. Log onto Commerce.
2. Using Agent Console, search for the shopper for whom you wish to create a new order.
3. Click **New** to create a new order.
4. Select a configurable product from the catalog.
5. Click on the **Configure** button to open the Oracle CPQ iFrame.
   Note: The Oracle CPQ iFrame is optimized for desktop, laptop, or tablet-size devices and is not recommended for mobile devices. If you need to display on mobile devices, please contact your Oracle CPQ Implementation team and inquire about the CPQ Mobile Layout.
6. Select the configuration options required for the order.
7. Click **Add** to Cart.

Once the configured item has been added to the cart, the agent can complete the order by going through the normal checkout process.

There is a validation check before the order is processed to ensure that the configuration options selected are valid. If they are valid, the order process completes and the order is placed. If they are not valid, an error message is displayed to the agent telling them that the configuration is invalid and that the order cannot be placed.

Configure an Item by a Shopper

These instructions detail how a shopper can configure an item via Commerce Storefront.

1. Shopper selects a configurable item from the product catalog.

2. Shopper clicks on the **Customize** button which opens the CPQ iFrame.
3. Shopper selects their desired configuration options for the item.
4. Shopper adds customized item to their cart.
5. Shopper goes to checkout and provides shipping and payment details.

There is a validation before the order is processed to ensure that the configuration options selected are valid. If they are valid, the order process completes and the order is placed. If they are not valid, an error message is displayed to the shopper telling them that the configuration is invalid and that the order cannot be place. The shopper is then unable to place the order until the configuration options have been changed and the configured item passes the validation check.

Request a Quote

With Oracle CPQ enabled in the integration price quotes may be requested for one or more items.

Quotes may be requested for one or more items on an order either by an agent from within the Agent Console, or by a shopper from the checkout page for their order. If you are also using Oracle CPQ Configuration functionality, the order may contain a combination of configured and non-configured items.

Request a Quote by an Agent

An agent can request a quote on one or more items in an order from the Commerce Agent Console. The agent must follow these instructions to request a quote:

1. Log onto the Commerce Agent Console.
2. Search for the shopper for whom you wish to generate a new quote.
3. Click **New** to create a new order, or select an existing unfulfilled order for the shopper.
4. Once you have an order with items in the cart, click on the **Request Quote** link in the order edit page. You can switch between the **Request Quote** page and the **Create Order** page by clicking on the appropriate link.
5. Add text to the **Quote Notes** text box as desired.
6. Click on the **Request Quote** button.
   Once the agent has submitted the quote request, the Request Quote webhook is triggered and all relevant information is passed to Oracle CPQ for a decision on the quote. The order status is changed to “Pending quote”. When an order is in Pending status, the agent cannot perform any operations on the order.

   A confirmation email is sent to the shopper informing them of the status of their order.
7. Once a response is received, the order status changes to “This order is a quote”, and the agent then has a number of options about how to proceed. The agent can: The agent can:

   - **Accept the quote**: If the shopper is satisfied with the quoted price returned from Oracle CPQ, the agent can accept the quote on their behalf by clicking on the Accept button and proceeding with the order as normal.
     Once payment information has been entered and the order placed the order status changes to “Submitted for fulfillment”. At this point the Update Quote

webhook is triggered and Oracle CPQ is informed that the quote has been accepted.

At this stage the agent can click on the Edit Order button, but the only edits allowed to the quote are changes to the shipping group, or the application of shipping discounts or promotions. The agent may not add or remove items from the cart, or change the quantities of items included in the order. The order status changes to “Order being amended” until the agent clicks on the Complete Order button.

- **Request a requote**: If desired, the agent can enter more details in the Request Quote textbox and click on the Request Requote button to request an updated quote. When the agent requests a requote the order status changes to “Pending quote”. When an order is in Pending status, the agent cannot perform any operations on the order.
- **Reject the quote**: The agent can click on the Reject Quote button to reject the quote. This cancels the shopper’s order and the order status changes to “this quote has been rejected”.
  Note: The response to a quote request includes provision for an expiry date for the quote. If the quote has expired the Accept Quote and Reject Quote buttons are disabled, but an agent can request a requote for the order.

  Once the agent responds to the quote a confirmation email is sent to the shopper informing them of the status of their quote.

Order statuses relating to quotes are included in the dropdown list of order statuses in the Order Details section of the Order Search page.

Request a Quote by a Shopper

A shopper can request a quote on one or more items in an order from the checkout page. The shopper must follow these instructions to request a quote.

1. Add the desired items to the shopping cart.
2. Proceed to the checkout page.
3. On the checkout page, enter supporting details in the **Request a Quote** text box.
4. Click the **Request Quote** button.

Once the shopper has submitted their quote request, the Request Quote webhook is triggered and all relevant information is passed to Oracle CPQ for a decision on the quote.

When a decision is made about the quote, the order is updated and the shopper then has three options about how to proceed.

They can:

- **Accept the quote**: This means the shopper is satisfied with the quote and they may continue through the purchase process with the prices provided. The checkout page is displayed and the shopper may enter their shipping details and proceed with payment.
- **Reject the quote**: This means that the shopper has rejected the quote provided by CPQ Cloud, and the order is canceled.
- **Request a requote**: The shopper can use the **Request Requote** text box to provide further information and request an updated quote.

Use account-specific pricing for configured items

Account-specific prices configured on Oracle CPQ can be displayed in Commerce.

Account-based shoppers can obtain account-specific prices configured on Oracle CPQ and display these prices in Commerce. This topic explains the concepts behind this feature.

Formerly, the Oracle CPQ iFrame would open in an item configuration as part of anonymous session. All details in the Oracle CPQ page, then, were independent of the logged in shopper/contact/account. Account-based shoppers can now obtain account-specific prices configured on Oracle CPQ and display these prices in Commerce. This is possible because the iFrame displayed on Commerce obtains the context of the related account as well the contacts associated with it so that the correct account-based pricing information is returned to Commerce.

For example, consider that an account-based shopper logs in and selects to purchase a configurable computer package. The prices that Oracle CPQ returns to the shopper are specific and unique to that shopper's account. The pricing that is specific for one shopper is not visible to another shopper. The shopper then changes the configuration of the computer package as needed, enters the quantities needed, and finally submits the order. In the case of an Agent configuring the package, the agent also sees the account-specific details when configuring a price.

With the Commerce/Oracle CPQ configuration integration enabled, Commerce sends different criteria to determine and obtain the account-based price of the configuration maintained on Oracle CPQ.

Understand a user case as well as the workflow used to obtain correct account-based pricing

An example of a user case where a shopper (or agent) would want to obtain account-based pricing could go something like this:

- The shopper selects a commerce site and browses through the items in the catalog.
- The shopper selects a configurable item and clicks on the Customize button, which opens an iFrame allowing them to customize/reconfigure the item.
- The shopper configures the item and expects to see the prices for the items which only that customer would be allowed to see for that account.
- The shopper places the order with the customer-specific pricing. The shopper, after submitting the order and within the designated remorse period, is able to update the configurations of the items as well as receive the customer specific pricing for those items.
- The shopper can cancel this order containing customer specific priced items (within the remorse period).
- The shopper can carry out returns and refunds.
- The shopper can exchange within the same configured item(s).

Some variations to this use case could include:

- The shopper gets the account specific pricing but when shopper account details change, the adjusted price specific to that account would appear.

- An agent placing an order for an account based shopper gets the account specific pricing specific to that shopper.

The workflow used to obtain the correct account-specific pricing is the following:

- The store sends the `Account ID(Customer ID, Customer Name)` through the Configure Product iFrame. The shopper's `accountId` has been encrypted using the SSE and the encrypted details are what is sent to the Oracle CPQ iFrame. Other properties like model, product line, locale, currency are not encrypted.
- The calls made to Oracle CPQ at this point internally call the Oracle CPQ Price API.
- The iFrame shows the account specific pricing for the account based on the `accountId`.
- The Price API looks for any customer pricing rules defined in Oracle CPQ and returns the correct account-specific pricing for that customer based on the accountId. If there are no prices configured specifically for the customer, then they are presented with the default prices.
- A sample widget can be customized by implementers to encrypt and pass additional properties along with the `accountId`. The re-configuration flow works as it already exists.

The main purpose of this workflow is to pass the customer account/organization details to the Oracle CPQ system and calculate the customer-specific price (if any pricing rules are defined).

The existing integration components should retain their existing functionality (i.e. the customer/system implementer should be able to switch as to whether they are using anonymous or customer specific pricing).

With this workflow, it is assumed that there is data synchronization of Customers (Commerce account-based customer accounts) across Oracle CPQ and Commerce. Oracle CPQ is the mechanism that has the ability to set up rule-based pricing which can be customer specific. The customer specific pricing rule(s) should be the source for the account-based pricing of the item. Finally, there is a check done that is part of the integration which makes sure that the logged-in user is validated.

Note: A customer can use Oracle CDM (Oracle Customer Data Management) to maintain that the accounts (organizations) are synchronized between Commerce and Oracle CPQ or they can just use the Commerce accounts. The `accountId` that is passed in the integration flow varies based on the implementation model.

Understand how Commerce and Oracle CPQ support account-specific pricing

To be able to obtain account-specific prices configured on Oracle CPQ and display it on Commerce via the returned iFrame, you need the iFrame to be extended to handle various attributes as part of getting the price from Oracle CPQ. By extending these attributes, you can then display the account specific pricing given by the Oracle CPQ system.

The `cpq-config-validation-app` SSE now validates the additional `accountID` from the `getConfiguration` call made to Oracle CPQ to find the profile associated with the order before calling the Submit Order endpoint. An appropriate error message is returned if the `accountId` does not match the values of the `orgId` of the profile in `order.json`. By passing these parameters from Commerce to Oracle CPQ during the Configuration Page launch, the Pricing logic in Oracle CPQ can be triggered within the Configuration user interface. Commerce provides the initial ability to pass the Account

ID, but an implementer can extend this to pass any other parameters from Commerce that Oracle CPQ can understand.

The integration takes an Access Token Based security approach to ensure that prices meant for users of one account are not visible to users of a different account. The key features of using this approach are the following:

- The authentication into Oracle CPQ continues to be an anonymous/ guest user method as it is today.
- There is no need for user mapping between the Commerce user and Oracle CPQ user as well as no need for additional user maintenance between Commerce and Oracle CPQ.
- The approach follows an established approach based on Assets Modify Punch-In.

SSE flow for Store and Agent

The following describes the SSE flow for Store and Agent

- When an account-based customer clicks on Customize for a product, the SSE endpoint gets triggered.
- The `accountId` of the account-based user and other configurable details like model, product_line, product_family, etc. get passed to the SSE.
- The validation of the `accountId` takes place first whether the logged in customer is associated with the `accountId` being passed or not.
- If the validation is successful, `accessTokenData` is generated containing the `accountId` and the expiration time which is then encrypted and signed with the private key to form the accessToken.
- A query string is formed using `accessToken` and another configurable list of parameters. This is then appended to the base URL and the Oracle CPQ iFrame that is launched with the account-based prices.
- The `accountId` is decrypted by Oracle CPQ using the Public key. The true `accountId` is then determined and prices are shown as per the pricing rule setup for this `accountId`.

The following illustrates a sample request:

```
{
  "accountId" : "or-100001",
  "configurableSkuDetails" : {
    "currency" : "USD",
    "locale" : "en",
    "model" : "sku50001",
    "product_line" : "laptopConfiguration",
    "product_family" : "Laptop"
  }
}
```

The following illustrates a sample response to that request:

```
{ "queryString":
"_bm_session_currency=USD&_bm_session_locale=en&model=sku50001&product_lin
e=laptopConfiguration&product_family=Laptop&segment=laptop&_from_partner=t
rue&accessTokenData=%7B%22expiryTime%22%3A%222019-11-06T15%3A40%3A49%2B05%
```

3A30%22%2C+
%22configAttrPunchinValues%22%3A%7B%22accountId%22%3A%22or-100001%22%7D%7D
&publicKeyVarName=shagul_rsa_public&accessToken=xboKIL0YMl1R1IERTBKzzfFbyV
AWq5bZgkWX%2Bf71YOJYlBu1GZ5aZay%2B5FS338joCIs8C7B9RrJlRXXkmd1U4zgqfPD2NJnf
bYzxCelhFpbwdau6n88qVH6WI%2BPCLzUJKrwJdNxuTd9O78ZL4pKW8g9mFhpnZcNec%2FRxpH
MrV%2BYm4S2iS5IZt7apTkt%2Bd%2BDDvm3Y0cmyEyfwcbhTjxKho904dJId0pf%2BU3VKcNIh
MRMtoeFFCskhQNiqA8gyjUqamyB1y%2BgZQ9WKqo84rYsPnjCHvOF5z%2BAjMF5FysbGQxLJAF
PAczACuLhn1XrmDjjYMD6T26ey2d%2BQbKlzGgMIsg%3D%3D" }

SSE flow for validating the account ID

The `cpq-config-validation-lib` SSE has the functionality to validate the `accountId` (part of External Data) from the `getConfiguration` call to Oracle CPQ with the organization ID of the account-based profile associated with the order before calling the Submit Order endpoint.

Understand best practices for using account-specific pricing

Although the integration allows for account-specific pricing, it does not, however, allow for re-pricing of configured items, when any of the following conditions occur:

- The price list group (currency) is changed by the shopper
- An anonymous user logs in which results in a change of price list group (for example, an anonymous shopper logs in as an account-based shopper)
- The account based shopper changes the current account in context

Simple (i.e., non-configured) items are re-priced but configured items are not. A shopper cannot even re-configure the item to get the updated price. This is because Oracle CPQ does not accept secure punch-in attributes during the process of re-configuring an item. Unfortunately, the only available option is to delete the configured item and add it again as a fresh configuration.

To avoid inconvenience to the shopper, it is recommended that you add an information message to your custom widgets. The message should be seen when an anonymous shopper tries to add a configured item to cart, (for example, when they click the Customize button or when they are on the Product Details page. The message should suggest that the shopper first login and then do the configuration.

Set up and configure Commerce and Oracle CPQ for account-specific pricing

Use the following procedures to set up and configure Commerce and Oracle CPQ for the account-specific pricing feature:

Configure Oracle Integration for account-specific pricing

Use the following steps to configure Oracle Integration for account-specific pricing while using the Commerce/Oracle CPQ integration:

- Download the Oracle Integration packages found in the `OCCS-CPQ_CONFIGURATION_INTEGRATION_4.0.par` package file from Oracle Marketplace or My Oracle Support.
- Import the package into the OIC Environment
- Configure the Oracle CPQ Connection in OIC.
- `getConfigurations` (4.0) is used to validate configuration and pricing from Oracle CPQ.

For more information on using Oracle Integration, refer to the product Help Library.

The next steps you need to complete are downloading and configuring the required Server Side Extensions (SSEs) used for account-specific pricing. There are three SSEs used by the integration to support this. These are the following:

- `cpq-config-validation-app.zip`
- `cpq-config-punchin-store.zip`
- `cpq-config-punchin-agent.zip`

The information that follows describes how to download and configure these SSEs.

Download and configure `cpq-config-validation-app.zip`

Use the following steps to download and configure the `cpq-config-validation-app.zip` Server Side Extension for account-specific pricing while using the Commerce/Oracle CPQ integration:

- Login as Administrator to Commerce
- From the Admin interface, download the Server Side Extension (SSE) `cpq-config-validation-app.zip` by clicking **Design - Developer - Server-Side Extensions**. This SSE triggers the Configuration integration just setup on Oracle Integration.
- Unzip the file.
- Update the `config.json` file with the Oracle Integration Hostname and Port information.
- Also, a the dd extension environment variables for `OIC_USERNAME` and `OIC_PASSWORD` using the `Admin` endpoint
- Run the Node Package Manager to install in the .ZIP contents in the root folder
- Zip the contents in the root folder.
- Upload to the Extension Server using the `serverExtension` endpoint

Download and configure `cpq-config-punchin-store.zip` and `cpq-config-punchin-agent.zip`

Use the following steps to download and configure the `cpq-config-punchin-store.zip` and `cpq-config-punchin-agent.zip` Server Side Extensions for account-specific pricing while using the Commerce/Oracle CPQ integration:

- From the Commerce Admin interface, download the `cpq-config-punchin-store.zip` and `cpq-config-punchin-agent.zip` Server Side Extensions by clicking **Design - Developer - Server-Side Extensions**. These use the `CPQ Punchin Lib` to generate a query string with an encrypted `accountID` that is passed to the CPQ iFrame. This enables CPQ to show account specific prices.
- Unzip the files.
- Generate a public and private key using the OpenSSL utility.
- Place the private key file in the **Keystore** folders.
- Specify the public key in the `config.json` files.
- The public key is also added to the CPQ Integration Center under **Authentication Certificate**.

- Run NMP to install the .ZIP files contents in the root folder
- Zip the contents in the root folder
- Upload to the Extension Server.

Configure Commerce for account-specific pricing with Oracle CPQ

Use the following steps configure Commerce for account-specific pricing while using the Commerce/Oracle CPQ integration:

- Set up the external pricing validation webhook in Commerce: `/ccstorex/custom/v1/validateCPQConfigurations`
  Do this by going to **Settings - Web APIS - Webhook** in the Admin user interface. Choose the **External Price Validation** function API and enter the requested information.
- Enable and configure the Commerce/Oracle CPQ Configuration Integration by going to **Settings - Oracle Integrations** in the Admin user interface. Select the Oracle CPQ integration from the list.
- Modify widgets in the following layouts:
  - Store layout: Product Details page
  - Agent layout: B2B Checkout Layout - Accordion element: Search and Add Items to Cart popup stack

  These take the `accountIDs` of the user and display the appropriate prices for that account.

You also need to have synchronized product catalogs between Oracle CPQ and Commerce. The models available in Oracle CPQ need to have a corresponding externally configurable SKU in Commerce. Also, make sure you have set up the accounts and desired users in Commerce and have set up the proper pricing rules in Oracle CPQ (since it is the provider of the prices to sent to Commerce).

Finally, double check that you have setup pricing rules on Oracle CPQ based on a parameter (for example, Account Id) and have also added the public key to the **Integration Center - Authentication Certificate** in Oracle CPQ.

Use multi-level items

This integration features support for a hierarchical structure for items available to shoppers for purchase.

This integration provides support for a hierarchical structure for items available for shoppers to purchase. Commerce supports an “n-level” hierarchical configuration model. This means that a configured item can contain sub-items that are also configurable items and that can in turn contain sub-items that are configurable items.

An example of this would be a bundled package for a cellphone. The bundle itself would be the top-level item. The cellphone would be a configurable sub-item, but this could then itself have configurable sub-items, such as an SD card. Commerce can provide a top-level price for the bundle, but can also provide a price breakdown for each configurable item within the bundle.

If a shopper adds a multi-level item to their cart, Commerce works with Oracle CPQ to display the information about the multi-level item in the shopper’s cart. The cart displays a total price and an item price for any configurable sub-items. If the shopper

changes any of the configurable sub-items, the price displayed for that sub-item changes and the total price is also amended accordingly.

When a shopper clicks on the Place Order button a validation check is carried out to ensure that the prices displayed for the configured items is still applicable. If it is then the order can proceed. If it is not, a message explaining this is displayed to the shopper and the cart is reloaded with up-to-date price information included for the configured items.

You can create a multi-level hierarchy in your catalog using either a recommended items model or a bill of materials model. You must refer to the relevant Oracle CPQ documentation for instructions on how to do this.

Use Quadplay/NPlay items

A standard, or single play, configured item represents a single service, such as Mobile Phone or IPTV that has a single set of configuration information, i.e. is based on a single configuration model in Oracle Oracle CPQ.

A Quadplay or NPlay configured item represents a package or bundle that combines multiple services in a single purchase and contains multiple sets of configuration information, i.e. is based on a single configuration model that also contains other configuration models in Oracle CPQ.

As an example, consider a case where the configured bundle contains 4 separate services (or ‘plays’) such as Landline, Internet, Mobile and IPTV. In this example, the bundle is called the Get4 Bundle. Unlike a standard configured item, the Get4 Bundle, as a Quadplay or NPlay configured item has configuration information at the following levels:

- Root level - in this example, the Get4 Bundle level.
- Branch level - in this example, the Landline, Internet, Mobile and IPTV levels.

With the support of Quadplay/NPlay configured items, the shopper adds the Get4 Bundle to the Oracle CX Commerce cart as a standard multi-level hierarchical configured item. This item also has the ability to be reconfigured if needed. The item is then validated and checked out as usual. For more detailed information working with Quadplay and NPlay items, refer to CX Communications - How to Customize and Extend – Configure NPlay Bundles with Oracle CPQ System Configuration white paper on the My Oracle Support site.

Understand Commerce Cloud Administration support for configuration metadata

In Oracle CPQ, a single model is also able to support multiple NPlay offers and additional versions of those offers. For example the same Model, Product Line, and Product Family might contain 3 variations on the same NPlay bundle such as the following:

- Starter Home Bundle
- Total Home Bundle
- Friends and Family Bundle

The same model might also support multiple versions of those bundle variations such

- Starter Home Bundle
- Starter Home Bundle 2017
- Starter Home Bundle 2018

and

- Total Home Bundle
- Total Home Bundle 2017
- Total Home Bundle 2018
  and
- Friends and Family Bundle
- Friends and Family Bundle 2017
- Friends and Family Bundle 2018

To work with these types of variations, when the shopper selects a version of a bundle in Commerce and chooses to configure it, the configuration request needs to include extra information to allow the configurator to load the correct version of the configuration model. This extra information is provided in what is called configuration metadata. This data is passed along as a collection of key value pairs and aid in helping to identify the correct bundle.

Understand configuration metadata details

Where a Oracle CPQ configuration model supports multiple products and product variations, this information may not be sufficient to pre-load the order iframe with the correct starting point. In such cases extra information (i.e., configuration metadata) can included in the iframe URL created by Commerce.

Again, think of configuration metadata as a collection of one or more key value pairs that identifies the correct starting point for the configuration model. Configuration metadata can be static or dynamic. Static configuration metadata is manually entered by the Commerce Cloud Administrator and stored on the SKU record in Commerce. Dynamic configuration metadata can be captured by the PDP UI widget and can be entirely implementation specific.

Note: Dynamic configuration metadata is not restricted to being captured on the PDP UI widget. The dynamic configuration can be derived from any relevant information such as shopper profile.

This means that merchants can decide what dynamic key value pair data they want to capture and pass in the configuration request for any SKU. Dynamic configuration metadata can be mandatory or optional (i.e., in some cases the shopper MUST enter a value for a key and in some cases they may optionally enter a value for a key).

Configuration metadata lets merchants define a single model for all variants of a configurable product and at purchase time pre-load the configuration model at the appropriate starting point based on the shopper's selection in Commerce.

The configuration metadata feature builds on the already existing support of the NPlay feature. Earlier there was support of the purchase of NPlay products but only where there is a one-to-one relationship between product and model (i.e., each NPlay product had to have its own unique corresponding configuration model in Oracle CPQ).

Enter configuration metadata via the administration user interface

To provide the configuration metadata needed for processing an order, the `configurationMetadata` property is exposed so that you can enter the information in the Commerce Cloud Administration interface. To get there click **Catalogue** then **Product** and finally **SKU**. This Administration panel lets you view, add, delete, and

edit the Configuration Metadata values as required. Any request from Commerce to configure an item will include configuration metadata where it is available.

An example of using configuration metadata might be a case where a Commerce Cloud Administrator receives an email from a colleague in Oracle CPQ to advise them that the configuration model with the correct configuration metadata for the Family Plan products SKUs is now complete. The email contains the information to further configure the SKU. The SKU is called sku_fp_001 and the information provided is the following:

- Product Family – mobile
- Product Line - bundles
- Model - sku_fp_001
- Bundle Version - 18.1
- Region - EMEA

The process for entering the configuration metadata via the Admin interface would go something like the following:

1. Navigate to the Commerce Cloud administration user interface panel and select **Catalog**.
2. Select the **Family Plan** product and select the SKU sku_fp_001 which is currently flagged as inactive.
3. Click on **Externally Configurable SKU**. You see the text “Oracle CPQ can configure this as a part of a complex product.”
   Note: For any SKU where you want to add configuration metadata, you must make sure that Externally Configurable SKU is checked when you first begin entering data. A new input property will be displayed which will allow you to begin to enter one or more key value pairs of data.
4. Slide the panel down until you see the Product Family, Product Line, and Model fields appear on the panel. Enter all of the correct metadata details (the ones sent to you in the email from your Oracle CPQ colleague) manually.
5. Slide the panel down to see the Configuration Metadata table, click the **Add** button to add a row.
6. Add Bundle Version to the **Name** field. In the field next to Bundle Version, add 18.1 (as the bundle version number). You can press **Tab** or **Enter** to create a value entry. Click the **Add** button when done. A new row in the metadata value table appears.
7. Add Region to the **Name** field. In the field next to Region, add EMEA. You can press **Tab** or **Enter** to create the value entry.
8. Slide the panel back up to the top of the SKU ID panel, click **Active**, and then click **Save**.

At this point, you have entered all of the details received from your Oracle CPQ contact. This information must be entered correctly. The details that are entered are not seen by the customer. The information is designed to populate the config iframe window with the correct information. As a final step you activate the SKU and save the details.

Note: Since the configuration metadata must be entered manually via the Commerce administration console, keep in mind the following rules:

1. There is no support for versioning of configuration metadata so when an SKU record is imported, make sure it does not contain any configuration metadata that should replace any existing configuration metadata assigned to that SKU.
2. If the imported SKU record includes configuration metadata (columns present in the import file) but there are no values included then any existing configuration metadata will be deleted.
3. If the import SKU record does not include configuration metadata (no columns present in the import file), then any existing configuration metadata should be retained.

A Commerce administrator can view, delete, edit, or add Configuration Metadata key value pairs for any SKU where the _Externally Configurable SKU_ property is selected. A Configuration request from Commerce to Oracle CPQ always includes the configuration metadata set in Commerce for that SKU.

To work with configuration metadata you must have the following prerequisites:

- Oracle CX Commerce account
- Oracle Integration Cloud account
- Oracle CPQ account

Assign shipping groups to sub-items

You can assign different shipping groups to product configuration sub-items and more.

With a configurable item, you have the ability to assign different shipping groups to product configuration sub-items. Different shipping groups can be assigned to different levels in a multi-level configurable item.

Previously, you could only assign a shipping group at the root level. Now, with a configurable item you have the ability to assign different shipping groups to sub-items of the root item. Different shipping groups can now be assigned to the following levels in a multi-level configurable item:

- An item contained as the root item
- An item that may be contained at one or more branch items of the root item
- An item that may be contained at one or more of leaf items of a specific branch

Formerly, you assigned a shipping group at the root item level. The assumption was that the integration layer managed updates to the "shipping group" relationship object. The result of this was something where "if all sub-items are shipped then set the `shippingGroupItem` status on the root item to the status SHIPPED."

The original method did not work if you sold configured items that are a combination of goods and services and the services that needed to be assigned to separate shipping groups. It is important that customers selling nPlay bundles be able to assign each "play" to its own shipping group. As an example, you should reasonably expect to be able to assign separate shipping groups to, say, a handset (shipped to your office via priority mail), a router (shipped to your home via standard mail) and set top box (shipped to your vacation House via standard mail).

The ability to assign different shipping groups at different levels is also important for the Cancel In-flight order feature which lets you cancel In-Flight orders. An order which has been submitted but not fulfilled is considered to be In-Flight. When an In-Flight order is canceled, the process results in the creation of a new Cancel Order and may

also result in the creation of a Return Request for items that may have already shipped to the shopper and must be returned, or at least refunded. In order to determine which items have to be returned, the system must be able to determine the shipping status of each item in the configuration.

In summary, all items in a multi-item configuration hierarchy from the root to the leaf level are assigned to a shipping group. You must also be aware that the assignment of shipping groups is also dependent on other key Commerce product features that directly impact the assignment of shipping groups.

For customers that are migrating to a release of Commerce that has this shipping group feature, the assignment of the shipping group logic is not adapted automatically. They will have to modify their `ClientConfiguration` settings. For first time users of the new release, the logic is adapted automatically.

Finally, logic changes are adapted only when the customer is using the Commerce user interface for all front-end behavior. If the customer is making endpoint calls directly, then they can call the `orders` endpoint with the required payloads without worrying about modifying `ClientConfiguration`.

Understand the details of assigning separate shipping groups to sub-items

When assigning shipping groups to sub-items, keep in mind the following:

- A `shippable` product property is provided and should be set to indicate that a product, whether a hard good or service, can be physically shipped to a purchase. When the `shippable` product property for a product is set to `FALSE`, it can be assigned to a Virtual Shipping Group so that Oracle CX Commerce does not attempt to calculate shipping charges for this product..
- If a product is a service then the physical address where the service is to be provided is provided by the fulfillment system based on the service account assigned to the item. This information is also provided by the client. Address information is mandatory for virtual shipping groups as it is required for tax calculations.
- If the product is a non-shippable good, (for example, a movie download, extended warranty etc.), address information is again mandatory as it is required for tax calculations.

Note: Address information is something used extensively in Commerce transactions. For all procedures and SSEs that require address information for endpoint inputs, in addition to using Commerce's default address formats, you can also use the REST API to create multi-country custom address formats. This lets you create country-specific address formats to ensure that your address formats align with the requirements of any external service that you might use. This means that addresses appearing in profiles, accounts, registration requests, order addresses and more can be customized. For more complete information on creating custom addresses and understanding how to use custom address formatting, refer to the following:

- Customize Address Formats using the API in *Extending Oracle CX Commerce*
- Work with address types in *Extending Oracle CX Commerce*
- Account Details in *Using Oracle CX Commerce*
- Work with account addresses in *Using Oracle CX Commerce*
- Work with account registration requests in *Using Oracle CX Commerce*

An `assetable` product property is provided which identifies those products that are sold as a service or subscription (for example a mobile phone tariff, magazine subscription etc). Assetable products must be assigned to the following when purchased:

- Customer Account
- Service Account
- Billing Account
- Billing Profile

These type of products are then assigned to a Virtual Shipping Group. Even if the product is a good (for example, a physical product), it must be assigned to a virtual shipping group This is because in these circumstances the Commerce purchasing process is not responsible for calculating shipping charges and the physical address where the item must be shipped but will be based on the service account assigned to the item.

The `onlineOnly` property is provided to identify products that can be purchased online but cannot be picked up in store. This means that an item can only be assigned an `inStorePickupShippingGroup` value if the `onlineOnly` property value for that product is `FALSE`.

In summary, configured item shipping groups are assigned at all of the following levels of the configured item:

- Root item of type `configurableCommerceItem`
- Branch items of type `configurableSubSkuCommerceItem`
- Leaf items of type `subSkuCommerceItem`

The assignment of shipping groups to configured items is then dependent on whether individual products are one of (or a combination of) the following types:

- `Shippable`
- `Assetable`
- Available for purchase online only
- Being sold as a package (`soldAsPackage` SKU property). The `soldAsPackage` property is available only where the configurable value for the SKU is `TRUE`. When `soldAsPackage = TRUE`, this means that the configurable item is purchased, shipped, returned, and exchanged as a single item.

Understand the store features related to assigning shipping groups to sub-items

The following store features are provided to support the assigning of shipping groups to sub-items:

- The ability to assign all items in a configuration hierarchy to an appropriate shipping group type.
- The ability to change the assignment of shipping groups at all levels in a configuration hierarchy.
- The ability to update the tax calculation process to support shipping group assignment at all levels of a configuration hierarchy.
- The ability to update the shipping charge calculation process to support shipping group assignment at all levels of a configuration hierarchy.

- The ability to update to the proportional application of promotion discounts to all items in a configuration hierarchy

See the rest of the topics in this section for more information.

Understand tax calculation and shipping charges when assigning shipping groups to sub-items

When assigning taxes and shipping charges for shipping groups assigned to sub-items, you assign different calculating processes from normal customer calculation processes.

Understand the tax calculation process when assigning shipping groups to sub-items

The processes for calculating taxes and shipping charges for shipping groups assigned to sub-items differ slightly than the normal customer calculation process. In summary, the method used is that if sub-items are shipped separately, then the root item and the child items are sent as different items to the taxation system which contains the cost of that item alone and no additional item in the package.

This means that there are two ways that tax calculation occurs with a shipping group. The first way is that the price of the fully configured package is sent to the taxation system as all the items in the product configuration have to be delivered to a single place.

In determining the correct amount of tax payable on a product, the four key parameters passed to the tax calculator are the following:

- `amount` - the amount paid by the shopper
- `quantity` - the quantity of the item being purchases
- `shipping charge` - the shipping charge that has been calculated for the item
- `taxCode` - the tax code assigned to the product

For configured items, tax is calculated for each line item in the configuration hierarchy but the amount passed to the tax calculator is always be the external price returned from the configurator for that item, in the context of the overall configuration.

For any configured item, the price of a sub-item may be included in the price of the root item, so that the amount passed to the tax calculator would be zero.

The net result of this is that, although tax will be calculated for each item in the configuration hierarchy, all of the appropriate data will be passed to the tax calculator. It is possible, however, that the amount of tax paid by the shopper may be skewed in circumstances where the configured item contains products with different tax codes.

Understand the shipping charge calculation process when assigning shipping groups to sub-items

To determine the correct shipping charges payable on a product, the key parameters passed to the shipping calculator are the following:

- `shippingMethod` - the shipping method selected for the item.
- `quantity`- the quantity of the item being shipped.

- `amount` - the amount paid by the shopper for the item.

For configured items, shipping charges are calculated for each line item in the configuration hierarchy. The amount passed to the shipping calculator, however, will always be the external price returned from the configurator for that item, in the context of the overall configuration.

For any configured item, the price of a sub-item may be included in the price of the root item, so that the amount passed to the shipping calculator would be zero.

The net result of this is the following:

- Shipping charges are calculated for each item in the configuration hierarchy and all of the appropriate data is passed to the shipping calculator. It is possible, however, that the amount of shipping charges paid by the shopper may be skewed in circumstances where the configured item price does not accurately reflect the actual proportional amount paid for an item in the configuration hierarchy.
- Shipping surcharges are included for any item in the hierarchy where such surcharges have been assigned to that product in Commerce.
  Note: The charges are only included for the root item if the whole configuration is sold as a package
- A merchant can always choose to apply a shipping surcharge for any item where there is a risk that the shopper will be undercharged for shipping when a particular product is purchased as part of a configured item.
- Shipping surcharges are not considered if any item presents itself in a virtual shipping group as those items are non-shippable and are not required to have shipping surcharges.

Understand shipping charge and tax calculation when assigning costs to items sold as a package

When assigning costs to items sold as a package, you assign processes for calculating shipping charges and taxes that differ slightly from normal customer calculation processes.

Understand the shipping charge calculation process when assigning costs to items sold as a package

The processes for calculating shipping charges and taxes when assigning costs to items sold as a package differ slightly from normal customer calculation processes. To determine the correct amount of shipping charges payable on an item configured as a package, the following key parameters are passed to the shipping calculator:

- `shippingMethod` - the shipping method selected for the item
- `quantity` - the quantity of the item being shipped
- `amount` - the amount paid by the shopper for the item

For items sold as a single item (root item) configured as a package, the following occurs:

- The `amount` passed to the shipping calculator is always the total price for the configured item.

- The `shippingMethod` passed to the shipping calculator will always be the shipping method assigned to the root item.
- The `quantity` passed to the shipping calculator is always be the quantity of the root item.

Shipping charges will be calculated accurately, given that you have decided that the configured item must be shipped as a unit. Any shipping surcharges assigned to a sub-item in the configuration hierarchy will not be included in the total shipping charges.

Understand the tax calculation process when assigning costs to items sold as a package

In summary, the way that tax calculation occurs with a shipping group sold as a package is that the price of the fully configured package is sent to the taxation system as all the items in the product configuration have to be delivered to a single place.

To determine the correct amount of taxes payable on an item configured as a package, the following key parameters are passed to the tax calculator:

- `amount` - the amount paid by the shopper
- `quantity` - the quantity of the item being purchased
- `shipping charge` - the shipping charge that has been calculated for the package item
- `taxCode` - the tax code assigned to the product.

For configured items sold as a package (i.e., where the `soldAsPackage` value for the root item = `TRUE`), taxes are calculated based on the root item only. For configured items sold as a package, the following occurs:

- The `amount` passed to the tax calculator is always the total price for the configured item.
- The `taxCode` passed to the tax calculator is always the tax code for the root item. This means that although taxes are calculated for the configured item, the amount is based only on the tax code of the root item.

Understand how promotion discounts are applied to multi-level items

Promotional discounts can be applied proportionally to multi-level items.

For a multi-level configured item, promotion discounts must be applied proportionally across the root and all of the sub-items in the hierarchy.

In Commerce, order level discounts are applied proportionally across all items in the order (unless an item is specifically excluded from benefiting from such a discount). For a configured item, a proportional discount must be applied to all items in the configuration hierarchy. For example if an order level promotion applies a 10% discount then that 10% discount must be applied to any configured item in the order.

For a multi-level configured item, however, the promotion discount must be applied proportionally across the root and all of the sub-items in the hierarchy. This applies only to configured items that are not sold as a package (i.e. where the `soldAsPackage` value on the root item = `FALSE`).

Add payment details to customer billing profile

You must add payment details to customer billing profiles so that this information is passed downstream to fulfillment and provisioning systems.

In Telco transactions there is critical contact information that must be passed downstream to fulfillment and provisioning systems. Based on the Customer Account Model, this information is the following:

- Customer Account
- Service Account
- Billing Account
- Billing Profile

This topic covers the processes involved in the updating of payment details in a Billing Profile.

Understand how billing profiles are handled

A Contact (that is a user, shopper, or customer) may have the following information in Oracle CX Commerce transactions:

- A Customer Account (Account of type “Customer”).
- A link to other Customer Accounts. This would occur where the merchant supports account models such as “Family Account,” “Household,” or “Family and Friends.”
- A link to one or more Service Accounts (Accounts of type “Service”).
- A link to one or more Billing Accounts (Accounts of type “Billing”).
- A link to one or more Billing Profiles.

In this type of transaction model, Commerce has the important ability to create or update a billing profile with payment details. This is important because the payment information for the billing profile needs to be captured and passed on to the primary CRM (Customer Relationship Management) system in a PCI compliant manner. For the Oracle Telco CX Solution, the primary CRM system is Oracle Engagement Cloud (OEC).

Note: In an integration like this, transaction payment details that are stored in the CRM system are used for recurring payments. This info is pulled by the billing system from CRM. To compare this with Commerce, Commerce handles one time/upfront payments by interacting with payment gateways.

At this point in time, Commerce is PCI compliant whereas Oracle Integration Cloud and Oracle Engagement Cloud are not. Commerce supports the storing of credit cards against a shopper profile. The card details are captured, however, in the store as part of the checkout flow and subsequently a tokenized version of the card is obtained from an integrated payment system as part of the payment authorization process. This token, along with a masked version of card number and the expiration date are then stored against the shopper’s profile so that the shopper can easily use the same card for future purchases.

A feature is now supported that offers a generic horizontal benefit to Commerce and contributes to the Telco specific vertical requirement. This feature uses a store API endpoint that allows a shopper to store credit cards as part of their profile without

actually purchasing an order. This endpoint can then be used to enable Commerce to pass credit card information to Oracle Engagement Cloud as part of the shopper billing profile when creating or for updating accounts in OEC.

Understand the Update Profile endpoint and Generic Payment webhook

As mentioned, Commerce provides an Admin and Agent Update Profile store API endpoint that lets you add and store customer credit cards as part of a shopper Billing Profile without actually purchasing an order.

The name of the endpoint is `addCreditCard`. The Admin URI for the endpoint is `POST /ccstoreui/v1/current/creditCards/`. The Agent URI for the endpoint is `POST /ccagentui/v1/profiles/{id}/creditCards`.

The endpoint can be used to invoke Add Card requests multiple times to let you add more than one card to a shopper Profile. Each new card is then stored against the profile. The inputs of this endpoint are:

- `cardType`
- `nameOnCard`
- `cardNumber`
- `expiryMonth`
- `expiryYear`

Both versions of this endpoint trigger the Generic Payment webhook for a Tokenize operation on the payment system. The payment system is expected to return a tokenized value of the card which is then saved against the billing profile. The endpoint then returns back a stored card ID.

The Admin Get Profile endpoint can then be used to get the token value of the card using the stored card ID. See Configure Payment Processing and Create a Generic Payment Gateway Integration for more complete information on this subject.

Add and update a Billing Profile to include a card token

There are two processes/flows that Commerce uses to capture the billing profile information that can be passed on to the primary CRM system. These are the following:

- Commerce creates an account(s) for a contact which includes creating one or more billing profiles.
- Commerce updates an existing billing profile.

To assist in these processes, Commerce provides a Customer Account Model Server Side Extension (SSE). The sections that follow provide the details on the various processes and flows that this SSE supports

Understand the OCC to OEC Account Create flow

The basic information on this flow is the following:

- SSE Name: Customer Account Model
- Endpoint Name: Create Accounts
- Flow Name: OCC to OEC Account Create flow

In this process, the SSE first identifies the payment type. If the payment type is Card, a check is done to see if the token for the card has been passed in (i.e., an existing card stored on the shopper's Commerce profile). If the token has been passed in, then a check is done to see that the basic card information has also been passed in the form of `maskedCardNumber` and `expiryMonth`. If the token has not been passed in (i.e., a new card is being introduced), then a check is made to look for the following "full card" information being passed in:

- Card Type
- Name on Card
- Card Number
- CVN (card verification number)
- Expiry Month
- Expiry Year

There is also a step in the SSE execution whereby an API call can be made to the storefront Profiles/Update Profile endpoint to retrieve a tokenized version of the card. The billing profile information passed to the OCC to OEC Account Create flow includes:

- Payment Method=Card
- Masked Card Number
- Card Expiry Date
- Tokenized representation of the card

The next section provides details on an update process/flow that the SSE handles.

Understand the OCC to OEC Account Update flow

The basic information on this flow is the following:

- SSE: Customer Account Model
- Endpoint Name: Update Accounts
- Flow Name: OCC to OEC Account Update flow

In this process, the SSE first identifies the payment type. If the payment type is Card, a check is made to see if the token for the card has been passed in (i.e., there is an existing card stored on the shopper's Commerce profile). If the token has been passed in, then a check is made that the basic card information has also been passed in the form of `maskedCardNumber` and `expiryMonth`.

If the token has not been passed in (i.e., a new card has been introduced), then a check is made to look for the following "full card" information:

- Card Type
- Name on Card
- Card Number
- CVN (card verification number)
- Expiry Month
- Expiry Year

The billing profile information passed to the OCC to OEC Account Update flow includes:

- Payment Method=Card
- Masked Card Number
- Card Expiry Date
- Tokenized representation of the card

Note: Keep in mind the following additional details regarding the OCC OEC Comms: Account Update flow:

- A check is made to verify that the payment type is credit card/debit card. If it is credit card/debit card, then a check is made to verify whether `creditCardNumber` is masked or non-masked.
- If `creditCardNumber` is masked, an additional check is made to verify that the masked `creditCardNumber` and `creditCardId` values provided are valid. If both are valid, then only `creditCardNickname` is allowed to update. All other fields/properties are not allowed to update.
- If `creditCardNumber` is unmasked, then the card is considered a new card. Tokenization occurs with the provided card details is similar to the Create Account flow. The process ends with the new card and token details stored in CDM.

Understand fields and properties supported by the billing profile

For the credit card or debit card payment type, the following fields/properties are supported:

- paymentMethod: “debitcard”/”creditcard”
- creditCardNumber (mandatory field)
- creditCardExpiryMonth (mandatory field)
- creditCardExpiryYear (mandatory field)
- creditCardContactName (mandatory field)
- creditCardType (mandatory field)
- creditCardSecurityCode
- creditCardNickname
- creditCardIin

For the bank transfer payment type, the following fields/properties are supported:

- paymentMethod: “bankTransfer”
- bankAccountNumber (This is the only mandatory field for bankTransfer payment type.)
- bankRoutingNumber
- bankAccountType
- bankAccountName
- bankSortCode
- bankName
- bankAddress

- bankIban
- bankSwiftCode

Understand the process changes as seen in the store interface

With a standard checkout flow where the shopper does not yet have a customer account model and is purchasing a service, the store interface captures the payment card details. This includes the following:

- The shopper is prompted to identify the payment type. If the payment type is “Credit/DebitCard,” the shopper is prompted to select either an existing card or enter new card details.
- If it is a new card the user interface captures the following required card information:
  - Card Type
  - Name on Card
  - Card Number
  - CVN (card verification number)
  - Expiry Month
  - Expiry Year

For a billing profile update checkout flow where the shopper, who does not have a customer account model and is purchasing a service, the store interface captures the payment card details. These details include the following:

- The shopper is prompted to identify payment type.
- If the payment type is “Credit/DebitCard,” the user interface captures the following the required card information:
  - Card Type
  - Name on Card
  - Card Number
  - CVN (card verification number)
  - Expiry Month
  - Expiry Year

Understand the Customer Account Model

For customers using the Customer Account Model SSE, there are a number of different account types that can be associated with a shopper within the Oracle CX Commerce/Oracle CPQ integration.

If you are using the Customer Account Model SSE, there are a number of different account types that can be associated with a shopper within Oracle CX Commerce. To configure the Customer Account model, use the provided SSE. To do this, click the **Design** icon in the Administration user interface. Then click **Developer** and **Server-Side Extensions**. Select the CustomerAccountModel-store SSE and/or the CustomerAccountModel-agent SSE.

Both SSEs enable integration with an external CRM system to retrieve and update the following:

- Contacts
- Accounts (Customer Billing and Service accounts)
- Account Roles (Admin, Buyer and User)
- Billing Profiles

Finally, the SSEs serve as the API for the pre-built integration with Oracle Engagement Cloud.

There are three account types available within Commerce relating to billable services, Customer account, Service account, and Billing account.

The details for these three accounts are captured when an order is placed and their relationship with the service is maintained after an order has been fulfilled.

In many instances these three accounts may all refer to the same person or organization, but there may also be instances when they differ, and it is important to understand the relationship between the different types of account.

In addition to the three account types, there is a Billing Profile, which includes information such as billing preferences.

All of the information required for the Customer, Service, and Billing accounts, and for the Billing Profile is captured during the order process in Commerce.

Customer Account

This type of account represents the person or organization that owns the service. It includes basic customer information, such as name and address and can receive both services and bills.

Customer accounts are the highest level in the account hierarchy and can perform all customer, service, or billing functions.

Service Account

This type of account represents the person or organization that receives the service.

The address associated with the Service account defines the physical location where the service must be delivered. This address is used to verify service and ordering eligibility.

Service accounts are required when the location and/or party receiving the service differ from the Customer account. If a Service account is required, it is always a child of a Customer account. There can be multiple Service accounts associated with a single Customer account.

A Service account cannot be used to perform any of the functions of a Customer or Billing account.

Billing Account

This type of account represents the person or organization that pays for the service.

Billing accounts are required when the location and/or party paying for a service differ from the Customer account. If a Billing account is required, it is always a child of a

Customer account. There can be multiple Billing accounts associated with a single Customer account.

A Billing account cannot be used to perform any of the functions of a Customer or Service account.

Billing Profile

A billing profile may be associated with either a Customer account or a Billing account. It captures information such as billing preferences, method of payment, and contact details. There may be more than one billing profile associated with a Customer or Billing account, and the shopper must choose which billing profile to use when placing an order for a service.

Note: Address information is something used extensively in Commerce transactions. For all procedures and SSEs that require address information for endpoint inputs, in addition to using Commerce's default address formats, you can also use the REST API to create multi-country custom address formats. This lets you create country-specific address formats to ensure that your address formats align with the requirements of any external service that you might use. This means that addresses appearing in profiles, accounts, registration requests, order addresses and more can be customized. For more complete information on creating custom addresses and understanding how to use custom address formatting, refer to the following:

- Customize Address Formats using the API in *Extending Oracle CX Commerce*
- Work with address types in *Extending Oracle CX Commerce*
- Account Details in *Using Oracle CX Commerce*
- Work with account addresses in *Using Oracle CX Commerce*
- Work with account registration requests in *Using Oracle CX Commerce*

Use Recurring Charge Items

This integration provides you with the ability to configure items with a recurring charge that can be passed on in purchase.

This integration enables you to provide items that come with a recurring charge available for shoppers to purchase. Examples of items that include a recurring charge include a service such as a data/call minutes/ text message bundle for a cellphone, or a subscription charge for a cable television package.

Items that include a recurring charge may have just a recurring charge or may have a recurring charge and a price. If an item has a price and a recurring charge, it is assumed that the item is not a service or subscription item. In this case the price represents an upfront payment and the recurring charge is the means by which the outstanding balance is paid.

Identification of items that include a recurring charge must be carried out through your Oracle CPQ Admin account. Please refer to the Synchronize Oracle CPQ Cloud Parts with Commerce SKUs section of the Implementation Guide contained in the Integrating Oracle CX Commerce with Oracle CPQ article on My Oracle Support.

If a shopper adds a recurring charge item to their cart, Commerce works with Oracle CPQ to display full information about the recurring charges associated with the order. This includes how much the recurring charge is for, the frequency of the recurring charge, and the duration for which the recurring charge will be made.

Note: The default value for frequency is monthly and the default value for duration is open-ended. If either of these is not the right value for the item they must be corrected in the Oracle CPQ Part for the item.

Items with a recurring charge are not included in order sub-total passed to the shipping calculator. If a cart contains only recurring charge items the order sub-total passed to the shipping calculator is zero, which means that no shipping charge is applied to the order.

Configure payment for recurring charge items

Commerce includes several built-in integrations with payment gateways that let your store accept credit cards, debit cards, gift cards, and PayPal payments. However, these integrations do not currently support recurring charges. If you wish to sell items with recurring charges you must use one of the methods set out below to configure Commerce payment processing to support recurring charges.

Configure credit card payments

Follow these instructions to configure your credit card payment processing to handle recurring charges:

1. Create a custom credit card payment extension.
   For detailed instructions about performing this step, refer to Create a credit card extension.
2. Install the custom credit card payment extension.
   For detailed instructions about performing this step, refer to Install the extension and configure the gateway.
3. Enable the payment gateway.
   For detailed instructions about performing this task, refer to Create a Credit Card Payment Gateway Integration and Create a Generic Payment Gateway Integration, .
4. Add custom properties to the Credit Card Payment webhook.
   For detailed instructions about performing this task, refer to Install the extension and configure the gateway .

   Note: This webhook is not site-specific. If you are running multiple sites on your Commerce instance, the configuration you supply applies to all sites that use this webhook.

Configure non-credit card payments

Follow these instructions to configure your generic gateway payment processing to handle recurring charges:

1. Create a custom generic payment extension.
   For detailed instructions on performing this task refer to the Supported payment methods and transaction types section of Create a Generic Payment Gateway Integration.
2. Install the generic payment extension.
   For detailed instructions about performing this step, refer to the Install the extension section of Create a Generic Payment Gateway Integration.
3. Enable the payment gateway.
   For detailed instructions about performing this task, refer to Create a Credit Card Payment Gateway Integration and Create a Generic Payment Gateway Integration.

4. Customize the payment details widget to capture payment information other than card details.
5. Add custom properties to the Generic Payment webhook.
   For detailed instructions about performing this task, refer to Send custom properties to a payment gateway.

   Note: This webhook is not site-specific. If you are running multiple sites on your Commerce instance, the configuration you supply applies to all sites that use this webhook.

Use Asset Based Ordering

The Commerce/Oracle CPQ integration features asset based ordering (ABO).

Understand asset definition and related properties

This integration supports an asset based ordering (ABO) model. Asset based ordering lets you sell tangible assets or subscription services delivered over a period of time; for example, mobile phone call and data plans, television and broadband packages. When these orders are subsequently fulfilled, the fulfillment system notifies Oracle CPQ via an asset API, and Oracle CPQ then creates an asset in the Oracle CPQ asset repository. To better understand asset based ordering and its related services, it is important that you first understand asset definition and the related properties.

In the Commerce/Oracle CPQ integration, Commerce acts as the first point of contact for registered and account-based shoppers. Commerce lets a shopper review and select their purchases as needed.

In Telco-related purchases, Oracle CPQ acts as the primary Asset system. Commerce makes a call to Oracle CPQ to retrieve the assets for a particular profile or account. Oracle CPQ then manages the retrieval of assets from multiple systems if necessary.

One of the underlying features of any Telco solution is the ability for a self-service channel (in this case, Commerce) to retrieve and display the complete set of assets owned by the shopper and then to allow the shopper to trigger operations on those assets. In order for this to happen, Oracle Commercesupports the following asset-related information properties at the order item level:

- Asset Key - the `assetKey` property (formerly `assetID`) is a unique identifier that is assigned to potential assets when adding items to a cart. This value is used throughout the asset life cycle by fulfillment, asset management, and order capture systems. In this case, the term "potential" is used meaning that not every item added to a cart gets completely fulfilled, a provisioning system may fail, etc. For configured items, the assetKey value is assigned as part of the asset configuration process in Oracle CPQ.
- Parent Asset Key - Some configured items in an order may be many levels deep in a BOM structure. In order to ensure that the BOM hierarchy is consistent throughout the asset life cycle, each item in the BOM hierarchy must be able to identify its direct parent. The `parentAssetKey` property makes this possible. For root items in a BOM hierarchy, the `parentAssetKey` value is `NULL`.
- Root Asset Key - Again, some configured items in an order may be many levels deep in a BOM structure. In order to ensure that the BOM hierarchy is consistent throughout the asset life cycle, each item in the BOM hierarchy must be able to identify its root asset. The `rootAssetKey` property makes this possible. For root items in a BOM hierarchy, the `assetKey` and the `rootAssetKey` value is the same.

Understand the mapping of an asset key to an item

In Oracle Commerce, a configurable SKU may be flagged as "non-assetable" which means that when this item is configured and purchased it will not be assigned a customer, billing, or service account and will not become an asset for the shopper. When this item is configured, however, Oracle CPQ returns asset key values for each item in the BOM by default.

Note: The flag name is `assetable` and the default value is `False`.

Commerce only maps asset key values to commerce items that are actually "assetable." The rules used in this process are the following:

- If the SKU selected for configuration is based on a product where the property value for `assetable = TRUE`, map the asset key data.
- If the SKU selected for configuration is based on a product where the property value for `assetable = FALSE` do not map the asset key data.

Understand the Asset Root

It is also important to point out that when a shopper chooses to configure a SKU in Commerce, the root item of the BOM returned from Oracle CPQ may not always be that same SKU, that is, the root item part number may not map directly to the selected configurable SKU.

Say, for example, a mobile product bundle that is represented by the "Red Bundle" SKU in Commerce is configured several ways. At the initial step of the configuration process, the shopper may be asked to select either the Standard Package, Student Package, or Value Package. Depending on the selection made, the root item of the configuration will be different.

So, based on this example, it is possible that the SKU selected by the shopper to configure the item will be based on a product where `_assetable _= TRUE` but the root item for the resulting configuration may be based on a product where `_assetable _= FALSE`.

The rule that decides whether a configured item should be assigned `_assetKey _property` is based on whether the SKU that corresponds to the root item of the configuration is "assetable" and not on whether the item that the shopper selected to be configured in Commerce is "assetable".

Understand asset based ordering and related service operations

As already discussed, asset based ordering lets you sell assets or subscription services delivered over a period of time. When these orders are subsequently fulfilled, the fulfillment system notifies Oracle CPQ via an asset API, and Oracle CPQ then creates an asset in the Oracle CPQ asset repository.

Once created, assets can subsequently be reviewed by shoppers in the My Services management area within the shopper account. The shopper can then administer an asset by creating and placing new commerce orders to perform a number of actions on the asset. These include the following:

- Modify
- Renew
- Terminate

- Suspend
- Resume
- Upgrade

A Services-store SSE and the Services-agent SSE can be configured from the administrator's user interface. To do this, click the **Design** icon in the Administration user interface. Then click **Developer** and **Server-Side Extensions**. Select the name of the SSE. Both SSEs enable integration with 3rd party asset management systems to retrieve and execute operations and services on assets available to the shopper. They also serve as the API for the pre-built integration with Oracle CPQ asset management.

For each of these operations the operation flow is basically the following:

- The shopper views their list of assets.
- The shopper selects an asset.
- The shopper selects the desired operation:
  - For a Modify operation, the system loads the Oracle CPQ hosted iFrame, the shopper makes their modifications, and selects to add to cart. This is the Oracle CPQ hosted iFrame presented to the shopper when they configure a new purchase prior to adding it the cart, reconfigure a new purchase prior to checking out, or modify an existing asset.
  - For an Upgrade operation, all available upgrade options are displayed on the Storefront Asset list and then the specific Asset Details pages. After you have selected a specific Asset, you can select the Upgrade option to view its upgrade details. When you click on an upgrade option, an iFrame is returned and opens up in the context of the available upgrade options. You can then choose your asset upgrade(s) and add them to your cart.
  - For all other operations, the system only makes a call to Oracle CPQ to execute the operation.
- Oracle CPQ asset records are updated.
- Oracle CPQ returns the required JSON representation of the terminated/renewed/suspended/resumed/modified/upgraded asset.
- Commerce transforms the JSON returned to a commerce item and adds it the cart.
  - For the Modify and Upgrade operations, the transformation is executed in the Commerce client layer.
  - For all other operations the transformation occurs in the Services SSE which uses the Asset Action OIC flow.
- The shopper continues shopping.
- When the shopper places the order, the `cpq-config-validation-app` SSE is triggered through the External Pricing Webhook. This SSE invokes `getConfiguration` for every flow except when the asset actions are Terminate and Suspend. The response received from OIC gets transformed from the `cpq-config-validation-app SSE` as the OIC flows, `getConfigurations`, and `getConfigBom` return a flat structure of items which is converted to a hierarchical structure. Validation is then done in the `cpq-config-validation-app` SSE to verify that data is not manipulated on client-side.

- The order items representing Asset Based Ordering operations are submitted downstream and contain all of the information required to ensure that the operation is fulfilled.

The specific Services actions are described in more detail later in this section. These actions are important for maintaining an efficient self-service channel. When a shopper performs any one of these actions on an asset, the Oracle CPQ asset repository is updated accordingly.

Since Commerce serves as the first point of contact for shoppers, it allows shoppers to review and select their purchases. In the case of a Telco commerce solution, the Oracle CPQ asset repository acts as the primary Asset system in which Commerce makes a call to Oracle CPQ to retrieve the assets for a particular profile or account. Oracle CPQ manages the retrieval of assets from multiple systems.

The Commerce Telco solution gives the shopper the ability to retrieve and display the complete set of Assets owned by the shopper/account as well as carry out the mentioned administration operations that can be performed on those assets.

When a shopper opens the My Services management area within their account they are presented with a list of the assets linked to their account. From here they can select an asset and click on the Details button next to the desired asset to see the detailed view of the service.

It is at this point that the shopper can choose between the Modify, Renew, Terminate, Suspend, Resume, and Upgrade actions.

Modify

If the shopper chooses Modify, Commerce loads the current configuration for the service in question and opens a screen that allows the shopper to modify the service as required. The new monthly charge for the service is updated automatically as the shopper makes their selections. The shopper can then add the modified service to their cart.

When the shopper goes through checkout and completes their order, Commerce submits a service modification request to the fulfillment system.

As mentioned, earlier the steps in this operation are typically the following:

- The shopper views their list of assets.
- The shopper selects an asset.
- The shopper selects a Modify operation. For a Modify operation, the system loads an Oracle CPQ hosted iFrame. The shopper makes their asset modifications and selects to add it to cart.
- Oracle CPQ asset records are updated and Oracle CPQ returns the required JSON representation the terminated/renewed/suspended/resumed/modified asset.
- Commercetransforms the required JSON returned to a commerce item and adds it the cart. This transformation is executed in the Commerce client layer.
- The shopper continues shopping and then checks out.

The order items representing ABO operations are submitted downstream and contain all of the information required to ensure that the operation is fulfilled.

Renew

If the shopper chooses Renew, Commerce determines the configuration ID that represents a renewal of the service in its current configuration and then adds a renewal instruction to the shopping cart and opens the Shopping Cart Details page.

When the shopper goes through checkout and completes their order, Commerce submits a service renewal request to the fulfillment system. This is handled and invoked via the Services SSE endpoint `/services/{id}/renewService` and the SSE invokes the OIC flow.

Terminate a service

If the shopper chooses Terminate, a configuration ID is sent back by Oracle CPQ that represents the termination of the service in question. A termination instruction is added to the shopping cart and the Shopping Cart Details page is then opened

When the shopper goes through checkout and completes their order, Commerce then submits the service termination request to the fulfillment system. This is handled and invoked via the Services SSE endpoint `/services/{id}/terminateService` which invokes the OIC flow.

Suspend a service

If the shopper chooses Suspend, it allows them to suspend a service. Commerce provides an endpoint that is used to suspend a service. When a shopper selects to suspend a service, they choose the Suspend action and then enter a valid suspend date.

By clicking on the Suspend button, Commerce determines the configuration ID that represents the suspension of the service in question, adds a suspension instruction to the shopping cart, and opens the Shopping Cart Details page. When the shopper goes through checkout and completes their order, Commerce submits a service suspension request to the fulfillment system. Also, when the Suspend action is chosen from the store user interface, the transaction date is set to current date (i.e., the date that the shopper suspended the service. This suspension may be indefinite or for set for a specific period of time by entering a date. A specific shopper use case example might be letting a shopper suspend a data plan for 30 days.

The Services SSEs support the Suspend operation which returns either a Configured Item or an Error. Services is part of the Oracle Integrated Cloud flow.

The Services API has an endpoint called Suspend Service. The endpoint can be triggered when a shopper selects to suspend a service, enters a valid suspend date and time, and selects to proceed. Inputs include the following:

- Asset Key
- Action - Suspend
- Transaction Date - The valid suspend date and time information that the shopper entered. The Suspend date is not equal to or later than the asset end date.

The API returns either a Configured Item or an Error.

Resume a Service

If the shopper chooses Resume, it allows them to resume a service that was previously suspended. Commerce provides an endpoint that is used to resume a

service. When a shopper selects to resume a service, they choose the Resume action and then enter a valid resume date and time to resume the service.

By clicking on the Resume button, Commerce determines the configuration ID that represents the service in question that is to be resumed, adds a resume instruction to the shopping cart, and opens the Shopping Cart Details page. When the shopper goes through checkout and completes their order, Commerce submits a resume service request to the fulfillment system. Also, when the Resume action is chosen from the store user interface, the transaction date is set to current date (i.e., the date that the shopper resumed the service).

The Services SSEs also support this Resume operation which returns either a Configured Item or an Error. Services is part of the Oracle Integrated Cloud flow.

The Services API has an endpoint called Resume Service. The endpoint can be triggered when a shopper selects to resume a service, enters a valid resume date and time and selects to proceed. Inputs include the following:

- AssetKey
- Action: Resume
- Transaction Date: The valid Resume date and time that the shopper entered. The Resume date is not equal to or later than the asset end date.

The API returns either a Configured Item or an Error.

Note: An action code of Renew, Terminate, Suspend, and Resume is assigned to an item when that respective operation has been applied to that item.

Upgrade an Asset

With Asset Based Ordering, you have the ability to upgrade an existing asset. If a shopper chooses the Upgrade operation, they can upgrade an asset to one of the upgrades available for the product. Any Root asset may have one or more upgrade options available at any time. Commerce SSE endpoints `getServices` and `getServices/{id}` return the upgrade options for each of the asset if the query param `"expand=occ_upgradeOptions"` is passed. Once the shopper selects the Upgrade action and clicks the Upgrade button, this action invokes the `upgradeService` SSE endpoint which gets the upgrade name as input and returns the query string that is to be used as punchin URL to launch the Oracle CPQ iFrame. From the user interface point of view, a shopper selects to upgrade an asset, choose the Upgrade action in the Asset Details view and then select the asset upgrade that they desire.
Oracle CPQ maintains a custom upgrade options table for Commerce to query in order to know which upgrades are available for a given asset. The key parameter controlling the operation is the SKU of one or more items that are part of a current asset bundle. The response received after initiating this operation includes all of the eligible SKUs that an asset can be upgraded to.

Commerce has an Upgrade endpoint to fetch all available upgrade options. The input for this endpoint is `currentModel` and `currentOffer`. The following presents the details on the information needed to retrieve the upgrade options table from Oracle CPQ:

- Oracle CPQ Table Name: `INT_UPGRADE_OPTIONS`
- Input (via URL parameter): `occ_Upgrade_options` query parameter which is a list of `currentSku` plus `currentModel` for the assets. Type: String. This query parameter is passed from Commerce to the SSE endpoint (described further in this section). After a `getAssets` call, you then pick the `currentModel` and `currentOffer` from each asset and invoke the Oracle CPQ upgrade options table.

- Output: `upgradeName, upgradeProductId(OCC)`

The following presents the details on the basic schema of the upgrade options table maintained by Oracle CPQ that contains the specified upgrade information:

Table 3-1 Oracle CPQ Upgrade Table

Column Name	Data Type and Description
currentSku	String. This value defines the current offer. This needs to be stored as an attribute of an asset record. This value is sent from Commerce while retrieving the upgrade options.
currentModel	String. The model name for which the upgrade offer is valid.
upgradeName	String. This value is passed to he Oracle CPQ iFrame while upgrading and is used by Oracle CPQ to default and render the upgrade options. This is not be used by Commerce for any purpose.
upgradeProductId	String. This is used by Commerce to identify the product corresponding to upgrade option. The product display name, description, images, etc. can be used to show upgrade details to the shopper.

Note: A combination of currentSku and currentModel is used as the parameter to find the matching upgrade options

The Oracle CPQ upgrade table is queried by Commerce to help identify the upgrades that are available for a given asset. These upgrade options are then presented to the shopper. An example of what the upgrade information would contain includes the following:

Table 3-2 Example of upgrade options returned to Commerce

currentSku	currentModel	upgradeName	upgradeProductID
4ForUDeal	nPlay	4ForUDeal	prod102

It is recommended that the currentSku column is indexed. The following presents additional details on each returned upgrade option:

- currentOffer - Maps to a configurable attribute on the root config model in Oracle CPQ. This needs to be stored as an attribute mapping onto the root asset as well. This value is sent from Commerce while retrieving the upgrade options.
- currentModel - Maps to the variable name of the root config model in Oracle CPQ which the upgrade offer applies to.
- upgradeName - Maps to the `_config_upgrade_name` that is passed from Commerceto Oracle CPQ, which drives recommendation rules on the upgrade. This is not used by Commerce for any other purpose.
- upgradeProductId - Maps to the Product Id of the upgrade offer in Commerce. This is used to show upgrade details (product display name, description, images, etc) to the shopper.

As mentioned, Commerce provides an Upgrade endpoint that is used in the operation to upgrade the asset. This endpoint is part of the Services SSE which works to complete multiple service operations (already mentioned in the above sections) via the Services API. For this operation, the Services API has an endpoint called Upgrade. The following information provides more detail on what is required by the API to upgrade an asset using this endpoint:

- SSE name: Services
- Endpoint name: `Upgrade`
- Endpoint trigger: The endpoint is triggered when the user clicks **Upgrade** against an upgrade option
- Inputs:
  - Logged in User Token
  - AssetID
  - upgradeName (returned from Oracle CPQ)
- Returns: Upgrade URL Query String. This is the string of data that is appended to the base Modify URL to ensure that the upgrade iFrame is correctly pre-populated based on the product that the shopper is upgrading from and the product that they are upgrading to.

The activity that occurs at the store user interface level during the Upgrade operation is the following:

- Select the Asset List view. This lets you view information about all of your assets/services. This view will also show the upgrade options (if available) for the asset. You cannot trigger an Upgrade operation from this view as the actual upgrade URL is not yet determined until the asset details are retrieved.
- Select the asset you wish to view and click the Details button so that you can view the asset details and as well as possible upgrade details.
- When you click the Details button of any asset, the asset details page is displayed which shows all of the details associated with that asset along with available asset action options.
- The asset details page also has a section showing the upgrade options available for that asset. When you display the asset details page, the product details of that SKU/Product are displayed. The Upgrade button is displayed next to any upgrade available for that asset.
- Click the Upgrade button in the Asset view of the asset that you want upgraded. Your upgrade option details are then displayed in the Asset Details view. You can also get the same results by clicking the link for the available upgrade from the Service list.
- Click Upgrade. When the Upgrade operation is initiated, the following occurs:
  - If there are upgrades available for the asset, the SSE endpoint returns an Upgrade URL Query String and creates the upgrade punch-in URL to load the iFrame containing the information about the available upgrades.
  - When you select to upgrade you are finally presented with a pre-configured modification to your asset bundle.
  - If the SSE endpoint returns an error, this means there are no upgrades available for this asset and an appropriate error message is displayed.

- Add the upgrade to your cart and submit the order to complete the upgrading process.

Finally, each of the Asset Based Ordering services operations described earlier may be carried out by a shopper or by an agent acting on the shopper's behalf.

Additional information related to using the Upgrade feature with Commerce

The following additional details should be kept in mind when using the Upgrade feature:

- In Commerce you can start a configuration upgrade from a configurable SKU (for example, "4ForU Deal") which in turn maps to a model "nPlay" in Oracle CPQ.
- Configuration metadata is set with key "offer" and value "4ForU Deal" for the above SKU and is passed to Oracle CPQ
- After the configuration upgrade is completed, the BOM returned from the Oracle CPQ for that configuration may have a different rootSKU (i.e., "nPlay") and that is what is added to cart. "4ForUDeal" may be a child of "nPlay".
- In Commerce, there is another SKU for "nPlay" that is configurable and maps to the same model "nPlay" in Oracle CPQ.
- After the order is submitted, an asset with "nPlay" is created which has an asset attribute of `Offer`. `Offer` then has a value of `4ForUDeal`.

Also, via the CommerceAdmin, you can create products with an `upgradeProductId` as the `productId` value, and mark them as '`notForIndividualSale`.' This lets you do the following:

- Have a unique name for each upgrade that can be displayed in the store
- Have a unique description to describe what the upgrade is
- Support locale specific translations
- Have the ability to upload images related to the `upgradeOption`.

Handle further upgrades to an asset that has already been upgraded

In some use cases, you may have a situation where you have an asset with `currentOffer=sku1234` that is being upgraded to **Upgrade 101**. When you then visit the **Asset Details** page again you are presented with the same upgrade option of **Upgrade101.** This can occur because the upgrade does not modify the `currentOffer` and it is still `sku1234` and its corresponding upgrade options are being fetched during the getAssets/getAsset flow.

The following details show how you can solve this type of situation:

Table 3-3 Example of how to handle further upgrades to an asset

currentOffer	currentModel	upgradeName	upgradeProductID
4ForUDeal	nPlay	4ForUDealPlus	4ForUDealPlus

- Let's say a shopper starts an upgrade configuration from the SKU "4ForUDeal" by passing the configuration metadata `offer=4ForUDeal`.

- After upgrading the configuration, the BOM sent from Oracle CPQ may have a different root SKU id such as "nPlay." "4ForUDeal" may be a child of it. It will also contain an attribute "offer" with value "4ForUDeal"
- An asset with "nPlay" as the currentModel gets created and the getAssets/getAsset flows return the asset details along with asset attribute `offer=4ForUDeal`.
- The offer attribute is sent as the currentOffer to the Oracle CPQ while retrieving the upgrade option 4ForUDealPlus.
- Once the upgrade has been performed by passing the upgrade name 4ForUDealPlus to Oracle CPQ in the queryString, the BOM returned from Oracle CPQ will have the attribute "offer" with value "4ForUDealPlus".
- After submitting the order and updating the asset, the asset attribute "offer" value now gets updated to "4ForUDealPlus".
- In subsequent getAssets and getAsset calls the asset attribute offer value will be returned as "4ForUDealPlus", so that there are no matching records for that currentOffer in upgrade options table in Oracle CPQ.

Understand the Disable Reconfiguration feature

Regarding these operations, the Oracle CX Commerce and Oracle CPQ integration also has the ability to prevent shoppers from attempting to reconfigure items in their cart that have been added by any of the following operations:

- Renew
- Terminate
- Suspend
- Resume

To assist in disabling reconfiguration on already configured items added by any of these actions, an action code of Renew, Terminate, Suspend, and Resume is assigned to an item when that respective operation has been applied to that item.

This code is assigned to make sure that shoppers are prevented from attempting to reconfigure an asset. The purpose of the code is to make sure the reconfigure session(s) fails, either at reconfiguration or order validation time.

Differentiate between new order items and ABO order items

To identify items in an order that are the result of an operation on an existing asset (Terminate, Renew, Suspend, Resume, Modify, Upgrade), Commerce has checked to see if there was an `assetId` value. If there was, Commerce assumed that the item is the result of an ABO and not a net new purchase. This approach worked on the assumption that an asset identifier would only be assigned when the asset record was created in Oracle CPQ.

Asset identifier values are now assigned at the time when a shopper adds an item to the cart. To ensure that Commerce can always reliably differentiate between new order items and ABO order items when an ABO item is added to the cart, a `lineType` property for each item in the configuration hierarchy is set to ASSET.

The rule used to differentiate between new order items and ABO order items is the following: If `assetKey` value is present and `_lineType = NULL` then the item is a new purchase and not an operation on an existing asset.

Retrieve assets for an order with an asset key

For the cancel in-flight feature, Commerceneeds a mechanism for retrieving all of the assets derived from a particular order. Commerce used to retrieve the assets for a particular order based on `assetID` (stored on the asset record in Oracle CPQ). Commerce now uses the `assetKey` value.

For any given order Commerce queries the Oracle CPQ assets API to retrieve the assets for the order based on the collection of `assetKey` values. This query is limited to the `assetKey` values for the root items in the order only

Understand restricting the quantity of assetable items

A shopper used to be able to increase the item quantity for a configured item in the cart in the same way as any other purchase. This action does not work where an asset key value has been assigned.

Asset keys are assigned to net new purchases as part of the configuration process. Oracle CPQ assigns an `assetKey` for the root and all child items in the configuration. If an item has been assigned an asset key then this asset key is used to identity a single instance of this asset throughout the fulfillment, provisioning and asset management processes. As a result, the quantity of an item cannot be greater than one.

Customize configurations in Commerce using the CPQ Configuration API

You can customize the configurations of complex products in Oracle CX Commerce by using the Oracle CPQ Configuration API to avoid being redirected to a Oracle CPQ hosted iFrame.

You can now customize the configurations of complex products in Oracle CX Commerce without being redirected to a Oracle CPQ hosted iFrame.

You can now customize the configurations of complex products in Commerce without being redirected to a Oracle CPQ hosted iFrame which may have a separate and distinct user interface look and feel that creates a disjointed user experience. This capability, known as the Direct API Configuration feature, is provided to build out support in Commerce for API driven product configurations where the user interface experience is controlled by Commerce and can be customized by Commerce partners. At a high level, this feature lets you do the following:

- Create brand specific configuration user interfaces and controls at the global level.
- Create a specific user interface experience for individual customizable products at the product level.

The goal of this feature is to provide full support of the Oracle CPQ Configuration API on Commerce Storefront frameworks. This includes providing a mechanism to dynamically create user interface elements that let shoppers select customizable products. It then presents them with the appropriate user interface elements to complete the customization process and add the each item to the cart. These user interface elements are generated dynamically in response to the selections made by the shopper at each step of the customization. The functionality of this feature is fully compliant with current Commerce Storefront frameworks.

The principal benefits of the Direct API Configuration feature are the following:

- iFrame is not required - The current functionality requires that the configuration system (Oracle CPQ) perform all of the configuration tasks. This means that the shopper's user interface experience is managed in 2 separate applications. Up to the point where the shopper selects a customizable product, their user interface experience is driven by Commerce. On the other hand, the configuration user experience is managed by Oracle CPQ and when the shopper adds the configured item to the cart the user experience reverts back to the control of Commerce. The addition of this feature means that customers will not be required to execute product configuration via an iFrame. This lets shoppers experience a consistent user interface with common look and feel across their storefront.
- Decoupling of the user interface and the configuration process - This feature ensures that the user interface framework is decoupled from the configuration process. This lets customers do the following:
  - Build configuration user interface components using the Commerce Design Page based on the Store Front 1.0 Framework.
  - Build configuration user interface components using a non-Commerce design user interface framework.
- Performance improvements - The use of the iFrame pattern also creates a performance concern. The former integration with Oracle CPQ functions well and the disjointed user experience can be managed to some extent with user interface customization. However, there is also no reliable evidence that this design pattern performs at the levels required for high volume customer-based Telco implementations, where hundreds of thousands of shoppers may be configuring complex Telco bundles at the same time. This feature attempts to address this concern.

The roles that Commerce and Oracle CPQ now take with this feature are the following:

- Oracle CPQ remains the primary configurator and controls the following:
  - What needs to be configured
  - The sequence in which components/attributes are presented
  - The configuration values that are required or accepted
- The Commerce client is responsible for how the configurator is displayed (without an iFrame).

Additional topics in the current chapter provide you with detailed use cases for this feature.

Understand the support of the Oracle CPQ Configuration APIs

This feature provides a downloadable extension to the Commerce application component that provides a collection of endpoints which lets the Storefront UI (regardless of which user interface framework you are using) do the following:

- Retrieve the end to end UI flow for a given Oracle CPQ Configuration Model
- Retrieve sufficient metadata to identify the user interface elements required for each attribute of the model. These elements include the following:
  - Input Controls (Radio Buttons, List Boxes, Toggles, Date/Time Pickers etc.)
  - Navigational Components (Breadcrumbs, Sliders, Image Carousels etc.)
  - Information Components (Progress Bars, Tool Tips etc.)
  - Containers such as accordion elements

- Retrieve data required to correctly populate each user interface element. This includes Label Names, Selectable Options, and more
- Create a product configuration
- Update a product configuration
- Update user interface flow
- Update a user interface elements
- Modify a product configuration
- Upgrade a product configuration
- Save a product configuration
- Transform a BOM (bill of materials) to a Commerce cart item
- Reconfigure a saved product configuration

This extension also handles the following management tasks:

- Maintains the state of the configuration until such time as it is saved.
- Makes sure that calls made from the user interface framework to the Commerce Extension are authorized.
- Makes sure that calls made from the Commerce Extension to Oracle CPQ Configuration REST APIs are authorized.
- Ensures that connections are made from the user interface framework to extension to Oracle CPQ REST APIs without OAIC (Oracle Integration) integration flows.
- Manages BOM (Bill of Materials) data objects returned from Oracle CPQ when the configuration is saved.

This Commerce Extension supports any user interface client, including those built on Commerce Storefront 1.0.

Understand supported integration-specific configuration APIs

The Oracle CPQ (Configure, Price and Quote) Cloud solution supports the complete quote-to-cash process from customer inquiry to order fulfillment. It guides users to optimal product options and configurations from simple to complex, automatically applying discounts and relevant up-sell and cross-sell opportunities. Oracle CPQ exposes objects and data through REST APIs. By exposing objects and data through REST APIs, Oracle CPQ promotes simpler API calls and more robust integration using HTTP standards. For the Direct API Configuration feature and current Oracle CPQ Integration support, the following configuration APIs are mostly used:

- Configuration Run-Time Data Services APIs - These endpoints expose information and perform an action for a configuration model. All Configuration Run-Time Data REST APIs follow a required product hierarchy starting with the product family then product line followed by the model. A variable name for the product entity is required. For example, `/config{prodFamVarName}.(prodLineVarName}.{modelVarName}/` is the standard Configuration Run-Time Data product path for an endpoint URL.
- Configuration Administration REST APIs - These APIs provide product configuration endpoints that expose configuration definition information for Configuration Product Families, Product Lines, Models, attributes, array sets, menu items, and translations. The information for these items is organized

in a hierarchical structure. The Configuration Administration REST API query parameters are supported to include and exclude child resources in a given resource. The response for each level in the hierarchy can include the details of the sub resources based on the query parameter passed in the request.

Customer Configuration flows dictate how users go through the site pages and the options available as they create a Transaction. Configuration flow rules consist of a condition and flow attributes. Actions display based on which node in the flow that the user has available based on defined criteria. Beginning in Oracle CPQ Release 18D,Oracle CPQ transformed the current configuration definition as REST endpoints to support UI interfaces. These services are available v7 and higher RESTful services.

Refer to the Oracle CPQ REST API documentation for more complete information.

Understand how the Direct API Configuration feature enhances Asset Operations

As mentioned, this feature provides Commerce with support of the Oracle CPQ Configuration API Layer while using the Commerce and Oracle CPQ integration. This means providing functionality that lets customers, using any user interface framework, configure and/or reconfigure customizable products by invoking the following from Oracle CPQ:

- Configuration Run-Time Data Services APIs
- Configuration Administration REST APIs

Building on this foundation, the feature further supports some asset-based operations whereby the configuration model retrieved from Oracle CPQ represents an existing asset. This lets the shopper execute the following configuration-related Asset Operations via direct API calls to the Oracle CPQ Configuration API:

- Modify
- Upgrade

Available Storefront and Agent endpoints for this feature let you modify and upgrade assets via direct API calls to Oracle CPQ thus removing the need to include an iFrame in this part of the shopping experience as well. This feature is limited to API only and customers will need to build their own UI elements to invoke these new endpoints.

By creating your own Modify and Upgrade user interface elements, you can deliver a seamless and consistent user experience even when modifying or upgrading complex products or services. The shopper user interface experience while modifying or upgrading a service can then be consistent with the rest of the site navigation experience as configuration user interface controls can be created in compliance with the Site Theme and CSS being used.

To fully implement the Asset Operations portion of this feature you must:

- Download and install the CpqConfiguratorStoreApp and CpqConfiguratorAgentApp SSEs
- Create a ‘Modify’ user interface element which can be coded into the Asset Details widget (which is not elementized)
- Create an ‘Upgrade’ user interface element which can also be coded into the Asset Details widget

The creation of the user interface elements should be a straightforward process for any developer partner with a working knowledge of Commerce development and `knockout.js`.

Refer to Use Asset Based Ordering for more information on these Asset Operations.

Understand Sys Config model support via Commerce and the Oracle CPQ Configuration API

In Oracle CPQ, certain parts of customizable (configurable) products are based on "Sys Config" models that are accessible via the Oracle CPQ Configuration API. The "Sys Config" model consists of a hierarchy of components and associated classes that are used to model the hierarchical nature of the Product and Promotion structure of that configurable product.

When products in Oracle CPQ are structured hierarchically, Product Families are created first. Families provide the broad classifications of products. The next parts created are Product Lines which are used to describe more specific product areas of Product Families. Finally, Models are created to provide detail about the most specific product traits.

Note: In a "Sys Config" model, an attribute of a model can also be another model so it is important that you fully understand the structured hierarchy of each product family.

Examples of the product hierarchies just described might look something like the following:

- Product Family: "Business Laptop"
  - Product Line: “EZCompute”
    - Model: "EZ"
    - Model: "EZ Pro"
- Product Family: ""Gamer Laptop""
  - Product Line: "Avenger"
    - Model: "Novice"
    - Model: "EZ Pro"

Note: In a "Sys Config" model, an attribute of a model can also be another model so it is important that you fully understand the structured hierarchy of each configured product family. For example, the "Novice" model in the "Gamer Laptop" product family could have its own "sub-model" that had a variation of the features (more memory, better graphics card, and so on) offered in the basic configuration of the parent "Novice" model. To summarize, this feature lets you reload the configurator with a model which can be an attribute of the root/parent.

For more complete information on models and Oracle CPQ REST APIs, refer to the Oracle CPQ documentation.

In the Commerce and Oracle CPQ Integration, Commerce works with the Oracle CPQ Configuration API to let you execute the configuration of complex "Sys Config" models via API calls to the Oracle CPQ Configuration API. The Commerce support of the Oracle CPQ Configuration API lets you open and customize desired models within a bundle configuration. An example of this might be a product bundle consisting of a Mobile service attribute as well as a Cable TV service attribute. In this example, each service attribute (Mobile and Cable TV) is its own model. Commerce support of the

Oracle CPQ Configuration API lets you open a product configurator directly on either of those service models.

Note: Keep in mind that a shopper can only interact with a model starting from the root asset of the configured product. Every Configure, Reconfigure, Modify, or Upgrade operation is an operation carried out on the root asset. Having retrieved the root asset (the complete product model), the shopper may then navigate to any attributes of the root. In some cases, an attribute may well be an attribute that is a sub-model.

As far as user cases go, this feature lets you (the developer) build out specific user interface experiences dealing with the configuration of customizable products from a desired catalog. In doing so, it lets you apply global, site, or even product-specific user interface template changes as well as control the user interface flow of the configuration process for each product. For customers, this feature lets them enjoy a seamless product customization experience without any indication that multiple applications are working together as part of an integration to handle the product configuration.

Implement configuration customization via the CPQ Configuration API.

You need to complete some initial tasks to implement the functionality that directly customizes configurations using the Oracle CPQ Configuration API for the first time on a customer storefront.

The Direct API Configuration feature lets you directly customize configurations using the Oracle CPQ Configuration API. This topic describes the tasks which a developer and designer would work together to implement this functionality for the first time for a customer storefront.

This feature lets you directly customize configurations using the Oracle CPQ Configuration API. This topic describes the tasks which a developer and designer would work together to implement this functionality for the first time for a customer storefront. This would be the set of tasks that would be carried out first to allow you to use the feature.

In this case, the customer does not want to use the hosted iFrame model for executing product customization on their site but would prefer customization via the Direct API Configuration feature. The specific reasons the customer is requesting the implementation of this feature are the following:

- The customer wants the customization user experience to be as seamless as possible.
- The customer wants their merchandising team to have as much control over the customization user interface "look and feel" as possible.
- The customer would prefer that the merchandising team manage the user interface experience in their design tools as much as possible.

The details for implementing and using the Oracle CPQ Configuration API feature are described in the sections that follow. In these descriptions, it is assumed that the Commerce and Oracle CPQ Integration is already configured and enabled.

Understand the role of the Commerce Configurator SSE in the Direct API Configuration feature

The Direct API Configuration feature uses a Commerce server-side extension (SSE) to provide a collection of endpoints which lets the storefront UI (regardless of the UI framework used) to configure products and services. The SSE accepts a configurator request, invokes the corresponding requests in Oracle CPQ, and processes the Oracle CPQ response before returning an optimized payload.

The SSE performs the following configurator actions:

- Configure - This action corresponds to the Oracle CPQ `_configure` endpoint and is the starting point for configuring a model. It returns all the necessary layout data, attribute, and configuration state data for a user interface to display a configurator model. Also, where a layout contains Pick Lists and/or Array Sets, it returns all data required for those components to be rendered.
- Update - This action corresponds to the Oracle CPQ `_update` endpoint. It will accept an updated configuration state from the client and return an new configuration state based on the changes made.
- Next - This action corresponds to the Oracle CPQ `_next` endpoint. This action is available when the model configuration is spanned across multiple nodes/ configuration flow layouts. It works similarly to the initial configure action as it also returns all the necessary layout data, attributes, configuration state, and pick list/array set data to display the particular layout for a stage in the flow.
- Previous - This action corresponds to the Oracle CPQ `_next` endpoint. This action is available when the model configuration is spanned across multiple nodes/ configuration flow layouts. It works similarly to the initial configure action as it also returns all the necessary layout data, attributes, configuration state, and pick list/ array set data to display the particular layout for a stage in the flow.
- Add to Cart - This action corresponds to the Oracle CPQ`_integration_addToCart` endpoint. This action returns a Commerce commerce item (cart item). It transforms a `Configuration_Details` response (returned from Oracle CPQ) to a Commerce commerce item (cart item). With the embedded configurator, approach the `Configuration_Details` response is returned to Commerce and it is the responsibility of the Commerce client to transform the response to a Commerce commerce item.
- Reconfigure - This action corresponds to the Oracle CPQ `_reconfigureClient` endpoint. It is similar to the Configure action but rather than starting a brand new configuration, it returns all the necessary layout data, attributes, configuration state, and pick list/array set data for a user interface to display a configurator model for an existing configuration. A `configId` parameter is used to identify the existing configuration.
- Interact - This action corresponds to the Oracle CPQ `_interact` endpoint. It is typically triggered by the user interface in response to a change to an attribute value when `ajaxEnabled` has been set to true for the user interface component. It takes the value for the attribute that has changed and returns a new configuration state based on the change made.
- Array Set - The action supports the following:
  - Add Row - This action corresponds to the Oracle CPQ `_set<arraySetVarName>/actions/_add` endpoint. It accepts a `cacheInstanceId` and adds a row to the `arraySet`.

  - Delete Row - This action corresponds to the Oracle CPQ `_set<arraySetVarName>/actions/_delete` endpoint. It accepts a `cacheInstanceId` and `removeIndex` and removes the row from the supplied index in the `arraySet`.
- Layout - This action retrieves the full `layoutCache` for a particular product and flow.
- Pick Lists - This action retrieves all options available for a particular pick list.
- UI Settings - This action retrieves all general/base user interface configuration settings from Oracle CPQ.
- Templates - The action retrieves configuration templates that are to be used for rendering a BOM table and a recommended parts table.

Implement the Direct API Configuration feature

To implement the Direct API Configuration feature in Commerce you must:

- Download and install the Oracle CPQ Configurator (Storefront/Agent) Server-Side Extension in Commerce.
- Create a "Customize Button for Direct API" user interface element for direct API configuration.
- Create a "Reconfigure Button for Direct API" user interface element for direct API reconfiguration."
- Create a JavaScript Library of user interface components that will be used to render the Layout response from Oracle CPQ (this could be Knockout Components, React, Commerce elements etc.).
- Include the "Customize Button for Direct API" element (button) in the Product Details widget in order to trigger a customization session.
- Include the "Reconfigure Button for Direct API" element (button) in the Shopping Cart widget in order to trigger a reconfigure session.
- Bundle the user interface elements and JavaScript library into a single extension that can be uploaded in a single step.
- Log in to Commerce Admin and navigate to **Settings → Extensions**.
- Upload the Oracle CPQ Configurator server-side extension.
- Upload the new extension containing the user interface elements and JavaScript library.

Implement the Direct API Configuration feature for Configure

If you decide to implement the Direct API Configuration feature for Configure do the following:

- Log in to Commerce Admin and navigate to **Design → Layout → Product Layout → Layout Settings**.
- Select **Product Layout → Grid View** and then select the **Product Details** widget.
- Select the **Element Library**. You should see three "Customize Button" user elements available. These include the following:
  - Customize Button - Supports the iFrame customization flow by using the iFrame URL stored in Commerce Admin and appending values for Product

Line, Product Family and Model to load the iFrame and kick off the configuration process.

  - Customize Button with Configuration Metadata - Supports the iFrame customization flow by using the iFrame URL stored in Commerce Admin and appending values for Product Line, Product Family, Model and a collection of one or more static or dynamic key value pairs of configuration metadata to load the iFrame in the correct state and kick off the configuration process.
  - Customize Button for Direct API - Supports the API driven customization flow. **Note:** You created this as directed in the previous section as the "Customize Button for Direct API" element.

- Add the **Customize Button for Direct API** to the Product Details widget.
- Save your changes.
- Navigate back to **Layout** → **Product Layout** → **Layout Settings**.
- Set the Layout Preview Product ID for 4ForU Deal offer. This is a configurable product that lets you buy services for Landline, Mobile, Internet and TV in a single bundle at a steep discount.
- Save your changes. Select **Product Layout** → **Preview**. You are presented with a preview of the product layout for the 4ForU Deal offer.
- Select to customize the offer. You are presented with the customizable options for the offer in a combination of user interface components including the following:
  - Panels
  - Tabs
  - Input fields
  - Radio buttons
  - Checkboxes
  - Multi-select lists
  - Single select lists
  - Date pickers
  - Pick Lists

  These components are presented as the default mapping for the corresponding Oracle CPQ model attributes and layout.
- Publish your changes.

Implement the Direct API Configuration feature for Reconfigure

If you decide to implement the Direct API Configuration feature for Reconfigure do the following:

- Select **Layout** → **Cart Layout** → **Grid View** and select the **Shopping Cart** widget.
- Select **Go to widget code**.
- Add the **Reconfigure Button for Direct API** to the Shopping Cart widget. **Note:** You created this as directed in the earlier section as the "Reconfigure Button for Direct API" element.
- Save your changes.

- Navigate back to **Layout → Cart Layout → Layout Settings**.
- Set the Layout Preview Product ID for the 4ForU Deal offer with a quantity of 1. This is a configurable product which lets you buy services for Landline, Mobile, Internet and TV in a single bundle at a steep discount.
- Save your changes.
- Select **Product Layout → Preview**. You are presented with a preview of the product layout for the 4ForU Deal offer.
- Select to customize the offer and add it to the cart.
- Select the cart and choose to edit the configurable item.
- You are presented with customizable options for the offer in a combination of user interface components including the following:
  - Panels
  - Tabs
  - Input fields
  - Radio buttons
  - Checkboxes
  - Multi-select lists
  - Single select lists
  - Date pickers
  - Pick Lists

  These components are presented as per the default mapping for the corresponding Oracle CPQ model attributes and layout.
- Publish your changes.

Commerce is now configured to use the direct API configuration process for customizable products.

Control user interface look and feel using the CPQ Configuration API

The Direct API Configuration feature lets you control user interface "look and feel" using the Oracle CPQ Configuration API.

You can use the Direct API Configuration feature to control user interface "look and feel" using the Oracle CPQ Configuration API. This ability lets you do things like the following:

- Apply a site-specific "Look and Feel" product customization to the user interface experience.
- Apply site-specific user interface components for a custom user interface experience.
- Add a new UI component to the configuration flow.
- Remove tabs from the product customization user interface experience.

- Apply a product type specific set of user interface components to the configuration flow.

Before you can accomplish these tasks, you must first make sure that the API driven configuration feature has been implemented (described in the previous topic). Also, it is assumed that the Commerce and Oracle CPQ Integration has already been configured and enabled.

In the sections that follow, you are provided with details for using this feature to carry out these customization tasks.

Apply a site-specific "Look and Feel" product customization to the user interface experience

Consider this situation. Say a customer wants a new custom user interface look and feel for their site. The customer's in-house design and brand management team have provided specifications as to:

- Color Schemes
- Style Header and Footer
- Navigation
- Buttons, input fields, check boxes, Multi-select Lists, single select Lists, date pickers, pick lists
- Component Sizes
- Component Styles
- Component Colors
- Component Fonts

You are instructed to change the user site interface look and feel so that it reflects the customer product customization changes. This is done by completing the following tasks:

- Refer to the Customizing your store layouts section on the Oracle Help Center. You can see that it is possible to apply the required user in look and feel by cloning and customizing a Commerce theme.
  Note: The included version of the JavaScript Library of Knockout UI Components used to render the Layout response from Oracle CPQ uses OOTB theme/styles, (i.e., Bootstrap Forms and Components). Also, by making changes at the provided Theme level, you can change the look and feel of the configuration UI experience without making any changes directly to the UI elements or JS Library.
- Clone the provided the theme and apply the required specifications for:
  - Backgrounds
  - Buttons
  - Navigation Menu
  - Menu
  - Typography

  This is done directly in the Design page.
- Use the Design page to access the theme's CSS and apply all of the remaining UI specifications.

- Save all your changes.
- Navigate to **Layout → Product Layout → Layout Settings**.
- Set the Layout Preview Product ID for 4ForU Deal offer, this is a configurable product which allows shoppers to buy services for Landline, Mobile, Internet and TV in a single bundle at a steep discount.
- Save your changes.
- Select **Product Layout → Preview**. You are presented with a preview of the product layout for the 4ForU Deal offer.
- The system displays the configurable options available in a combination of UI components such as the following:
  - Panels
  - Tabs
  - Input fields
  - Radio buttons
  - Checkboxes
  - Multi -select lists
  - Single select lists
  - Date pickers
  - Pick Lists

You can now see that all of the user interface components are displayed in accordance with the new theme that you have created and are in accordance with the rest of the site.

Apply site-specific user interface components for a custom user interface experience

A case may arise where a customer wants the customization user interface experience to be slightly different from the rest of the site to convey the feeling of personal design when they are building their tailored product.

The customer's in-house design and brand management team has provided specifications to make changes to the following user interface elements:

- Buttons - Primary Buttons should contain an icon
- Input Fields - Should all have labels
- Check boxes - Should be rendered as sliders
- Multi-select lists - Should be displayed as a collection of check boxes
- Single select lists - Should be displayed as drop down lists
- Date pickers - Should be displayed as Tumbler Scrolls
- Color pick list - Should be displayed as a swatch matrix with a tone slider

As a member of the SI user interface design team, you are instructed to implement the new product customization user interface look and feel. You see that in order to change how the Oracle CPQ model user interface components are rendered inCommerce, changes must be made to the JavaScript Library of Knockout user interface components used to render the Layout response from Oracle CPQ. This

JavaScript Library is part of the Oracle CPQ Configurator user interface extension which was uploaded at feature implementation time.

To implement the new product customization user interface look and feel, complete the following tasks:

- Log in to Commerce Admin and navigate to **Settings** → **Extensions**.
- Deactivate the Oracle CPQ Configurator user interface extension.
- Delete the Oracle CPQ Configurator user interface extension. This extension includes the Direct API versions of the Configure and Reconfigure user interface elements as well as a common JavaScript Library that defines the mapping of Oracle CPQ user interface components to Commerce Knockout Components.
- Create new versions of the following:
  - Configure element (if you want the button to appear differently or launch the configuration in a new widget)
  - Reconfigure element (if you want the button to appear differently or launch the configuration in a new widget)
  - JavaScript Library (In the JavaScript library for each component that is to be rendered differently modify the HTML, JavaScript and define new styles which must also be added to the global stylesheet).
- Bundle the user interface elements and JavaScript library into a single extension that can be uploaded in a single step.
- Navigate to **Settings** → **Extensions** and upload the new version of Oracle CPQ Configurator user interface extension.
- Reapply the "Customize via direct API" for Configure by doing the following:
  - Navigate to **Design** → **Layout** → **Product Layout** → **Layout Settings**.
  - Select **Product Layout** → **Grid View** and select the Product Details widget.
  - Select the Element Library.
  - Add the Customize Button for Direct API to the Product Details Widget.
  - Save your changes.
  - Navigate back to **Layout** → **Product Layout** → **Layout Settings**.
  - Set the Layout Preview Product ID for 4ForU Deal offer. This is a configurable product which allows shoppers to buy services for Landline, Mobile, Internet, and TV in a single bundle at a steep discount.
  - Save your changes.
  - Select **Product Layout** → **Preview**. You are presented with a preview of the product layout for the 4ForU Deal offer.
  - Select to customize the offer. You are presented with the customizable options for the offer in a combination of user interface components including the following. These are presented as per the new Knockout user interface components.
    * Buttons
    * Input Fields
    * Checkboxes

        * Multi-select lists
        * Single select list
        * Date pickers
        * Color pick list
- Add the customized offer to the cart.
- Select the cart and chooses to edit the configure item. You are presented with the customizable options for the offer in a combination of user interface components. These are presented as per the new Knockout user interface components. These include the following:
    - Buttons
    - Input Fields
    - Checkboxes
    - Multi-select lists
    - Single select list
    - Date pickers
    - Color pick list
- Publish your changes.

Upon completing these tasks, you will see that the product customization user interface look and feel and components are now distinct from the store design theme and in accordance with the customer's specifications.

Add a new user interface component to the configuration user interface flow

Sometimes a customer may want new to add a new user interface component that shoppers will use to select an image that will be imprinted on the shopper's mobile phone case.

In this example, the customer's in-house design and brand management team have developed a new "Image Carousel" user interface component that shoppers will use to select the image to be imprinted. This new user interface component is used as the user interface control for Oracle CPQ model attributes which require the shopper to select an image.

As a member of the SI user interface design team, you are instructed to ensure that this new user interface component is displayed correctly in Commerce. To add the new user interface component to the configuration user interface experience via direct API, complete the following tasks:

- Log in to Commerce Admin and navigate to **Settings** → **Extensions** and do the following:
    - Deactivate the Oracle CPQ Configurator user interface extension.
    - Delete the Oracle CPQ Configurator user interface extension. This extension includes the direct API versions of the Configure and Reconfigure user interface elements as well as a common JavaScript Library of user interface Components used to render the Layout response from Oracle CPQ.
- Create new versions of the JavaScript Library to include the new 'Image Carousel' user interface component, including HTML, JavaScript and Style Definitions which must also be added to the global stylesheet.

- Bundle the user interface elements and new JavaScript library into a single extension that can be uploaded in a single step.
- Navigate to **Settings → Extensions** and upload the edited version of the Oracle CPQ Configurator user interface extension.
- Reapply the '"Customize via direct API" for Configure.
- Preview the product layout and make sure that the new image carousel user interface component renders correctly when customizing a product.
- Preview product layout and make sure that the new image carousel user interface component renders correctly when reconfiguring a product.
- Publish your changes.

Upon completing these tasks, you should see that the product customization user interface now includes a new user interface component in accordance with the customer's specifications.

Remove tabs from the product customization user interface experience

In this case, the customer's in-house design and brand management team have requested that all tabs be removed from the product customization user interface as they have received feedback from customers that they are confusing.

As a member of the user interface design team, you are instructed to remove all tabs from the customization user interfaces. To remove all tabs, complete the following tasks:

- Login to Commerce Admin and navigate to **Settings → Extensions.**
- Deactivate the Oracle CPQ Configurator user interface extension.
- Delete the Oracle CPQ Configurator user interface extension. This extension includes the direct API versions of the Configure and Reconfigure user interface elements as well as the JavaScript Library of user interface Components.
- Edit the JavaScript Library to change how tabs are rendered (stacked, side by side etc.)
- Navigate to **Settings → Extensions** and upload the edited version of Oracle CPQ Configurator user interface Extension.
- Reapply the "Customize via direct API" for Configure Preview the product layout and make sure that there are no tabs displayed when customizing a product.
- Preview the product layout and make sure that there are no tabs displayed when reconfiguring a product.
- Publish you changes.

Upon completion of these tasks, you will note that the product customization user interface no longer displays any tabbed layout in accordance with the customer's specifications.

Apply a product type specific set of user interface components to the configuration flow

In this case the, the customer's in-house design and brand management team want the shopper's configuration experience to be different when they customize shippable goods (for example, "Build your own laptop") and when they customize services such as the Phones4All offer.

For this, a new set of "Service Configuration user interface Components" has been developed by the in-house design and brand management team for the following:

- Buttons
- Input Fields
- Checkboxes
- Multi-select lists
- Single select list
- Date pickers
- Color pick list

As a member of the user interface design team, you are instructed to ensure that when a shopper is customizing a service these new user interface components will be displayed. This is done by completing the following tasks:

- Log in to Commerce Admin.
- Navigate to **Settings** → **Extensions**.
- Deactivate the Oracle CPQ Configurator user interface extension.
- Delete the Oracle CPQ Configurator user interface Extension. This extension includes the direct API versions of the Configure and Reconfigure user interface elements as well as the common JavaScript Library.
- Edit the JavaScript Library by adding conditional `IF` statements that map the Oracle CPQ user interface components to the new "Service Configuration user interface Components" where `Product Type = Service`.
- Navigate to **Settings** → **Extensions** and upload the edited version of Oracle CPQ Configurator user interface extension.
- Reapply the "Customize via direct API" for Configure.
- Publish your changes.
- Create a new "Services Product Layout" for products where `Product Type = Service`.
- Create a new "Service Product Details Widget."
- Add the "Customize Button for direct API" user interface element to the Product Details Widget.
- Add the "Service Product Details Widget" to the "Services Product Layout."
- Save your changes.
- Navigate back to **Layout** → **Services Product Layout** → **Layout Settings**.
- Set the Layout Preview Product ID for 4ForU Deal offer.
- Select **Product Layout** → **Preview**. You are presented with a preview of the product layout for the 4ForU Deal offer.
- Select to customize the offer. You are presented with the customizable options for the offer in a combination of user interface components. This includes each of the new "Service Configuration user interface Components." This includes the following:
  - Buttons

  - Input Fields
  - Checkboxes
  - Multi-select lists
  - Single select list
  - Date pickers
  - Color pick list

  These are now presented correctly.

- Publish your changes.

Upon completing these tasks, the product customization user interface now displays the new product type specific user interface components in the configuration flow.

Customize and reconfigure a product by direct use of the CPQ Configuration API

You can customize and reconfigure a product by directly using of the Oracle CPQ Configuration API.

You can use the Direct API Configuration feature to customize a product by implementing and using the Oracle CPQ Configuration API. This feature give you the ability to do the following:

- Customize a product where the "Customize via direct API" feature has implemented in Commerce
- Reconfigure a product before checking out

Before you can accomplish these tasks, you must first make sure that the Direct API Configuration feature has been implemented (described in a previous topic of this section of the guide). Also, it is assumed that the Commerce and Oracle CPQ Integration is already configured and enabled. In the section that follows, you are provided with details for using the feature to carry out these specific customization tasks as just described.

Apply customizations to a product by directly using the Oracle CPQ Configuration API

The list of tasks that follow describe a situation where a shopper customizes a product where the Direct API Configuration feature has been implemented in Commerce.

In this case, a System Integration Partner has already implemented the feature and the SI user interface design team may have already done some user interface customizations by directly using the Oracle CPQ Configuration API.

For this example, it is assumed that the Commerce and Oracle CPQ Integration is already configured and enabled.

Use the following guidelines to accomplish the specified goals.

- As an example, let us say that the shopper has noticed a lot of web advertising by their cell phone service for their new Phones4All offer which allows them to buy a single deal with phones and plans for up to 6 people at huge savings on handsets, accessories and monthly bills.

- The shopper navigates to their cell phone service and selects the Phones4All offer. The shopper selects to customize the offer. The UI element **Customize Button for Direct API** invokes the `.../v1/configurations` SSE endpoint passing the following parameters:
  - `productFamily`
  - `productLine`
  - `model`
  - `locale`
  - `currency`
  - `configurationMetadata`
- The `.../v1/configurations` SSE endpoint triggers the following Oracle CPQ API endpoints:
  - `GET_configUISettings`
  - `GET_pageTemplates`
  - `POST_config`
  - `GET_Layout_ Cache`
- The .../v1/configurations SSE collates the data returned from Oracle CPQ, strips out all extraneous information, and returns a "combined configuration data response."
- The shopper is presented with a set of customization options that they can use to tailor the offer to their specific needs.
- The first option the shopper is presented with is the number of lines required.
- The shopper selects 4 lines.
- The shopper selects **Next**.
- The UI element **Customize Button for Direct API** invokes the `.../v1/configurations/{cacheInstanceId}/page` SSE endpoint (where `cacheInstanceId` represents the current reconfiguration instance in Oracle CPQ) by passing the following parameters:
  - `productFamily`
  - `productLine`
  - `model`
  - `locale`
  - `currency`
  - `op: next`
- The `.../v1/configurations/{cacheInstanceId}/page` SSE endpoint triggers the following Oracle CPQ API endpoints:
  - `POST_next`
  - `GET_Layout_ Cache`
- The `.../v1/configurations/{cacheInstanceId}/page` SSE collates the data returned from Oracle CPQ, strips out all extraneous information, and returns a "combined configuration data response."

- The shopper is presented with the configuration options for Handset and Plan for Line 1 including:
  - Handset - including Capacity, Color, Tablet, and Watch
  - Plan - Silver or Gold
- The shopper selects the "Samsung S10" handset
- The UI element **Customize Button for Direct API** checks the `isUpdatable` property for the handset attribute.
- The `isUpdatable` property value is TRUE (this means that when an option is selected for this attribute, the configuration model must be updated as this selection impacts other model attributes).
- The UI element **Customize Button for Direct API** invokes the `.../v1/configurations/{cacheInstanceId}` SSE endpoint (where `cacheInstanceId` represents the current reconfiguration instance in Oracle CPQ) passing the following parameters:
  - `productFamily`
  - `productLine`
  - `model`
  - `locale`
- The `.../v1/configurations/{cacheInstanceId}` SSE endpoint triggers the `POST_update` Oracle CPQ API endpoint.
- The `.../v1/configurations/{cacheInstanceId}` SSE collates the data returned from Oracle CPQ, strips out all extraneous information and returns a "combined configuration data response."
- The shopper sees that some of the options that were previously available for capacity, color, table and watch have been updated and that they are now limited to those compatible with their selected Samsung S10 handset.
- The shopper selects the 256GB capacity option for the handset.
- The UI element **Customize Button for Direct API** checks the `isUpdatable` property for the handset attribute. The `isUpdatable` property value is FALSE (this means that when an option is selected for this attribute the configuration model need not be updated as this selection does not impact other model attributes).
- The shopper completes the customization for Line 1 and moves on to line 2.
- When the shopper is part way through the customization of Line 2, they decide that they may need to make a change to the handset capacity for Line 1.
- The shopper selects **Previous**.
- The UI element **Customize Button for Direct API** invokes the `.../v1/configurations/{cacheInstanceId}/page` SSE endpoint (where `cacheInstanceId` represents the current reconfiguration instance in Oracle CPQ) passing the following parameters:
  - `productFamily`
  - `productLine`
  - `model`
  - `locale`

  - `currency`
  - `op: previous`
- The `.../v1/configurations/{cacheInstanceId}/page` SSE endpoint triggers the `POST_previous` and `GET_Layout_ Cache` Oracle CPQ API endpoints.
- The `.../v1/configurations/{cacheInstanceId}/page` SSE collates the data returned from Oracle CPQ, strips out all extraneous information, and returns a "combined configuration data response."
- The shopper is presented with the configuration options that they selected for Line 1.
- The shopper changes the capacity for the Line 1 handset and continues to customize the rest of the lines.
- The shopper completes the customization of their Phones4All offer.
- The shopper selects **Add to Cart**.
- The UI element **Customize Button for Direct API** invokes the `.../v1/configurations/{cacheInstanceId}/add-to-cart` SSE endpoint (where `cacheInstanceId` represents the current reconfiguration instance in Oracle CPQ) passing the following parameters:
  - `productFamily`
  - `productLine`
  - `model`
- The `.../v1/configurations/{cacheInstanceId}/add-to-cart` SSE endpoint triggers the POST_integration_add_to_cart Oracle CPQ API endpoint.
- The `.../v1/configurations/{cacheInstanceId}/add-to-cart` SSE transforms the Oracle CPQ response to a Commerce cart item and adds the configured item to the Commerce cart.
- The shopper proceeds to checkout.

When all of this has completed, a multi-level configured item is added to Commerce cart.

Reconfigure a customized product before checking out

In this situation, a shopper decides to make a change to a customized product after adding it to the cart but before checking out.

Say, for example, the customer has customized their Phones4All offer and has added it to the cart. Before checking out, however, the shopper reviews their choices and realizes that by including the Apple Watch with Line 4 the offer is more than $200 over their budget. The following details illustrate what occurs if a typical shopper wishes to reconfigure an already customized product before checking out:

- The shopper selects to edit the Phones4All item in her cart.
- The user interface Shopping Cart widget with a Reconfigure Button for Direct API invokes the `.../v1/configurations` SSE endpoint passing the following parameters:
  - `productFamily`
  - `productLine`

  - `model`
  - `locale`
  - `currency`
  - `configId` (identifies the specific instance of configuration in Oracle CPQ which is to be reconfigured)
- The `.../v1/configurations` SSE endpoint triggers the following Oracle CPQ API endpoints:
  - `GET_configUISettings`
  - `GET_pageTemplates`
  - `POST_config`
  - `GET_Layout_ Cache`
- The `.../v1/configurations` SSE collates the data returned from Oracle CPQ, strips out all extraneous information, and returns a "combined configuration data response."
- The shopper is presented with all of the customization options and selections that they have made.
- The shopper navigates to Line 4 and removes the Apple Watch selection.
- The shopper selects to save and their cart is updated.

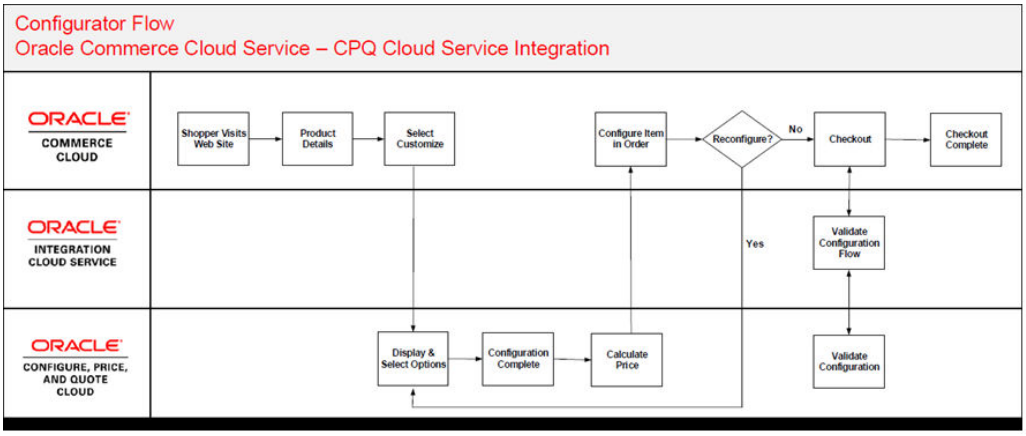
The Commerce cart is now updated with the newly reconfigured item.

A

Appendix A: Configurator Flow

A Configurator process flow occurs between Oracle CPQ and Commerce during the integration.

The following presents a diagram of the integration Configurator Flow:

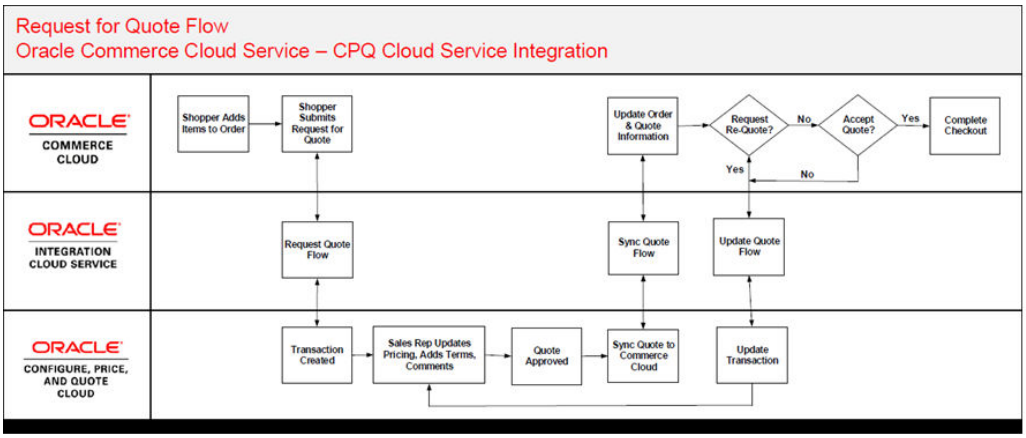


B

Appendix B: Request for Quote Flow

A Request for Quote process flow occurs between Oracle CPQ and Commerce during the Quote integration.

The following presents a diagram of the Request for Quote integration flow between Commerce, OIC, and Oracle CPQ Cloud when using theOracle CX Commerce-Oracle CPQ Quote integration



Glossary

Index