



# Oracle CPQ with Subscription Management Integration Guide



Oracle CPQ Updates 21D and Later

January 2023

Copyright © 2023, Oracle and/or its affiliates

# TABLE OF CONTENTS

Revision History.....	3
Introduction.....	3
Purpose.....	4
Audience.....	4
Prerequisites.....	4
Acronym List.....	4
Subscription Flows Overview.....	6
Product Modeling Setup.....	8
Oracle CPQ Package Installation and Setup.....	10
Create OIC Integration.....	10
Install Subscription Management Package.....	11
Define Email Notifications for Subscription Renewals.....	11
Subscription Pricing Setup.....	12
OIC Integration Installation and Setup.....	13
Import Packages and Integrations.....	13
Configure Oracle CPQ and OSS Connections in OIC.....	14
Configure Oracle CPQ and OSS Usage Rating Connections in OIC.....	15
Register Third Party Application for Subscription in Fusion.....	16
OIC Lookup Details.....	18
OIC Mapping Details.....	19
Oracle CPQ Field Setup.....	24
Business Unit ID Field.....	24
Subscription Profile ID Field.....	24
Account Fields.....	24
Billing Frequency – Price Periodicity Field.....	25
Part Custom Fields.....	26
Layout Fields.....	27
Demo Product Setup.....	28
Install the BOM Data Table Packages.....	29
Install the Parts Package.....	29
Install the Vision Vehicles SUV Demo Product Package.....	30
Verify the Addition of All BOM Parts.....	31
Deploy the Home Page.....	31
Installed Oracle CPQ Elements.....	32
Commerce Attributes.....	32
Commerce Actions.....	32
Library Functions.....	40
Validation Rules.....	40
Hiding Rules.....	40
Workflow Steps and Step Transitions.....	41
Timer Configuration.....	42
Oracle CPQ Account Integration.....	43
Library Functions.....	43
Manual Data Table Changes.....	44
Add Template Dependencies to File Manager.....	45
Oracle CPQ Account Lookup Integration.....	46
Account REST API Services.....	48
Reference Accounts Integration.....	52
Account Search Data Tables.....	52
Subscription Workbench.....	54
Subscription Pricing Integration.....	59
Enable Subscription Pricing.....	59
Charges.....	59
Discounts.....	59
Pricing Engine Setup.....	60
Appendix A: Create Subscription Workflow.....	63

Appendix B: Amend Subscription Workflow .....	65
Appendix C: Add Amended Lines to Existing Subscription .....	68
Appendix D: Renew Subscription Workflow .....	69
Appendix E: Terminate Subscription Workflow .....	70
Appendix F: Commerce Attributes .....	71
Appendix G: Update Asset Timer BML .....	76
Appendix H: Customer Details BML .....	79
Appendix I: Open Transaction Line BML .....	82
Appendix J: Save BML .....	83
Appendix K: Payload Template File Content .....	84
Appendix L: Library Function BML .....	85
Appendix M: Subscription Pricing Utility BMLs .....	90
Appendix N: Implementation for Discount Effectivity Types .....	98
Appendix O: Calculate Price API .....	102
Appendix P: Troubleshooting .....	104
Manually Add BOM Parts to Oracle CPQ .....	104
Warning Message with Initial Install of Subscription Management Package .....	104
Resolve Issues with Submit Order Action .....	104
Enable the OSS Renew Event in the OIC Environment .....	105

## REVISION HISTORY

DATE	WHAT'S CHANGED	NOTES
12 JAN 2023	Updated content in the following sections: <ul style="list-style-type: none"><li>• <a href="#">Install Subscription Management Package</a></li><li>• <a href="#">Configure Oracle CPO and OSS Usage Rating Connections in OIC</a></li><li>• <a href="#">Appendix P: Troubleshooting</a></li></ul>	Document revised for additional content clarification.
01 OCT 2021	All sections of the document were updated and the following new sections were added: <ul style="list-style-type: none"><li>• <a href="#">Appendix N: Implementation for Discount Effectivity Types</a></li><li>• <a href="#">Appendix O: Calculate Price API</a></li></ul>	Document revised for 21D support for granular pricing for subscription products and new Vision Vehicles SUV demo product.
07 JUL 2020	Updated content in the following sections: <ul style="list-style-type: none"><li>• <a href="#">Prerequisites</a></li><li>• <a href="#">Configure Oracle CPQ and OSS End-to-End Integration and Renew Event Connections in OIC</a></li></ul>	Document revised for Oracle CPQ and Oracle CX Sales product branding and content clarification.
13 SEP 2019	Added the following sections: <ul style="list-style-type: none"><li>• <a href="#">Oracle CPO Account Lookup Integration</a></li><li>• <a href="#">Subscription Workbench</a></li></ul>	Document revised for Oracle CPQ 19B features.
21 FEB 2019		Initial document creation for Oracle CPQ 19A.

## INTRODUCTION

Businesses across the industries are looking to adapt the change in buyer behavior and embrace subscription business models. In order to address these needs, Oracle recently launched the Oracle Subscription Management cloud application. By integrating front and back office business processes on one platform, Oracle Subscription Management allows organizations to build predictable, recurring revenue models by providing an end-to-end subscription solution that manages billing and revenue recognition and also informs customer-facing personnel with a complete view of purchasing behavior.

As part of the end-to-end subscription solution, Oracle CPQ 21D provides an integration with the Oracle Subscription Management application. This allows customers to create and manage products and services they can sell using a subscription model. The subscription model supports the ability to manage a given product or service as a recurring or usage-based price item.

Leveraging this integration, customers can enable their sales teams to capture subscription orders and perform subscription management activities throughout the lifecycle of these customer relationships. The integration, enabled by Oracle's next generation Oracle Integration Cloud (OIC) middleware, sometimes referred to as Oracle Autonomous Integration Cloud Service (OAIC) or Integration Cloud Service (ICS), comes with a Subscription Management package that includes installable artifacts for both Oracle CPQ and OIC.

The following functionality is available with the Subscription Management integration:

- Create a subscription in OSS by creating and submitting a Transaction in Oracle CPQ.
- Amend a subscription by changing the subscription quantity, duration, or product.
- Renew an existing subscription.

- Terminate an active subscription.
- Use the Rating/Pricing Engine to display subscription charges.
- Set up the Subscription Management integration using integration resources.

**Note:** Oracle Subscription Management (OSS) is a separately licensed product. Please contact your Oracle Sales representative for more details.

## Purpose

This installation guide describes how to implement the reference integration between Oracle CPQ and OSS using OIC.

## Audience

This installation guide is intended for administrators responsible for setting up and configuring the Oracle CPQ - Subscription Management solution. This guide assumes administrators have prior Oracle CPQ, OSS, and OIC administration experience.

## Prerequisites

Administrators must integrate the Oracle CPQ Release 21D or later Sales Cloud Reference Application image, commonly referred to as the “Ref App”, with the following:

- OSS Release 21D or later using OIC 15.4.3 or later middleware, which is used to establish an integration between Oracle CPQ and OSS.
- [Subscription Ordering Package](#) which is used to support the Subscription Management amend, renew, and terminate workflows. For additional information, refer to [Doc ID 2182966.1](#) on [My Oracle Support](#).
- [Customer Data Management \(CDM\) Integration](#) which supports account integration.

Optionally, Oracle recommends integration with Oracle CX Sales (formerly called Oracles Sales Engagement Cloud) to enable account, user, product synchronization, and reconciliation of Oracles CX Sales opportunities with Oracle CPQ Transactions. For information, refer to [Doc ID 2015009.1](#) on [My Oracle Support](#) for the Oracle CPQ – Oracle CX Sales Integration Guide.

### Notes:

- Administrators performing the Subscription Management installation must have the Oracle CPQ, OSS, and OIC sites setup with administrator privileges.
- An error will not display to end users when a BML integration queries a non-existing external integration (e.g. the CDM integration for EBS).
- For information about how to obtain any of the above prerequisites, contact an [Oracle sales representative](#).

## Acronym List

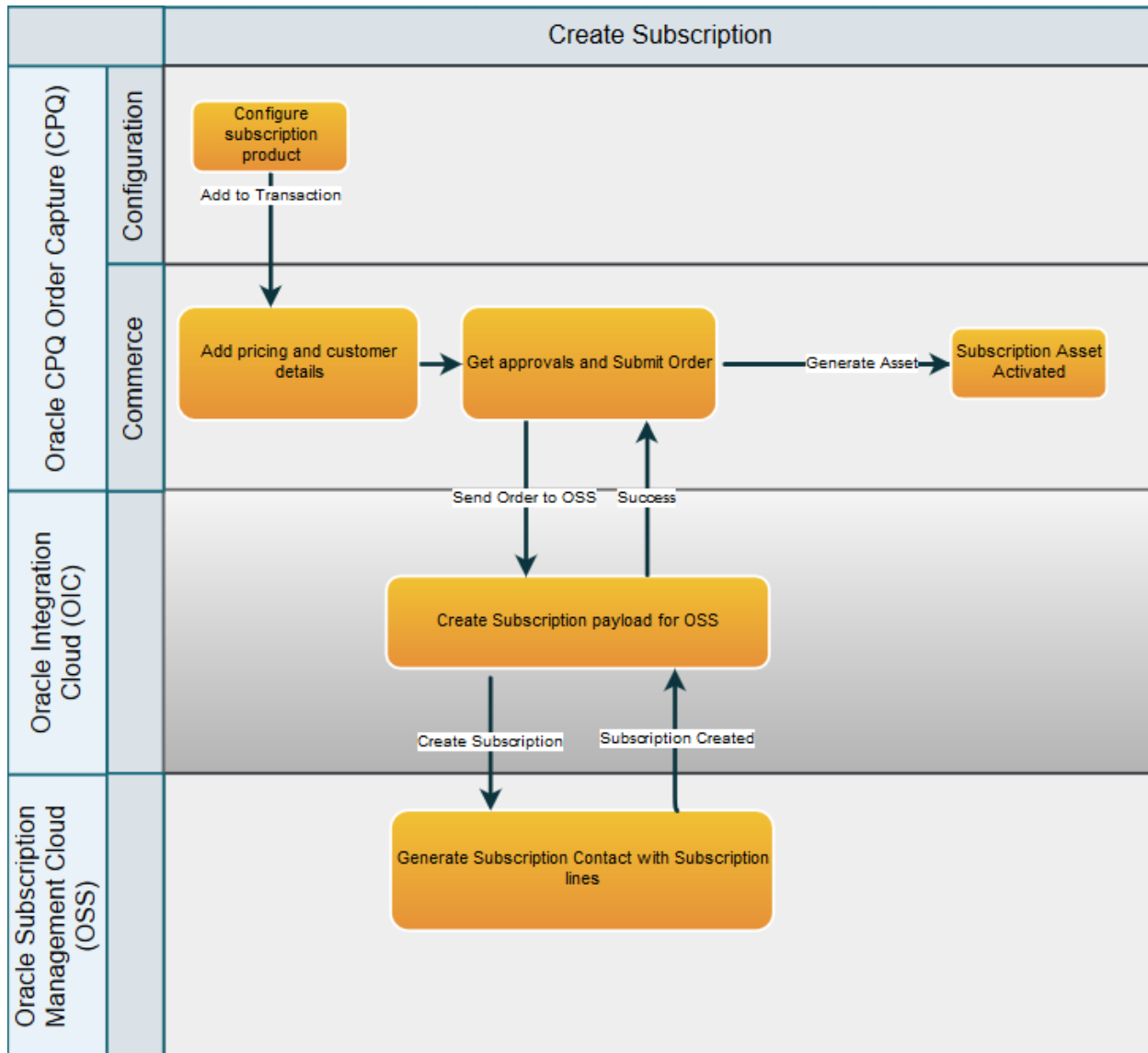
Definitions of the acronyms used within this document are provided in the below table. For additional information, refer to the Oracle CPQ Administration Online Help.

ACRONYM	DEFINITION	DESCRIPTION
BML	Big Machines Extensible Language	A scripting tool used to capture a company's complex business logic within Oracle CPQ Configuration and Commerce. BML is based on many different programming languages.
BOM	Bill of Material	Fulfillment systems often maintain BOMs containing complex, multi-level part structures that differ from the Configuration attributes used in Oracle CPQ when sales users configure products. BOM Mapping provides a data-driven mechanism

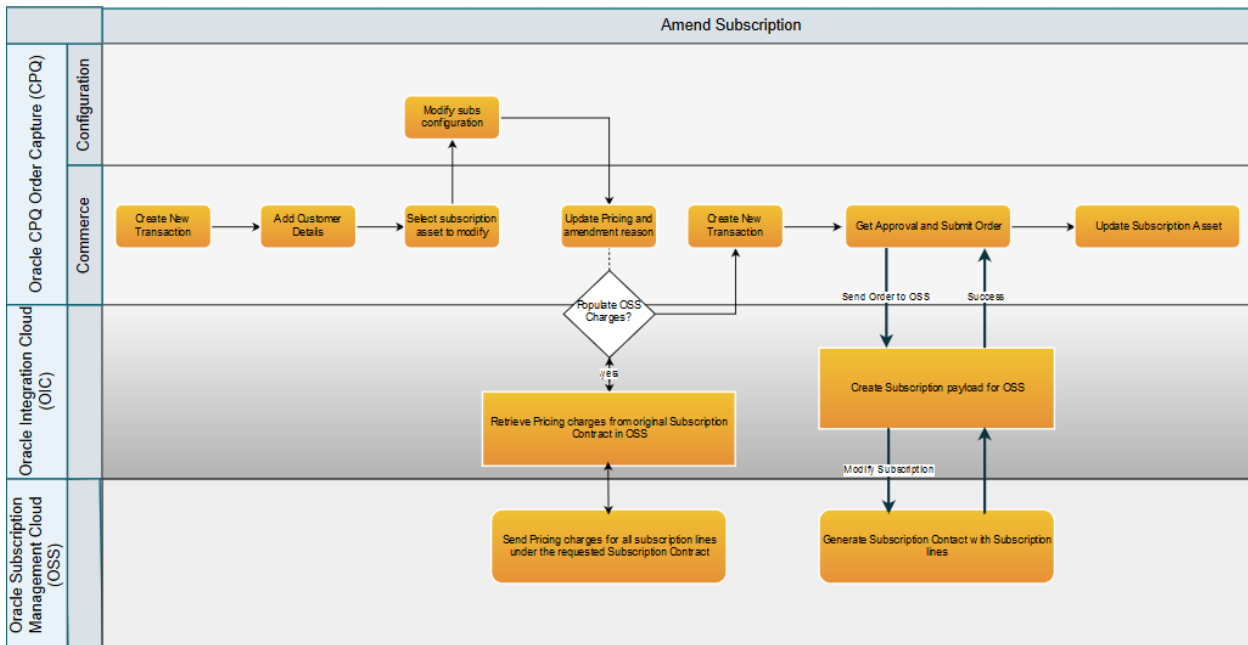
ACRONYM	DEFINITION	DESCRIPTION
		for mapping these differing product views. To use the Subscription Management solution to create a subscription from an Oracle CPQ Transaction, the subscription products must be modeled as a BOM.
CDM	Customer Data Management	An integration and common object for EBS and Fusion. The Subscription Management solution retrieves account information from CDM, which functions as the account master.
CPQ	Configure, Price, and Quote	The Oracle solution that enables companies to streamline their entire opportunity-to-quote-to-order process, including product selection, configuration, pricing, quoting, ordering, and approval workflows.
EBS	E-Business Suite	A comprehensive set of integrated and global business applications for managing and automating processes across an Enterprise: EBS Customer, EBS Order Management, EBS Material Reservation, and EBS Inventory On Hand Balance. CDM is a common object for EBS.
Fusion PIM	Product Information Management	The product master for OSS subscriptions. Administrators have the option of replacing PIM with an alternate solution such as Salesforce.
OIC	Oracle Integration Cloud	The Oracle middleware used by Oracle CPQ to provide an all-encompassing, standard Oracle solution for all integration needs. By using OIC, system integrators can manage all Oracle CPQ integrations from a single location with a consistent toolset.
OSS	Oracle Subscription Management	Allows organizations to build predictable, recurring revenue models by providing an end-to-end subscription solution that manages billing and revenue recognition and also informs customer-facing personnel with a complete view of purchasing behavior.
WSDL	Web Services Description Language	When setting up Oracle CPQ and Subscription Management Cloud connections in OIC administrators must enter the Oracle CPQ WSDL URL that integrates with OIC.

## SUBSCRIPTION FLOWS OVERVIEW

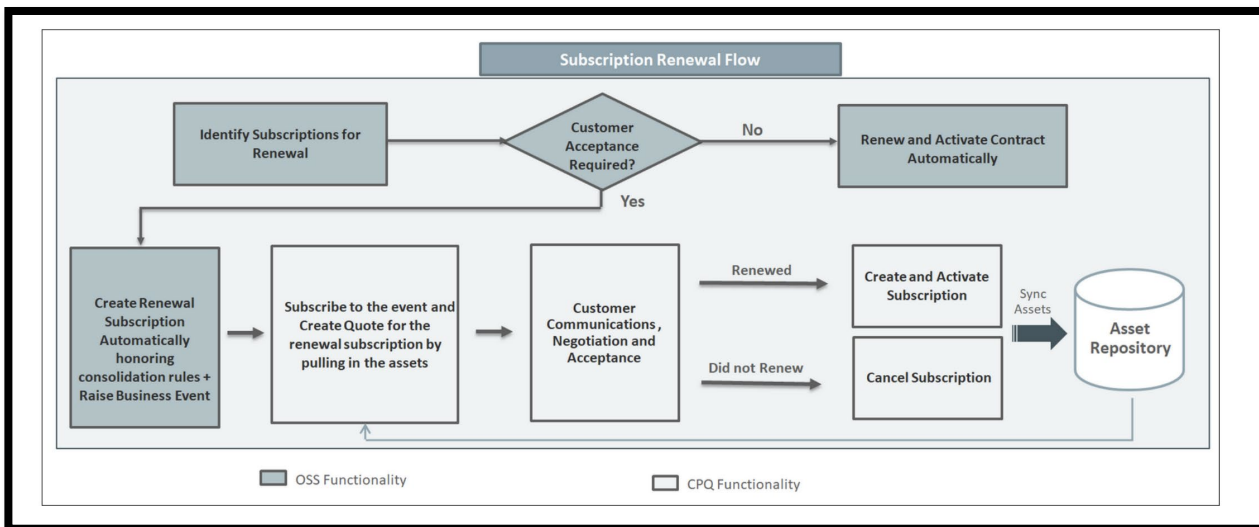
The following diagrams show the integration data flow between the involved applications in the Subscription Management solution.



Create Subscription



### Amend Subscription



### Renew Subscription



## PRODUCT MODELING SETUP

The steps for creating an Oracle CPQ Transaction for a subscription product mastered in PIM are summarized below and are also included in the *Oracle CPQ– Oracle CX Sales Integration through Oracle Integration Cloud Service Implementation Guide* located on [My Oracle Support \(Doc ID 2075213.1\)](#).

1. **Integrate Oracle CPQ with Oracle CX Sales:** Integrating Oracle CX Sales (formerly called Oracle Sales Engagement Cloud) with CPQ provides customers seamless support for the entire selling cycle. The integration allows users to create Transactions and sales orders for an opportunity originating in Oracle CX Sales.
2. **Synchronize Products:** Oracle CX Sales comes provisioned and fully integrated with the Oracle Fusion Product Model module, which is used to create and manage product items for use in Oracle CPQ. Products are referred to as parts in CPQ and as items in Oracle Product Information Management.

The synchronization process pushes or pulls new and updated products between Oracle CPQ and the Fusion Product Model. Users can create products in the Oracle Fusion Product Model and synchronize the products into CPQ. Likewise, they can create parts in Oracle CPQ and synchronize them into the Oracle Product Information Management.

### Notes:

- To create an Oracle CPQ Transaction using the Subscription Ordering flow, users must model the subscription product as a BOM in Oracle CPQ.
- The Subscription Management Integration supports the use of System Configuration BOMs.
- Administrators can use a root model to bundle multiple offerings of subscription products.
- Use the BOM definition to determine whether to update the old subscription or create a new subscription during the Amend flow.
- The reference integration illustrated in this guide assumes that the customer's Oracle CPQ environments are provisioned with a Reference Application, also known as a "Ref App image", which comes with a set of Commerce attributes and actions that administrators can configure to meet their business requirements. Customers who do not have the Reference Application deployed on their CPQ environment can create these Commerce attributes and actions as mentioned in this guide to complete the below described flows.
- In Oracle CPQ 21C and earlier, subscription product line items under the Model part were rolled up into one Model-level price and part number. Oracle CPQ 21D introduces a new subscription pricing structure that provides a consistent pricing structure for subscription and non-subscription products by allowing single price type for each line item. With the new pricing structure, a price change can be made at a more granular level since every charge is represented with a distinct part. Every line item/part charge is able to roll-up to the Model price. Refer to [Pricing Engine Setup](#) for price setup details.

### **Sample Scenario: Modeling for the Vision Vehicles SUV Subscription Product**

In this sample scenario, Outright SUV Buy and SUV Lease are two Oracle CPQ subscription offerings. To create a subscription from an Oracle CPQ Transaction, model these subscription products as a BOM. Since users can switch between products when performing a product upgrade or downgrade from the CPQ Model Configuration page, administrators can bundle the products with a root model and add a configurable attribute to select one of the offerings.

When a subscription is created, all product lines in the Transaction are added to the same subscription. When there is the root model, users can track the content of the subscription using a single asset key.

Refer to [Demo Product Setup](#) for detailed information.

### **BOM Modeling for Mandatory and Optional Products**

Companies often choose to bundle their subscription products to give more value to the customer and make the selling process convenient for the customer. The bundle generally consists of two categories of products - the ones that are mandatory as part of the bundle and others which are optional.

Users can add, update, or delete optional products independently during the Amend flow. All mandatory parts should be grouped together in a model and expose options only to switch between models. There can be multiple models defined with various flavors of mandatory products. Optional parts should expose option to configure same in configurator layout.

Defining parts as optional in BOM definition table enables Transaction creation with BOM instance with or without optional parts. Optional parts can be added, updated or deleted independently.

### ***BOM Modeling for Amend Flow***

When amending an existing subscription, the Amend flow can create a new subscription in OSS or apply changes to the existing subscription in OSS. The approach is based on the flag defined in BOM and the Configurator.

To enable this functionality, create a new Commerce attribute at the Transaction Line level. Define the attribute as follows:

- Label: updateOldSubscriptionFlag
- Variable Name: oRCL\_sm\_updateOldSubscription
- Type: Single Select Menu. Supported values are: True and False.

In the Oracle\_BomAttrMap table, make an entry as follows for each product in BOM. This configuration ensures the updateOldSubscriptionFlag is set based on the BOM definition and configuration. Sales users can expose the updateOldSubscription flag in the layout by defining Configurator attributes.

- targetType: LINE\_ATTRIBUTE
- targetVariableName: oRCL\_sm\_updateOldSubscription
- SourceType: STATIC\_ENTRY
- Populate BomItemMapVarName
- Populate RootBomMapVarName

## ORACLE CPQ PACKAGE INSTALLATION AND SETUP

Oracle creates implementation packages as a way to distribute elements needed by customers to implement new Oracle CPQ features. As shown in the below table, implementation packages are available to facilitate the installation of the Subscription Management solution.

PACKAGE NAME	DESCRIPTION
Oracle CPQ Subscription Management Package	A granular migration package that works with any BOM (including System Configuration BOM) and contains new Commerce attributes, actions, and rules supporting Subscription Management.
OIC Integration for End to End Flow (CPQ-OSS-Integration-21D)	The integration in OIC for creating, amending, renewing, and terminating subscriptions.
OIC Flow for the Subscription Management Cloud Renew Event	The integration in OIC for creating renewed Transactions in Oracle CPQ.
OIC Usage Rating Flow	The OIC flow for rating usage charges for OSS by invoking the Oracle CPQ Pricing REST API.
Populate Amend Charge Flow	The integration flow that populates charge information from the original subscription into the amended Oracle CPQ Transaction.

### Create OIC Integration

Creating an OIC integration enables Oracle CPQ to connect to back office systems, on premise environments, and other Oracle products in a consistent, enhanced manner. Before installing the pre-built Oracle CPQ packages, administrators must create an OIC integration in the Oracle CPQ Integration Center.

**Note:** OIC is also known as Integration Cloud Service (ICS). When creating the OIC integration in the Integration Center, select Integration Cloud Service as the integration type.

To create an OIC integration, perform the following steps:

1. Open the Admin Home page.
2. Select **Integration Center** under Integration Platform. The Integration Center opens.
3. Click **Create Integration**.
4. Select **Integration Cloud Service** from the **Type** drop-down.
5. Enter **OSSICS** in the Name field. The variable name should be **oSSICS** on both the source and the target site.
6. Enter the discovery URL in the following format: `https://<hostname>/icsapis/v1/integrations`. The `hostname` is the OIC environment name.
7. Enter the username for the OIC environment in the **Username** field.
8. Enter the password for the OIC environment in the **Password** field.
9. Click **Test** to verify the connection.  
If the status returned is “Test Connection Passed”, proceed to step 10.
10. Select the **Enable Integration** check box.
11. Click **Save**.

## Install Subscription Management Package

The Subscription Management package is a granular migration package containing new elements in support of the Subscription Management solution. As a granular migration package, administrators can add or remove specific elements from the package or remove specific elements when importing the package.

To install the Subscription Management package, perform the following steps:

1. Download the Subscription Management package (i.e. CPQ\_OSS\_Package\_21D\_1.zip) from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Open the Admin Home page.
3. Select **Migration** under **Utilities**.
4. Select **Import Package** from the **Select A Mode** drop-down. The Upload Package dialog opens.
5. Click **Browse** and navigate to the Subscription Management package.
6. Click **Upload**.
7. Click **Migrate**.

When the migration completes, check the migration logs for errors.

**Note:** When installing the Subscription Management package for the first time, and the OIC site does not have Subscription Management-related integration installed yet, the migration succeeds and Commerce migration displays a warning message. Refer to [Appendix P: Troubleshooting](#) for more information.

## Define Email Notifications for Subscription Renewals

Administrators can define email notifications to alert subscription owners when a subscription is renewed. For additional information about this email notification, refer to [Appendix D: Renew Subscription Workflow](#).

To define email notifications for subscription renewals, perform the following steps:

1. Open the Admin Home page.
2. Select **Process Definition** under **Commerce and Documents**. The Processes page opens.
3. Select **Steps** from the **Navigation** menu next to the applicable Commerce process.

**Note:** The Oracle Quote to Order Commerce process is included with the Oracle CPQ Ref App. If customers have chosen to overwrite the **Oracle Quote to Order** Commerce process with an alternate Commerce process, select the name of the alternate process.

4. Click **List**.
5. Expand the **Start** step.
6. Expand **Admin**.
7. Expand **Save**.
8. Double click **Save -> In Progress**.
9. Click **Define Notifications** from the bottom right corner. The Add Notification Rule page opens.
10. Select the **Send Email** option.
11. Select the **Advanced Recipient Email Address(es)** option under **Recipient(s)**.
12. Click **Define Function**.
13. Change the email ID on line 4 of the BML and set it as required.

14. Click **Save** and **Close**.

15. Repeat the above steps for the **Start** step and the **Sales User**.

## Subscription Pricing Setup

The Subscription Management package comes with a Subscription Pricing configuration for the sample BOM for the Vision Vehicles service. If you are adding new products, complete the following actions for new products in BOM data:

- Define the price definition for products in Oracle Pricing Data Tables.
- Define tier information, if any, for products in the Pricing Data Table.
- You can re-use the existing Pricing profile, enhance condition to run pricing for products in BOM, or add a new Pricing profile.

**Note:** Refer to [Pricing Engine Setup](#) for detailed information.

## OIC INTEGRATION INSTALLATION AND SETUP

This section contains information about importing the OIC Integrations into your OIC environment and creating web service connections between Oracle CPQ and OSS.

- **OIC Integration for End-to-End Flow (CPQOSSIntegrations\_21D.par):** Contains the integrations in OIC for the Create Subscription, Amend Subscription, Renew Subscription, and Terminate Subscription workflows.
- **OIC Flow for OSS Renew Event:** Contains the integration for creating a new Transaction in Oracle CPQ based on an OSS Renew event.
- **OIC Usage Rating Flow:** Provides the rating usage charges for OSS by invoking the Oracle CPQ Pricing API. Refer to [Appendix O: Calculate Price API](#).

**Note:** For information about the REST APIs used in these integrations, refer to the [OIC Mapping Details](#) section of this installation guide.

### Import Packages and Integrations

Import the OIC integrations into OIC to create integrations between Oracle CPQ and OSS.

To import the OIC package into OIC, perform the following steps:

1. Download the OIC integrations from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Log in to the OIC site as an admin user.
3. Click **Integrations**.
4. Click **Packages**.
5. From the top right corner, click **Import**. The Import Package dialog opens.
6. Select **CPQOSSIntegrations\_21D.par**.
7. Click **Import**.

To import the OIC integrations into OIC, perform the following steps:

1. Download the OIC integrations from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Log in to the OIC site as an admin user.
3. Click **Integrations**.
4. Click **Integrations** again.
5. From the top right corner, click **Import**. The Import Integration dialog opens.
6. Select the downloaded Renew integration.
7. Click **Import**.
8. Repeat the above steps for the Rate Usage integration.

**Note:** If an integration with the same name is already present in the OIC environment, deactivate the integration before replacing it.

## Configure Oracle CPQ and OSS Connections in OIC

Configure connections for the integration(s) imported from [Import Packages and Integrations](#)

**Note:** Record the integrations designated from the [Import Packages and Integrations](#). Each integration provides a connection icon that you can hover over to get connection name.

The OIC CPQ-OSS-Integration-21D Integration creates two connections required for Oracle CPQ and OSS.

**CPQ SOAP Connection** - `https://<CPQ_HOSTNAME>.com/v2_0/receiver/commerce/oraclecpqo?wsdl`

**OSS REST Connection** - `https://<OSS_HOSTNAME>.com/crmRestApi/resources/latest`

The Renew Event Integration creates four connections in OIC. These connections are created with the integration import and require set up of end point and security credentials, as follows:

**OSC225\_CONMGR** (or any other name): Oracle CX Sales Connection. This creates a Renewal Opportunity in the CX Sales CRM as part of Renewal Flow initiated by OSS.

- Interface catalog url:  
`https://<OSC_HOSTNAME>/helpPortalApi/otherResources/latest/interfaceCatalogs`
- Service catalog WSDL url: `https://<OSC_HOSTNAME>/fscmService/ServiceCatalogService?wsdl`
- OSC Events Catalog URL: `https://<OSC_HOSTNAME>/soa-infra`
- Security credential: {username/password of the OSC environment}

**CPQ\_SLC06EIIY\_ABO**: REST Connection.

- Connection Type : REST API URL
- End point Connection url: `https://<CPQ_HOSTNAME>/rest/v13/assets`
- Security cred: {username/password of the CPQ site}

**CPQ\_SLC06EIIY\_TRANSACTION**: REST Connection.

- End point : Connection Type : REST API URL
- Connection url: `https://<CPQ_HOSTNAME>/rest/v13/commerceDocumentsOraclecpqoTransaction`
- Security cred: {username/password of the CPQ site}

**FUSION\_SUB**: REST Connection.

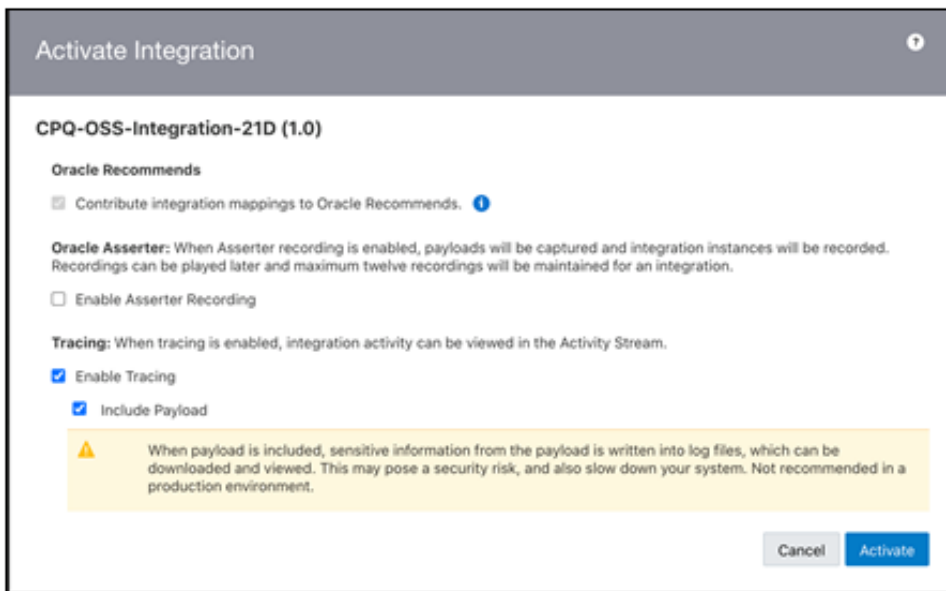
- Connection url: `https://<OSC_HOSTNAME>/crmRestApi/resources/latest`
- Security credential: {username/password of the OSC environment}

To configure the Oracle CPQ and OSS connections in OIC, perform the following steps:

1. Log in to OIC as an admin user.
2. Click **Connections**. Search and open the connection.
3. Click **Configure Connectivity**.
4. Update the connection URL to point to the new host.
5. Click **OK**.
6. Click **Configure Security** to specify the login credentials to access the application.
7. Provide the user credentials.
8. Click **OK**.
9. Click **Test** from the top right corner.
10. Click **Save** when the connection is complete.

11. After the integration and connections are setup, activate the newly imported integration by clicking the active icon corresponding to the integration name. After successfully activating the integration, an activation window displays.

**Tip:** Oracle recommends that the **Enable Tracing** and **Include Payload** checkboxes are selected when activating an integration. Doing so will capture valuable troubleshooting information that may be useful in troubleshooting a failed activation.



Sample Subscription CPQ-OSS-Integration-21D Activation Window

12. Click **Activate**. A message displays at the top of the page after a successful activation.

## Configure Oracle CPQ and OSS Usage Rating Connections in OIC

Configure connections for the integration(s) imported from [Import Integrations](#).

**Note:** Record the integrations designated from the [Import Integrations](#). Each integration provides a connection icon that you can hover over to get connection name.

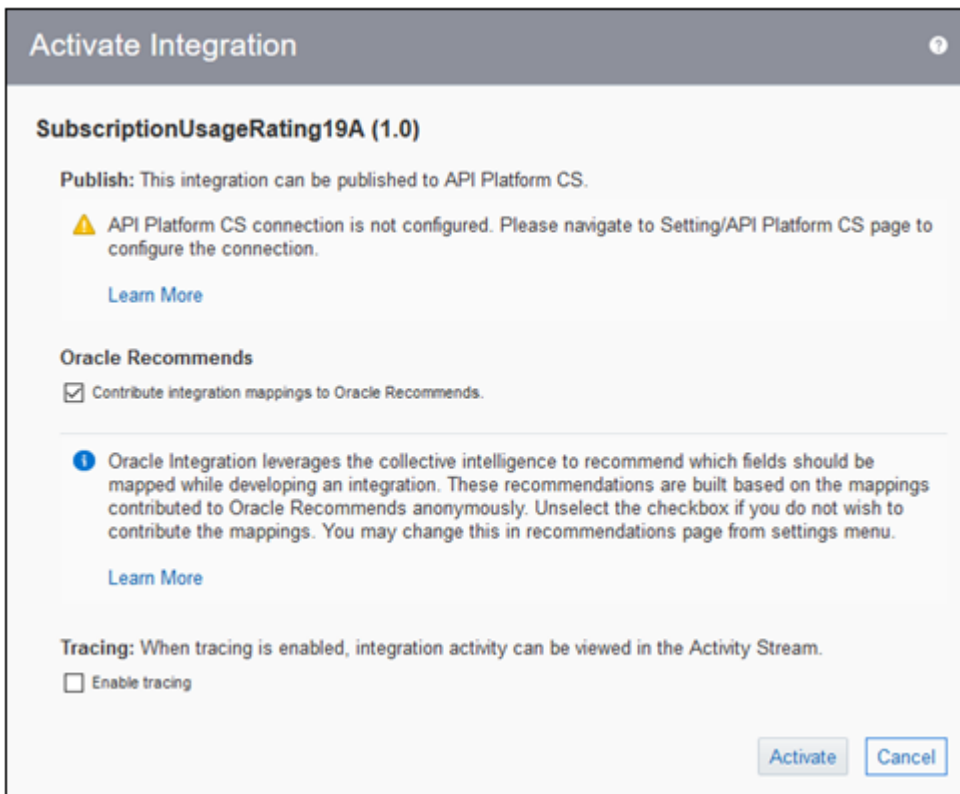
To configure the Oracle CPQ and OSS Usage Rating connections in OIC, perform the following steps:

1. Log in to OIC as an admin user.
2. Click **Connections**. Search and open the connection.
3. Click **Configure Connectivity**.
4. Update the connection URL to point to the new host. For example:  
`https://<OIC_hostname>.integration.ocp.oraclecloud.com:443/ic/api/integration/v1/flows/rest/SUBSCRIPTIONUSAGERATING19A/1.0/price.`
5. Click **OK**.
6. Click **Configure Security** to specify the login credentials to access the application.
7. Provide the user credentials.
8. Click **OK**.
9. Click **Test** from the top right corner.
10. Click **Save** when the connection is complete.



11. After the integration and connections are setup, activate the newly imported integration by clicking the toggle button corresponding to the integration name. After successfully activating the integration, an activation window displays.

**Tip:** Oracle recommends that the **Enable Tracing** and **Include Payload** checkboxes are selected when activating an integration. Doing so will capture valuable troubleshooting information that may be useful in troubleshooting a failed activation.



Sample Usage Rating Activation Window

12. Click **Activate**. A message displays at the top of the page after a successful activation.

**Note:** Starting from Oracle CPQ 21D map quantity field of OSS to CPQ's Usage Value (oRCL\_PRC\_usageValue) for products of type Usage.

## Register Third Party Application for Subscription in Fusion

To register third party application for subscription in Fusion, perform the following steps:

1. Log in to OIC as an admin user.
2. Click **Integrations**.
3. Click **Integrations** again.
4. Click on the **SubscriptionUsageRating** integration name.
5. Click the **Menu** icon in the right-hand portion of the page and select **Primary info**.
6. Copy the Identifier value and make note of the Version (for example, 1.0).
7. Log in to the Fusion application using your FSM setup credentials.
8. Click the **Menu** icon in the top-left corner of the Fusion page.
9. Click **Others**.
10. Click **Setup and Maintenance**.

11. Click the Down arrow in the top-left portion of the page just above the Functional Area list.
12. Select **Sales** from the drop-down.
13. Enter "register" in the **Search Tasks** field and click the **Search** icon to search for the **Register Third Party Applications for Subscription** task.
14. Click **Register Third Party Applications for Subscription** in the popup window. The Register Third Party Applications for Subscription task is highlighted in the Task list.
15. Click on the link for Register Third Party Applications for Subscription. The Register Third Party Applications for Subscription page displays.
16. Click **Add (+)** to add a new row.

Application Type	Application Name	Integration Type	Endpoint URL	Security Policy	User Name	Password
Pricing	CPQ	REST	https://oic.oraclecloud.com/443/ics/integration/v1/flows/rest/SUBSCRIPTIONUSAGERATING19A/price	oraclefusi_username_bksp_00e	username	*****

#### Fusion – Register Third Party Applications for Subscriptions

17. Select **Pricing** as the **Application Type**.
18. Enter **CPQ** as the **Application Name**.
19. Select **REST** in the **Integration Type** drop-down.
20. Enter the endpoint URL into the **Endpoint URL** field.

The endpoint URL follows the following format:

`https://<OIC_HOSTNAME>.integration.ocp.oraclecloud.com:443/ic/api/integration/v1/flows/rest/<IDENTIFIER>/,VERSION>/price`

For example:

`https://<OSS_hostname>.integration.ocp.oraclecloud.com:443/ic/api/integration/v1/flows/rest/SUBSCRIPTIONUSAGERATING19A/1.0/price`

**Note:** The IDENTIFIER is case sensitive.

21. Enter your **User Name**.
22. Enter your **Password**.
23. Click **Save**.

## OIC Lookup Details

### CPQ-OSS-PriceTypeDVM

CPQ-PriceType	OSS-PriceType
One Time	ORA_ONE_TIME
Recurring	ORA_RECURRING
Usage	ORA_RECURRING_USAGE

### CPQ-OSS-PricePeriodicityDVM

CPQ-PricePeriodicity	OSS-PricePeriodicity
Per Month	0zG
Per Year	0zE

### CPQ-OSS-AdjustmentTypeDVM

CPQ-AdjustmentType	OSS-AdjustmentType
Percent Off	ORA_DISCOUNT_PERCENT
Amount Off	ORA_DISCOUNT_AMOUNT
Price Override	ORA_PRICE_OVERRIDE

## OIC Mapping Details

This section contains the OIC mapping details for the OSS payload coming from Oracle CPQ.

### SUBSCRIPTION

OSS PAYLOAD ATTRIBUTE	DATABASE DATA TYPE	MAPPED TO CPQ ATTRIBUTE	COMMENTS
SubscriptionNumber	VARCHAR2(120 CHAR)	transactionID_t	Concatenation is done to keep Subscription number unique each time.
SourceSystem	VARCHAR2(30 CHAR)	Hard Code 'CPQ'	
SourceKey	NUMBER(18,0) VARCHAR2(120 CHAR)	transactionId_t	
BusinessUnitId	NUMBER(18,0)	Buld(BusinessUnitId_t)	One time set up field. The BusinessUnitId is unique for the given site.
subscriptionProfield	NUMBER(18,0)	subscriptionProfileId_t)	
PrimaryPartyId	NUMBER(18,0)	partyId	
Pricing System	VARCHAR2(30 CHAR)	"CPQ"	
Currency	VARCHAR2(15 CHAR)	currency_t	
StartDate	DATE	defaultRequestDate_t	
EndDate	DATE	SubscriptionEndDate_t	
BillingFrequency	VARCHAR2(30 CHAR)	oRCL_billingFrequency_t	
BillToAccountId	NUMBER(18,0)	accountNumber_t	
BillToSiteUseld	NUMBER(18,0)	billToSiteUseld_t	
GenerateBilligSchedule	VARCHAR2	"Y"	"Y" value instructs OSS to generate the bill schedule

## PRODUCTS

This entity is mapped from the Transaction Line when the subscription type is "subscription". Fields are mapped from the Transaction Line. Product is an array in the OSS payload.

FIELDS IN SUBSCRIPTION PAYLOAD	FIELD TYPE	FIELD IN CPQ TRANSACTION LINE	COMMENTS
LineNumber	VARCHAR2(300 CHAR)	_part_id + "-" + _sequence_number	
ProductName	VARCHAR2(300 CHAR)	_part_number	Ensure the CPQ partNumber is the same as the itemNumber in PIM when creating a part in CPQ.
Quantity	NUMBER	requestedQuantity_l	
ItemUnitOfMeasure	VARCHAR2(15 CHAR)	requestedUnitOfMeasure_l	
StartDate	DATE	contractStartDate_l	
EndDate	DATE	contractEndDate_l	
SourceKey	NUMBER(18,0) VARCHAR2(120 CHAR)	transactionID_l	
SourceNumber	NUMBER(18,0) VARCHAR2(120 CHAR)	transactionNumber_t	
SourceLineKey	VARCHAR2(120 CHAR)	_sequence_number	
ExternalAssetKey	VARCHAR2(120 CHAR)	itemInstancel_d_l	
ExternalRootAssetKey	VARCHAR2(120 CHAR)	rootAssetKey_l	
ExternalPriceBookId	VARCHAR2(120 CHAR)	PriceBookId from CPQ	Not mandatory
ExternalPriceBookName	VARCHAR2(30 CHAR)	PriceBookNameFrom CPQ	Not mandatory

## CHARGES

In Oracle CPQ, charge information is part of the CPQ Transaction Line. Subscription pricing populates charge fields.

OSS PAYLOAD FIELD	DATA TYPE	MAPPED TO CPQ TRANSACTION LINE FIELD	COMMENTS
ExternalKey	NUMBER(18,0) VARCHAR2(120 CHAR)	_part_number	Part number from charge line in CPQ.
ExternalParentKey	NUMBER(18,0) VARCHAR2(120 CHAR)	_part_number	Part number from charge line in CPQ.
ChargeName	VARCHAR2(300 CHAR)	_part_number	Part number from charge line in CPQ.
PriceType	VARCHAR2(300 CHAR)	priceType_l	Lookup : CPQ-OSS-PriceTypeDVM
PricePeriodicity	VARCHAR2(30 CHAR)	pricePeriod_l	Lookup: CPQ-OSS-PricePeriodicityDVM
UnitListPrice	NUMBER	listPrice_l	
BlockSize	NUMBER	oRCL_pRC_blockSize	Usually a static value defined as a Line attribute in BOM attribute table.
Allowance	NUMBER	oRCL_pRC_blockAllowance	Usually a static value defined as a Line attribute in BOM attribute table.
TieredFlag	VARCHAR2(3 CHAR)	oRCL_pRC_tierd_l	Y/N
TierType	VARCHAR2(30 CHAR)	oRCL_pRC_tiertype_l	

## CHARGE TIERS

OSS PAYLOAD FIELD	DATA TYPE	SOURCED FROM CALCULATED INFORMATION IN CPQ TRANSACTION LINE	TYPE
Tier Sequence	NUMBER	oRCL_pRC_tierInfo.oRCL_pRC_tierSequence	Integer
TierFrom	NUMBER	oRCL_pRC_tierInfo.oRCL_pRC_tierFrom	Integer
TierTo	NUMBER	oRCL_pRC_tierInfo.oRCL_pRC_tierTo	Integer
ListPrice	NUMBER	oRCL_pRC_tierInfo.oRCL_pRC_tierListPrice	
PriceFormat	VARCHAR2(30 CHAR)	oRCL_pRC_tierInfo.oRCL_pRC_tierPriceFormat	Single Select Menu: 1) PER UNIT [ORA_PER_UNIT] 2) PER BLOCK [ORA_PER_BLOCK]
BlockSize	NUMBER	oRCL_pRC_tierInfo.oRCL_pRC_tierBlockSize	

## CHARGE ADJUSTMENTS

A charge adjustment is an array inside a charge. The source for this entry is the "orcl\_prc\_discounts" Data Table.

OSS PAYLOAD FIELD	DATA TYPE	CPQ LINE LEVEL ATTRIBUTES	TYPE
AdjustmentSequence	NUMBER	1	Integer  As we are supporting only 1 discount per line, the sequence number is defaulted to 1 always.
AdjustmentName	VARCHAR2(300 CHAR)	"CPQ Adjustment"	Text  As we are supporting only 1 discount per line, the adjustment name is defaulted to "CPQ Adjustment" always.
AdjustmentType	VARCHAR2(30 CHAR)	customDiscountType_I	Look Up: CPQ-OSS-AdjustmentTypeDVM
AdjustmentValue	NUMBER	"customDiscountValue_I	Integer
AdjustmentBasis	VARCHAR2(30 CHAR)	"ORA_LIST_PRICE"	We always support discounts on List Price
AdjustmentEffectivity	VARCHAR2(30 CHAR)	discountEffectivityType_I	All Term [ORA_ALL_TERM]. The default implementation supports only All Term. But customer can implement the other options like <ul style="list-style-type: none"> <li>• Periods from Start Date [ORA_PERIODS_FROM_START_DATE]</li> <li>• Periods before End Date [ORA_PERIODS_BEFORE_END_DATE]</li> <li>• Date [ORA_PERIODS_BEFORE_END_DATE]</li> <li>• Specific periods [ORA_SPECIFIC_PERIODS]</li> </ul> Refer <a href="#">Appendix N: Implementation for Discount Effectivity Types</a> .



## ORACLE CPQ FIELD SETUP

Oracle CPQ administrators must setup fields in the CPQ environment and obtain the values for these fields from the OSS administrator. Setting up these fields is a one-time setup step.

### Business Unit ID Field

The Business Unit Id, or organization Id, uniquely identifies a Fusion site. Administrators must setup this field in Oracle CPQ on a per-site basis. For new sites, configure the field in Fusion. The Commerce process has a 'Buld' attribute at the Transaction level. Administrators must obtain the value for this field from the OSS administrator and populate the attribute in the Transaction accordingly.

### Subscription Profile ID Field

The Subscription Profile Id is an OSS attribute that determines the nature of the subscription created. Obtain the Subscription Profile Id from the OSS administrator. Then, populate the "Subscription Profile Id" value in the Transaction of the Commerce process.

### Account Fields

The account information provided during subscription creation is used for subscription billing purposes. With the Subscription Management solution, a sales user can obtain the account information for a customer by entering a customer company name and clicking Customer Details. The fields listed below are populated and are mapped to the associated OSS fields. These field mappings support the creation of a new subscription in OSS based on the information provided in the CPQ Oracle Transaction. The fields should be added to the Customer Details tab of the Transaction layout.

- **Party ID:** Associated with the primary **PartyId** field in OSS.
- **Account Number:** Associated with the **BillToAccountId** field in OSS.
- **Bill To Site Use ID:** Associated with the **BillToSiteNumber** field in OSS.
- **Customer ID:** Associated with the **\_customer\_id** attribute in CPQ for ABO use.

The screenshot displays the Oracle CPQ interface for the 'Customer Details' tab. At the top, there is a 'Transaction' header with several action buttons: Save, Return to Sales Cloud, Update Opportunity, Generate Proposal, Delete Transaction, Customer Rejection, Customer Details (which is the active tab), and Customer Assets. Below the header, there are navigation tabs: Transaction Details, Customer Details (selected), Pricing Details, and Troubleshooting and Support Controls. The main content area is divided into two columns of input fields. The left column contains fields for Customer Company Name 2, Customer Address, Customer Address 2, Customer City, Customer State, Customer Zip, Customer Country, Customer Phone, Customer Fax, and Customer Email. The right column contains fields for Customer Company Name, Customer ID (with value 1002), Account Number (with value 1002), Party Id (with value 1002), and Bill To Site Use Id (with value 1009).

Sample Image of Customer Details Tab

## Billing Frequency – Price Periodicity Field

This is a menu field that provides the Billing Frequency for a subscription. The Billing Frequency can be customized to fit the specific customer environment. The Billing Frequency option can be obtained using the following REST API call:

`https://<OSS-host-name>.us.oracle.com/crmRestApi/resources/latest/timeCodeUnits`

### Menu Attribute Editor (oRCL\_billingFrequency\_t) Document : Oracle Quote to Order > Transaction

**General** | **Default** | **Modify** | **Document Views** | **Mapping**

#### Main Information

\*Label:  x

Variable Name:

Required:

Array Control Attribute:

Description:

Exclude From XML:

Menu Type:

Constrained Menu Option Behavior:

Trigger Auto Update:

JET Layout Path: Unassigned

Mobile Layout Path: Unassigned

#### Menu Population

*Menu Entry:	Displayed Text	Variable Name	
	YEAR [0zE]		<input type="button" value="Add Entry"/>
	MONTH [0zG]		<input type="button" value="Remove"/>
	QUARTER [0zF]		<input type="button" value="Add Entry"/>
	WEEK [0zH]		<input type="button" value="Remove"/>
	DAY [DY]		<input type="button" value="Add Entry"/>

[Back to Top](#)

### Billing Frequency Field

**Note:** The conversion rate multiplied by the BaseUOMCode provides the UserUOMCode, which is populated in the Billing Frequency menu.

## Part Custom Fields

This section contains the part custom fields that CPQ administrators must configure as a one-type setup step.

### Part Field Setup – Enable Subscription Pricing

#### Subscription Type

To help identify charge line and product, set up “\_part\_custom\_field9” as a Product Type configured as shown below. The field is empty for model lines. For subscription product types, enter the value as "subscription".

String Editor		
<b>General</b>		
*Field Name:	Product Type	
*Variable Name:	_part_custom_field9	
Required:	<input type="checkbox"/>	
Default Value:		
<b>Show Field</b>	<b>Pages</b>	<b>Page Specific Label</b>
<input type="checkbox"/>	Search Page	
<input type="checkbox"/>	Search Results Page	
<input type="checkbox"/>	BOM Page	
<input type="checkbox"/>	Detail Page	
<b>Search Behavior</b>		
Leading Wildcard	Not Allowed	
Trailing Wildcard	Not Allowed	

#### Product Type Field Setup

**Note:** Use \_part\_custom\_field9 to identify a product. Administrators must mark any subscription products as 'subscription' in \_part\_custom\_field9. Subscription products will have value "subscription" for this field.

Part Editor													
<b>Product Information</b>													
*Part Number:	Crossover Outright Buy												
Part Display Number:	Crossover Outright Buy												
Direct Buy:	Direct Buy												
Description:	Crossover Outright Buy												
Extended Description 1:	<input type="text"/> <input type="button" value="Edit"/>												
Extended Description 2:	<input type="text"/> <input type="button" value="Edit"/>												
Units:	<input type="text"/>												
Lead Time(Days):	<input type="text"/>												
Company Associations:	<table border="0"> <tr> <td>Available:</td> <td>mumba54485 (mumba54485)</td> <td>Selected:</td> <td></td> </tr> <tr> <td></td> <td><input type="text"/></td> <td><input type="button" value="➤"/></td> <td><input type="text"/></td> </tr> <tr> <td></td> <td></td> <td><input type="button" value="➤"/></td> <td></td> </tr> </table>	Available:	mumba54485 (mumba54485)	Selected:			<input type="text"/>	<input type="button" value="➤"/>	<input type="text"/>			<input type="button" value="➤"/>	
Available:	mumba54485 (mumba54485)	Selected:											
	<input type="text"/>	<input type="button" value="➤"/>	<input type="text"/>										
		<input type="button" value="➤"/>											
<b>Pricing Information</b>													
	EUR: <input type="text"/>												
	GBP: <input type="text"/>												
	USD: <input type="text"/>												
<b>Extended Information</b>													
Product Group:	<input type="text"/>												
Price Period:	<input type="text"/>												
Price Type:	<input type="text"/>												
Max discount %:	<input type="text"/>												
Cost:	0.0												
Product Type:	subscription												

#### Part Custom Field Setup

## Layout Fields

This section contains the attributes and actions that need to be added in the Layout. Refer to [Appendix F: Commerce Attributes](#).

### **Transaction Level Attributes**

The following attributes are required for troubleshooting purposes only.

- Subscription Id
- Subscription Status
- Subscription Profile Id
- RenewDraftSubscriptionNumber

### **Transaction Level Actions**

The following Actions are required to be added in the Layout:

- Submit Order

**Note:** The attributes and actions defined as [Account Fields](#) must be added to the Transaction Level – Customer Detail.

### **Transaction Line Attributes**

The following attributes need to be added to the Line Item Grid in the Layout:

- fulfillment status
- ActionCode
- InstanceId
- RootAssetKey
- Change Reason
- Change Code
- Amend Replacement
- Update Old Subscription
- Tier Type
- Tiered
- Periodicity
- Quantity (requestedQuantity\_I)
- Contract Periods
- Contract Start Date
- Contract End Date
- Discount (customDiscountValue\_I)
- Discount Type (customDiscountValue\_I)
- Custom Discount Amount(customDiscountAmount\_I)
- Net Price
- Net Amount
- Contract Value (List)
- Contract Discount
- Contract Value (Net)

**Note:** The fulfillment status and ActionCode attributes are already in the Line Item Grid if the site has been configured for ABO. If they are not already in the Line Item Grid, they should be added.

### Edit Transaction Line Attributes

The following attributes needs to be added to the Line Item Layout.

- Amend Replacement
- Tier Array
- Block Size
- Block Allowance
- Unit List Price

## DEMO PRODUCT SETUP

Oracle provides a Vision Vehicles SUV demo product, which is a sample Configuration in Oracle CPQ. Use the Vision Vehicles SUV demo product to understand the functionality available in the Subscription Management solution. To understand the various Subscription Management flows supported, refer to [Appendix A: Create Subscription Workflow](#), [Appendix B: Amend Subscription Workflow](#), [Appendix C: Add Amended Lines to Existing Subscription Workflow](#), [Appendix D: Renew Subscription Workflow](#), and [Appendix E: Terminate Subscription Workflow](#).

PACKAGE NAME	FILE NAME	DESCRIPTION
Demo_Product_Vision_Vehicles	Demo_Product_Vision_Vehicles_1.zip	The data used for the Vision Vehicles SUV demo product.
BOM Data	datatable_ItemDef.zip datatable_ItemMap.zip datatable_AttrMap.zip datatable_AttrDef.zip	The BOM data used for Data Tables.
Parts	Part1.zip	The parts added for the Vision Vehicles demo product with product type as subscription: <ul style="list-style-type: none"> <li>• VEHICLE               <ul style="list-style-type: none"> <li>○ Crossover SUV Subscription</li> <li>○ Sports SUV Subscription</li> <li>○ Luxury SUV Subscription</li> <li>○ Crossover Outright Buy</li> <li>○ SUV Charging Station use</li> <li>○ Sirius XM Radio</li> <li>○ WiFi Service</li> <li>○ Standard Maintenance Plan</li> <li>○ Premium Maintenance Plan</li> </ul> </li> </ul>

**Note:** The packages shown in the above table are optional packages. Administrators must install the packages to make the Vision Vehicles SUV demo product visible in Oracle CPQ.

## Install the BOM Data Table Packages

The BOM Data Table packages adds the demo product data into Oracle BOM tables, such as the Oracle\_BomItemDef, Oracle\_BomItemMap, and Oracle\_BomAttrMap BOM data tables. When the BOM Data Tables packages are installed, the BOM Configuration for Vision Vehicles SUV model is added to the sample Configuration.

To install BOM Data Tables package, perform the following steps:

1. Download the BOM Data Table packages (i.e. datatable\_ItemDef.zip, datatable\_ItemMap.zip, datatable\_AttrMap.zip) from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Open the Admin Home page.
3. Select **Data Tables** under **Developer Tools**. The Data Tables page opens.
4. Click **File** and select **Import** from the **Options** drop-down.
5. Click **Browse**.
6. Select the downloaded zip file **datatable\_ItemDef.zip**.
7. Select **Oracle BOM Tables** as the Destination Folder. Create the folder if it is not present.
8. Click **Import**.
9. Repeat steps 4 through 8 for each of the BOM data table files (i.e. **datatable\_ItemMap.zip** and **datatable\_AttrMap.zip**).
10. Click **Menu** and select **Status Log** to check the status of the uploaded files.
11. Check the log corresponding to the uploaded files for errors.
12. Double-click on each data table (**Oracle\_BomItemDef**, **Oracle\_BomItemMap**, and **Oracle\_BomAttrMap**) from the Navigation and click **Schema** tab.
13. Select index and key columns for VariableName field.
14. Save and Deploy.

## Install the Parts Package

The Parts package adds parts for the Vision Vehicles SUV demo product. When the Parts package is installed, the parts are added to the sample Configuration based on the selections. To view an example of these services in the CPQ sample Configuration, refer to [Appendix A: Create Subscription Workflow](#), [Appendix B: Amend Subscription Workflow](#), [Appendix D: Renew Subscription Workflow](#), and [Appendix E: Terminate Subscription Workflow](#).

To install the Parts package, perform the following steps:


1. Download the Parts package (i.e. Part1.zip) from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Open the Admin Home page.
3. Select **Upload** under **Utilities**. The Upload Files List page opens.
4. Click **Browse**.
5. Select the downloaded zip file.
6. Click **Add**.
7. Click **Upload**.
8. Click **Refresh** to check the status of the uploaded file.
9. Check the log corresponding to the uploaded file for errors.

## Install the Vision Vehicles SUV Demo Product Package

The BOM Package contains the product data used for the Vison Vehicles SUV demo product.

To install the Package, perform the following steps:

1. Download the package (i.e. Demo\_Product\_Vision\_Vehicles\_1.zip) from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Open the Admin Home page.
3. Select **Migration** under **Utilities**.
4. Select **Import Package** from the **Select A Mode** drop-down. The Upload Package dialog opens.
5. Click **Browse** and navigate to the demo package.
6. Click **Upload**.
7. Click **Migrate**. When the migration completes, check the migration logs for errors.
8. Navigate to the Admin Home page.
9. Select **Catalog Definition** under **Products**. The Supported Products page opens.
10. Click **List**. The Supported Product Families page opens.
11. Click **Add**.
12. Select **Vision Vehicles**.
13. Click **Save**.
14. Open the Admin Home page.
15. Select **Catalog Definition** under **Products**. The Supported Products page opens.
16. Click **List**. The Supported Product Families page opens.
17. Click **List** next to **Vision Vehicles**. The Product Line Administration List page opens.
18. Select **Models** from the **Navigation** drop-down next to **Consumer Vehicles**.
19. Click **List**. The Model Administration List page opens.
20. Select **BOM Mapping** from the **Navigation** drop-down next to **SUV**.
21. Click **List**. The BOM Mapping: Rules List page opens.
22. Verify that the **Default** BOM Rule exists.



The screenshot shows the 'BOM Mapping: Rules List' page. The breadcrumb trail is 'Model : Vision Vehicles > Consumer Vehicles > SUV'. The table has columns for 'Select', 'Order', 'Name (Variable Name)', 'Status', and 'Overall Status'. There is one row with a checkbox, the number '1' in the order field, the name 'Default (Default)', a dropdown menu set to 'Active', and the overall status 'Active'. At the bottom right, there are buttons for 'Translations', 'Add', 'Update', 'Delete', and 'Back'. A 'Back to Top' link is also present.

Select	Order	Name (Variable Name)	Status	Overall Status
<input type="checkbox"/>	1	Default (Default)	Active	Active

Default BOM Rule

## Verify the Addition of All BOM Parts

To verify that all of the BOM parts are added to the Oracle CPQ site, perform the following steps:

1. Open the Admin Home page.
2. Select **BOM** under **Products**. The BOM Administration Platform opens.
3. Select **BOM Root Item List** under **BOM Products**. The BOM Root Items Administration List page opens.
4. Click on the variable names to verify that parts were added. The BOM Item Tree Administration page opens. Missing parts are shown in red on this page.

**Note:** If parts are missing, follow the steps in the Troubleshooting section of this installation guide to manually add the missing parts to your Oracle CPQ site.

## Deploy the Home Page

Deploy the Oracle CPQ Home page to make the changes applied by the Subscription Management package available on the Oracle CPQ Home page.

To deploy the Home page, perform the following steps:

1. Open the Admin Home page.
2. Select **Home Page** under **Styles and Templates**. The Home Page Setup page opens.
3. Verify that the Product Family definition is accurate in the Home page:  
Expand **Vision Vehicles** under Catalog.  
Select the **Model Punch-in** icon next to **Consumer Vehicles**. The Model Punch-ins List opens.  
Make sure **SUV** exists in the Model Punch-ins List. If it does not exist, add it.
4. Click **Deployment Center** from the Home Page Setup page. The Deployment Center opens.
5. Click **Deploy**.
6. Click **Refresh** to verify the successful deployment of the Home page.

**Note:** Be sure to register the demo BOM in the ABO table (oracle\_abopart2Model) with the following:

- Partnumber: VEHICLE
- ProductLine: consumerVehicles
- Segment: visionVehicles
- Model: SUV



## INSTALLED ORACLE CPQ ELEMENTS

Installing the Oracle CPQ Subscription Management package simplifies the implementation of the Subscription Management solution by adding the below elements to the Oracle CPQ Ref App.

- Commerce Attributes
- Commerce Actions
- Library Functions
- Validation Rules
- Hiding Rules
- Steps and Step Transitions
- Timer Configuration

### Commerce Attributes

The installation of the Oracle CPQ Subscription Management package adds several attributes to the Commerce process and modifies several existing attributes included with the Oracle CPQ Release 18B or later Ref App. For a complete list of the Commerce attributes used by the Subscription Management solution, refer to [Appendix F: Commerce Attributes](#).

### Commerce Actions

The installation of the Oracle CPQ Subscription Management package adds or updates the following actions to the Commerce process:

- Submit Order
- Update Asset Timer
- Customer Details
- Open Transaction Line
- Transaction - Save
- Transaction Line - Save

**Note:** The Oracle Quote to Order Commerce process is included with the Oracle CPQ Ref App. If customers have chosen to overwrite the Oracle Quote to Order Commerce process with an alternate Commerce process, the Oracle CPQ package adds the actions to the alternate process.

### SUBMIT ORDER

The Submit Order action is associated with the Submit Order button on the Transaction page. The action is used to create a subscription in OSS.

#### **Create Subscription Integration of Type Integration Cloud Service**

Administrators must create an Integration Cloud Service type integration that invokes OIC to initiate the Create Subscription workflow.

To create the integration, perform the following steps:

1. Open the Admin Home page.
2. Select **Process Definition** under **Commerce and Documents**. The Processes page opens.
3. Locate the Commerce process associated with the Subscription Management integration.
4. Select **Integrations** from the associated Navigation menu.
5. Click **List**. The Integrations page opens.

6. Click **Add**. The Select Integration Types page opens.
7. Select the **Integration Cloud Service** option.
8. Click **Next**. The Edit Integration page opens.
9. Populate the following fields:
  - Name: Create Subscription
  - Variable Name: createSubscription
  - Description: Enter an optional description.
  - Timeout: Enter a value between 0 and 600,000 milliseconds.
  - Action: Select the Import option
  - Services: Select CPQ-OSS-Integration-21D
10. Click **Add**.

**Note:** The Timeout value is used when calling the CPQ-OSS-Integration-21D services. If the service does not respond within the specified time, Oracle CPQ aborts the web service call and invokes an error.

### Create Subscription Integration

An image of the Configuration for the Create Subscription integration is provided below:

The screenshot shows the 'Edit Integration' configuration page. The 'Integration Information' section includes the following fields:

- Type: Integration Cloud Service
- \*Name: Create Subscription
- \*Variable Name: createSubscription
- Description: Create Subscription
- Timeout: 600000 milliseconds
- Action:  Import  Export
- \*Services: CPQ-OSS-Integration-21D
- Transaction ID: (empty field)

At the bottom right, there are links for 'Preview SOAP XML' and 'Preview Middleware Response'.

Configuration for Create Subscription Integration

Once the Create Subscription integration is created, the integration is added to the Submit Order action's integration tab.

The screenshot shows the 'Admin Action' configuration page for '(submitOrder\_t)'. The 'Integration' tab is selected, showing the following integration list:

- OSC - Opp Import (Import)
- OSC - Delete All Revenue Items (Import)
- OSC - Create Revenue Items (Export)
- OSC - Upsert Quote (Export)
- Get OSS Charge (Integration Cloud Service)
- test (BML)
- Rich (BML)

The 'Selected Integration' section shows 'Create Subscription (Integration Cloud Service)' selected. Below the list, there are buttons for 'Translations', 'Apply', 'Update', 'Update and New', and 'Back'. A 'Back to Top' link is also present.

Create Subscription Integration in Submit Order Integration Tab

## UPDATE ASSET TIMER

The Update Asset Timer action is used to configure a Timer to create an asset on a requested date. The BML used by the Update Asset Timer action is included for reference in [Appendix G: Update Asset Timer BML](#). The BML is associated with the Advanced Modify – Before Formula function.

An image of the BML Configuration for the Update Asset Timer is provided below.

BML Editor		
Action : Oracle Quote to Order > Transaction > Update Asset Timer		
<div style="display: flex; border-bottom: 1px solid black;"> <span style="border: 1px solid black; padding: 2px;">BML</span> <span style="border: 1px solid black; padding: 2px; margin-left: 5px;">Debugger</span> </div>		
System Variable Name	Type	Description
<code>system_buyside_id</code>	String	Current Unique Buy-Side ID
<code>system_selected_document_number</code>	String	Document Number of selected line item
Variable Name for (Transaction)	Type	Description
<code>currency_f</code>	String	Currency
<code>paymentTerms_f</code>	String	Payment Terms
<code>transaction_customer_id</code>	String	Customer Id
<code>maxRequestDate_f</code>	String (date)	Max Request Date
Variable Name for (Transaction Line)	Type	Description
<code>transactionLine</code>	Collection of Sub Documents	
<code>document_number</code>	String	Document Number
<code>contractStartDate_f</code>	String (date)	Contract Start Date
<code>itemInstanceid_f</code>	String	Instance ID
<code>fulfillmentStatus_f</code>	String	Fulfillment Status
<code>requestDate_f</code>	String (date)	Request Date
<code>line_bom_parent_id</code>	String	Parent Line Item BOM ID
<code>oRCL_ABO_ActionCode_f</code>	String	Action Code
Imported Util Functions		
<code>Json_ORCL_ABO_abo_updateAsset(Json.bn_json,JsonArray.lines_json_array)</code>		

### Update Asset Timer BML Configuration

## CUSTOMER DETAILS

The Customer Details action is used to support account integration. When users click the Customer Details tab on the Transaction page, a Customer Company Name field is available. By entering a customer company name and clicking Customer Details, the account fields (Party ID, Account Number, and Bill To Site ID) are populated and mapped to the associated OSS fields. These field mappings support the creation of a new subscription in OSS based on the information provided in the Oracle CPQ Transaction.

### Customer Details BML

The BML associated with the Customer Details action is included for reference in [Appendix H: Customer Details BML](#). The BML is associated with the Define Advanced Modify - After Formulas function.

### BML Configuration for Customer Details

An image of the BML Configuration is provided below.

The screenshot displays the BML Editor interface for the action "Oracle Quote to Order > Transaction > Customer Details". It includes a "System Variable Name" table, a list of "Imported Commerce Functions", and a code editor with a BML script. A "Functions" dropdown menu is open, showing a list of functions like "atoi", "atof", "decodebase64", etc.

System Variable Name	Type	Description
Variable Name for (Transaction)	Type	Description
customerCompanyName_t	String	Customer Company Name

**Imported Commerce Functions**

- String getTemplateLocation(String system, String operation)
- String getPassword(String sysName)
- String Dictionary getUserAttributes(String system)
- String invokeWebService(String system, String soapReq)

Expected return type - **String**

Operators: +, -, \*, /, (, ), =, <>, <, >, <=, >=, ==, MOD, AND, NOT, OR

```
121 {
122   PartySiteNumberXmlFragment = get(outputPartySiteNumber, xpathPart
123
124   xpath3 = string[1];
125   xpath3[0] = "//nsl:PartySiteNumber";
126   output3 = readxmlsingle(PartySiteNumberXmlFragment, xpath3);
127   if (containsKey(output3, xpath3[0]))
128   {
129     partySiteNumber = get(output3, xpath3[0]);
130   }
```

Functions: All Functions, atoi, atof, decodebase64, encodebase64, endswith, formatascurrency, find, getcurrencyvalue, Add

## OPEN TRANSACTION LINE

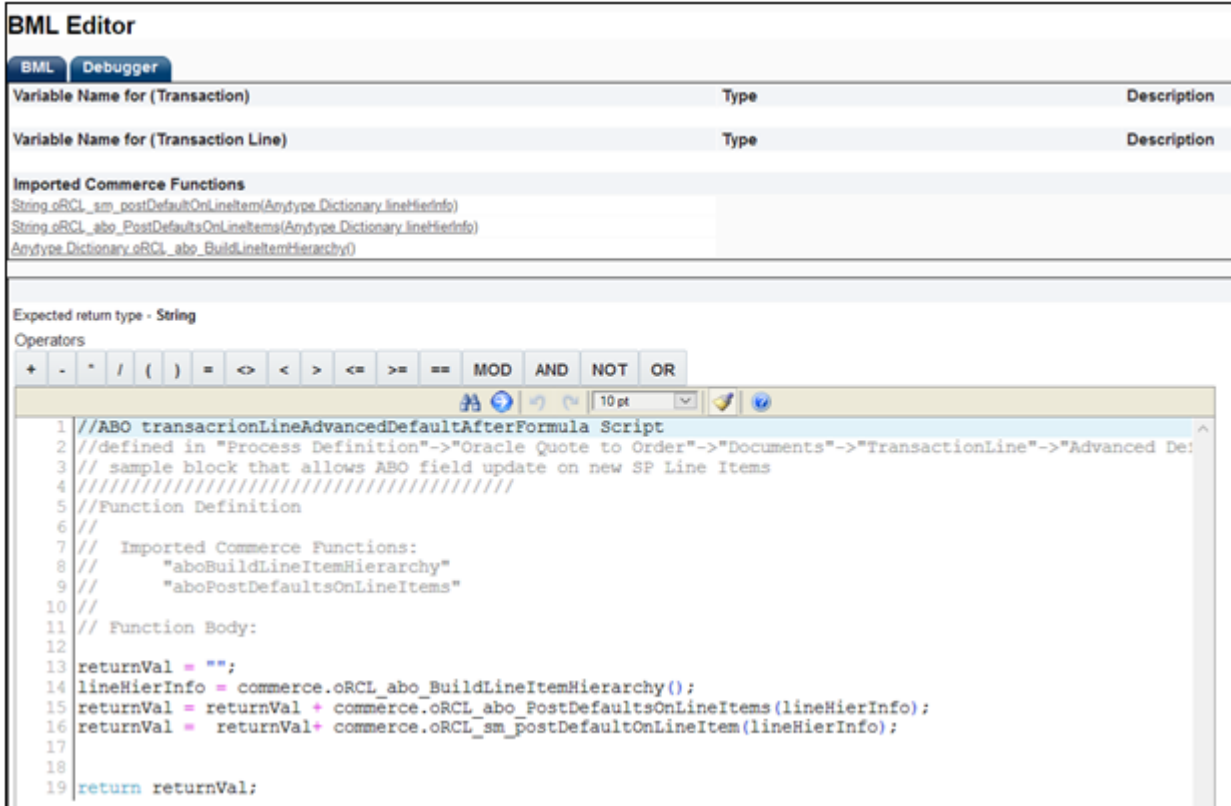
The Open Transaction Line action is used to display the details for a specific Transaction Line such as product, charge information, and charge tiers.

### Open Transaction Line BML

The BML for the Open Transaction Line action is associated with the Define Advanced Modify - Before Formulas function. The BML is included for reference in [Appendix I: Open Transaction Line BML](#).

### BML Configuration for Open Transaction Line

An image of the BML Configuration is provided below.



The screenshot shows the BML Editor interface. At the top, there are tabs for 'BML' and 'Debugger'. Below the tabs, there are two tables with headers: 'Variable Name for (Transaction)' and 'Variable Name for (Transaction Line)', both with columns for 'Type' and 'Description'. Underneath these tables, there is a section for 'Imported Commerce Functions' listing three functions: 'commerce.oRCL\_sm\_postDefaultOnLineItem', 'commerce.oRCL\_abo\_PostDefaultsOnLineItems', and 'commerce.oRCL\_abo\_BuildLineItemHierarchy'. The main area of the editor shows the 'Expected return type - String' and a toolbar with various operators and symbols. The script content is as follows:

```
1 //ABO transactionLineAdvancedDefaultAfterFormula Script
2 //defined in "Process Definition"->"Oracle Quote to Order"->"Documents"->"TransactionLine"->"Advanced De:
3 // sample block that allows ABO field update on new SP Line Items
4 ///////////////////////////////////////////////////////////////////
5 //Function Definition
6 //
7 // Imported Commerce Functions:
8 //     "aboBuildLineItemHierarchy"
9 //     "aboPostDefaultsOnLineItems"
10 //
11 // Function Body:
12
13 returnVal = "";
14 lineHierInfo = commerce.oRCL_abo_BuildLineItemHierarchy();
15 returnVal = returnVal + commerce.oRCL_abo_PostDefaultsOnLineItems(lineHierInfo);
16 returnVal = returnVal + commerce.oRCL_sm_postDefaultOnLineItem(lineHierInfo);
17
18
19 return returnVal;
```

## Transaction - SAVE

The Save action saves the current state of the Transaction.

### Save BML

The BML for the Save action is associated with the Define Advanced Modify – Before Formulas and Define Advanced Modify – After Formulas functions. The BML is included for reference in [Appendix J: Save BML](#).

### BML Configuration for Save

An image of the BML Configuration for Define Advanced Modify – Before Formulas is provided below.

The screenshot shows the BML Editor interface. At the top, it says "Action : Oracle Quote to Order > Transaction > Save". There are two tabs: "BML" and "Debugger".

Variable Name for (Transaction Line)	Type	Description
transactionLine	Collection of Sub Documents	
_document_number	String	Document Number
contractStartDate_1	String (date)	Contract Start Date
pricePeriod_1	String	Period
priceType_1	String	Price Type
contractEndDate_1	String (date)	Contract End Date

**Imported Commerce Functions**  
Integer oRCL\_sm\_calculateContractPeriods(String contractStartDate, String contractEndDate, String priceType, String periodicity)

Expected return type - String [Editor Help](#) [Return/Input Values Help](#)

Operators: +, -, \*, /, (, ), =, <>, <, >, <=, >=, ==, MOD, AND, NOT, OR

```
1 contractPeriodsStr = stringBuilder();
2 for line in transactionLine {
3   sbappend(contractPeriodsStr, line._document_number, "~contractedPeric
4   string(commerce.oRCL_sm_calculateContractPeriods(line.contractStartDa
5 }
6 return sbtoString(contractPeriodsStr);
```

Functions: All Functions, atoi, atof, decodebase64, encodebase64, endswith, formatascurrency, find, getcurrencyvalue, Add

Save BML Configuration

An image of the BML Configuration for Define Advanced Modify – After Formulas is provided below.

The screenshot shows the BML Editor interface. At the top, there are tabs for 'BML' and 'Debugger'. Below the tabs is a table with columns 'Variable Name for (Transaction Line)', 'Type', and 'Description'. Underneath the table is a section for 'Imported Commerce Functions' with the text 'String calculateRollupRevenues()'. The main area is a code editor with a toolbar and a status bar. The code editor contains the following text:

```
1 return commerce.calculateRollupRevenues();  
2
```

On the right side, there is a 'Functions' dropdown menu with a list of functions: 'atoi', 'atof', 'decodebase64', 'encodebase64', 'endswith', 'formatascurrency', 'find', and 'getcurrencyvalue'. Below the list is an 'Add' button. The status bar at the bottom shows 'Position: Ln 1, Ch 1' and 'Total: Ln 1, Ch 42'.

### Transaction Line - SAVE

The Save action saves the current state of the Transaction Line.

### Save BML

The BML for the Save action is associated with the Define Advanced Modify – Before Formulas and Define Advanced Modify – After Formulas functions. The BML is included for reference in [Appendix J: Save BML](#).

### BML Configuration for Save

An image of the BML Configuration for Define Advanced Modify – After Formulas is provided below.

# BML Editor

BML Debugger

System Variable Name	Type	Description
<code>_system_current_document_number</code>	String	Current Document Number

Variable Name for (Transaction Line)	Type	Description
<code>transactionLine</code>	Collection of Sub Documents	
<code>  _document_number</code>	String	Document Number
<code>  _price_calculation_info</code>	String	Calculation Information
<code>  oRCL_pRC_tierd_l</code>	String	Tiered

Imported Util Functions
<code>String _SM.oRCL_pRC_populateCharges(JsonArray charges, String documentNumber)</code>

Expected return type - String

Operators

+ - \* / ( ) = <> < > <= >= == MOD AND NOT OR

```
1 //At TransactionLine level save
2 //System Variable Name Type Description
3 //_system_current_document_number String Current Document Number
4
5 //Variable Name for (Transaction Line) Type Description
6 //transactionLine Collection of Sub Documents
7 //  _document_number String Document Number
8 //  _price_calculation_info String Calculation Information
9 //  oRCL_pRC_tierd_l String Tiered
10
11 //Imported Util Functions
12 //String _SM.oRCL_pRC_populateCharges(JsonArray charges, String documentNumber)
13
14 returnString = "";
15 for line in transactionLine {
16     if(line._document_number == _system_current_document_number){
17         if(line.oRCL_pRC_tierd_l == "N" AND line._price_calculation_info <> "") {
18             calcInfo = jsonarrayget(jsonarray(line._price_calculation_info),0,"json");
19             charges = jsonget(calcInfo,"charges", "jsonArray");
20             returnString = returnString + util._SM.oRCL_pRC_populateCharges(charges, line._document_number);
21         }
22     }
23 }
24 return returnString;
```



## Library Functions

The installation of the Oracle CPQ Subscription Management package adds several library functions to the Commerce process. To view the BML associated with these library functions, refer to [Appendix L: Library Function BML](#).

### String PostDefaultOnLineItemSM

PostDefaultOnLineItemSM is used for querying and populating charges and discount fields during the creation of a Transaction Line. PostDefaultOnLineItemSM is called from Advanced Default - After Formulas. The Return Type, input information, and attributes used by this library function are shown below.

**Commerce BML Library Function Editor: Properties & Parameters**

Name: PostDefaultOnLineItemSM # Parameter Name Parameter Type  
 Variable Name: oRCL\_sm\_postDefaultOnLineItem 1 lineHierInfo Anytype Dictionary  
 Description: This BML is used for populating root asset Key, auto adjustments and charges for lines.  
 Return Type: String

**Function Editor**

Hide Tools | Editor Help

Attributes | Function Wizard | Debugger | Library Function(s)

**Main Document Attribute**

#	Attribute

**Sub Document Attribute**

transactionLine

#	Attribute
1	_part_number
2	_document_number
3	_price_calculation_info
4	itemInstanceId_I
5	rootAssetKey I

**System Attribute**

#	Attribute

## Validation Rules

Validation Rules are used to validate attribute or field values. They are linked to an action and only run when a specific action is clicked by the user. When the Oracle CPQ Subscription Management package is installed, the Validation Rules in the below table are added to the Commerce process.

VALIDATION RULE NAME	LEVEL	DESCRIPTION
Change Reason and Code Population	Transaction Line	Validates that the Change Reason and Change Code fields are populated on the CPQ Transaction page during an Amend flow.
Empty Billing Frequency	Transaction	Validates that the Billing Frequency field is not empty.

## Hiding Rules

Hiding Rules tell Oracle CPQ to hide select attributes when a pre-defined condition is satisfied. They are made up of a condition and an action. The values of the attributes selected as the condition attributes determine the result of the condition, which when True trigger the hiding of the action attributes.

This section identifies the Hiding Rules included in the Oracle CPQ Subscription Management package. The Hiding Rules apply to the Transaction Line level.

### Amend Replacement Visibility

If one or more of the following conditions are satisfied, Amend Replacement is hidden:

- Change Reason is empty or Term Change [ORA\_TERM\_CHANGE] or Quantity Change or Non-Compliant
- Change Code is Empty
- Action code is not ADD[ADD]

### Condition Summary

Change Reason Equals "" OR "Term Change [ORA\_TERM\_CHANGE]" OR "Non-Compliant [ORA\_NON\_COMPLIANCE]" OR "Breach [ORA\_BREACH]" OR "Quantity Change [ORA\_CHANGE\_QUANTITY]" OR (Change Code Equals "") OR (Action Code Not Equals "Add [ADD]"

### Usage Value Visibility

Usage Value and Usage Value UoM attributes are hidden if below condition is satisfied:

- Price Type is not empty or Usage [Usage]

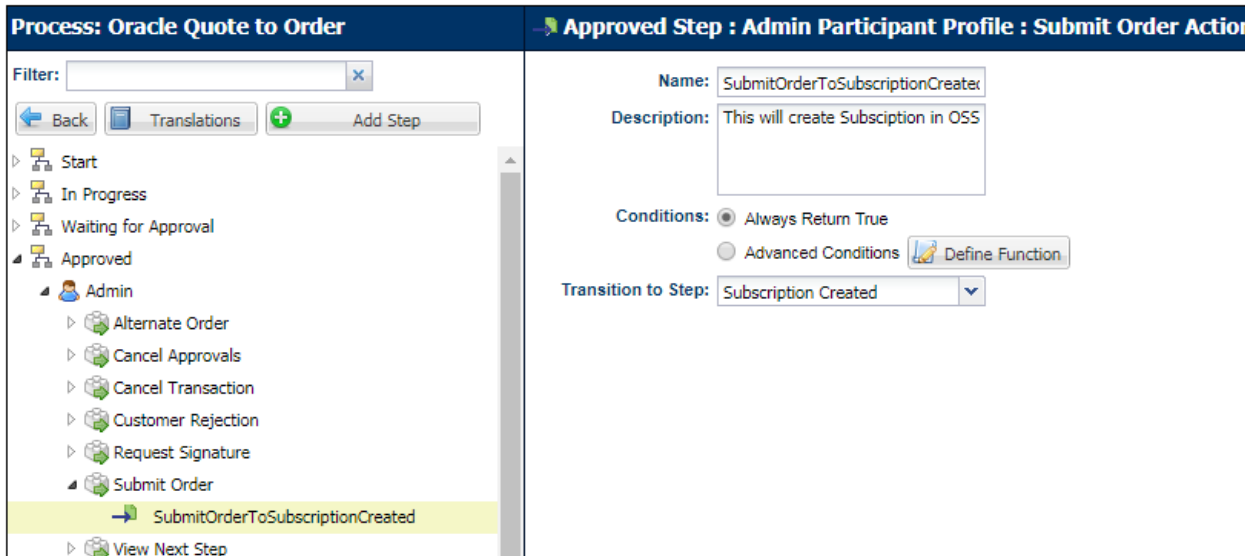
### Condition Summary

Price Type Not Equals "" OR "Usage [Usage]"

## Workflow Steps and Step Transitions

Workflows are made up of steps that define the states of a Transaction and step transitions that designate when the next step is triggered. The Oracle CPQ package adds a new step and step transition to the Commerce process.

- **Subscription Created:** A new workflow step added to the Oracle CPQ Ref App by the Oracle CPQ Subscription Management package. A Timer for creating an asset is configured within this step.
- **SubmitOrderToSubscriptionCreated:** The action variable name of a new step transition displayed in the Approved step used to transition the sales user to the Subscription Created step.



SubmitOrderToSubscriptionCreated Step Transition

## Timer Configuration

Administrators can schedule Oracle CPQ to automatically perform Commerce actions using Timers set up on Modify type Commerce actions. Based on a defined duration of time, a Timer triggers when the elapsed time exceeds the specified duration. Timers are managed within individual workflow steps, and administrators can only associate one Timer with a workflow step.

When using the Subscription Management solution, the Subscription Created workflow step triggers an Update Asset Timer when the Max Request Date is the current date. The Timer creates an asset in the asset database one minute after the Subscription Created step is reached.

The below image shows the Configuration of the Timer. For additional information, refer to [Appendix G: Update Asset Timer BML](#).

The screenshot shows the configuration interface for a timer within the 'Subscription Created Step'. The interface is divided into several sections:

- Name:** Subscription Created
- Variable Name:** subscriptionCreated
- Description:** (Empty text area)
- Forward Rule:** Three radio buttons are present:  No Forwarding Rule,  All Groups, and  Advanced Forwarding Rule.
- Remove Timer:** A button with a red 'X' icon.
- Timer Section:**
  - Name:** createAssets
  - Variable Name:** createAssets
  - Description:** (Empty text area)
  - Map To Profile:** Sales User Profile (dropdown)
  - Map To User:** superuser [Super User] (dropdown)
  - Elapsed Time:** 0 days, 0 hours, 1 minutes (input fields with dropdown arrows)
  - Relative to Date Attribute:** Transaction : Max Request Date [maxRequestDat] (dropdown)
  - Document:** Transaction (dropdown)
  - Action:** Update Asset Timer (dropdown)
  - Action Rule:**  Simple,  Advanced, and a **Define Function** button.

At the bottom right of the form, there are **Save** and **Close** buttons.

Timer Configuration

## ORACLE CPQ ACCOUNT INTEGRATION

This section identifies the library functions that support account integration and the manual changes that administrators must make to the INT\_SYSTEM\_DETAILS Data Table and the INT\_SYSTEM\_TEMPLATES Data Table to support account integration.

Account Integration is not required for customer setup where CPQ is integrated to a CRM system and account information is imported on a transaction as part of the quote/transaction creation from the CRM.

**Note:** For more details about Account Integration with Oracle Customer Data Management (CDM), refer to CPQ-CDM Integration Whitepaper on [My Oracle Support](#) under CPQ to Fusion Financials Integration ([Doc ID 2012010.1](#)).

### Library Functions

The library functions within this section support the CDM integration by retrieving account details.

#### **String** *getTemplateLocation(String system, String operation)*

Queries the template location from the INT\_SYSTEM\_TEMPLATES Data Table based on the system and operation. This is a Commerce library function.

The Return Type, input information, and attributes used by this library function are shown below.

Commerce BML Library Function Editor: Properties & Parameters			
<b>Name:</b>	getTemplateLocation	<b>#</b>	<b>Parameter Name</b>
<b>Variable Name:</b>	getTemplateLocation	1	system
<b>Description:</b>	Queries the template location from the datatable INT_SYSTEM_TEMPLATES based on the System and Operation.	2	operation
<b>Return Type:</b>	String		<b>Parameter Type</b>
			String
			String

#### **String Dictionary** *getUserAttributes(String system)*

This is a Commerce library function that queries the user name and password from the INT\_SYSTEM\_DETAILS Data Table and adds it to the payload.

The Return Type, input information, and attributes used by this library function are shown below.

Commerce BML Library Function Editor: Properties & Parameters			
<b>Name:</b>	getUserAttributes	<b>#</b>	<b>Parameter Name</b>
<b>Variable Name:</b>	getUserAttributes	1	system
<b>Description:</b>	Queries UserName,Password from DataTable INT_SYSTEM_DETAILS and adds it in the payload.		<b>Parameter Type</b>
<b>Return Type:</b>	String Dictionary		String

Function Editor		
Hide Tools		
Attributes   Function Wizard   Debugger   Library Function(s)		
Main Document Attribute	Sub Document Attribute	System Attribute
<b>#</b> <b>Attribute</b>	<b>#</b> <b>Attribute</b>	<b>#</b> <b>Attribute</b>
1 orderId	transactionLine	1 _system_user_first_name

### String invokeWebService(String system, String soapReq)

This is a Commerce library function that invokes Web Services and returns the response. The Return Type, input information, and attributes used by this library function are shown below.

Commerce BML Library Function Editor: Properties & Parameters			
Name:	invokeWebService	#	Parameter Name
Variable Name:	invokeWebService	1	system
Description:	Invokes Web service and returns the response.	2	soapReq
Return Type:	String		Parameter Type
			String
			String

## Manual Data Table Changes

The INT\_SYSTEM\_DETAILS and INT\_SYSTEM\_TEMPLATES Data Tables are added to the Oracle CPQ Ref App site for account integration.

### INT\_SYSTEM\_DETAILS

SYSTEM(KEY)	USERNAME	ENDPOINT	DESCRIPTION
TCA-OrgService	<Enter the username here to call the web service endpoint>	<Enter the web service endpoint to call the service related to TCA>	TCA Find Organization details
TCA-AccService	<Enter the username here to call the web service endpoint>	<Enter the web service endpoint to call the service related to TCA>	TCA customer Account details

### Schema

As shown below, administrators must manually make **System** a primary key by selecting the **Key** check box.

Data		Schema								INT_SYSTEM_DETAILS	
<input type="checkbox"/> Live (Not Deployed)		Save		+ Add Column		+ Add Foreign Key		+ Add Relationship		✖ Remove Relationship	
#	✖	Index	Key	Name	Description	Type	Date Added	Date Modified	Validation Type	Validation Mapp	
1	✖	<input type="checkbox"/>	<input type="checkbox"/>	System		String	24.10.2017 10:34	06.07.2018 05:18	None		
2	✖	<input type="checkbox"/>	<input type="checkbox"/>	Description		String	24.10.2017 10:34	06.07.2018 05:18	None		
3	✖	<input type="checkbox"/>	<input type="checkbox"/>	Username		String	24.10.2017 10:34	06.07.2018 05:18	None		
4	✖	<input type="checkbox"/>	<input type="checkbox"/>	Password		Secure	24.10.2017 10:34	06.07.2018 05:18	None		
5	✖	<input type="checkbox"/>	<input type="checkbox"/>	MaxLinesInPayl...		Integer	24.10.2017 10:34	06.07.2018 05:18	None		
6	✖	<input type="checkbox"/>	<input type="checkbox"/>	Endpoint		String	19.06.2018 06:50	06.07.2018 05:18	None		
7	✖	<input type="checkbox"/>	<input type="checkbox"/>	SoapEndpoint		String	24.10.2017 10:34	06.07.2018 05:18	None		

System Selected as Primary Key

## INT\_SYSTEM\_TEMPLATES

SYSTEM(KEY)	OPERATION(KEY)	TEMPLATES
TCA-OrgService	FindOrg	<Enter the template URL path that is uploaded in File Manager>
TCA-AccService	FindAcc	<Enter the template URL path that is uploaded in File Manager>

### Schema

As shown below, administrators must manually make **System** and **Operation** primary keys by selecting the associated **Key** check boxes.

#	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Name	Description	Type	Date Added	Date Modified	Validatio...	Validation Mapp
1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	System	Name of the integrating system	String	22/07/2014 02:58	22/07/2014 02:59	None	
2	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Operation	Web service operation name for a feature	String	22/07/2014 02:58	22/07/2014 02:59	None	
3	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Template	Stores the template link/url	String	22/07/2014 02:59	22/07/2014 02:59	None	

System and Operation Selected as Primary Keys

## Add Template Dependencies to File Manager

Oracle CPQ administrators must add the Find Organization payload template file and the Customer Account payload template file to File Manager. These template files support account integration.

To add the template dependencies to File Manager, perform the following steps:

1. Download the payload template files (i.e. findOrganizationPayload.txt and customerAccountPayload.txt) from [My Oracle Support \(Doc ID 2508999.1\)](#).
2. Open the Admin Home page.
3. Create a new folder named TCA.
4. Click **Browse** under **Add Files**. The Choose File to upload dialog opens.
5. Navigate to the findOrganizationPayload.txt file and click **Open**.
6. Click **Add File**. The findOrganizationPayload.txt file displays in File Manager.
7. Complete steps 2-6 for the customerAccountPayload.txt file.

**Note:** To view the BML included in the payload template files, refer to [Appendix K: Payload Template File Content](#).

# ORACLE CPQ ACCOUNT LOOKUP INTEGRATION

Beginning in Oracle CPQ 19B, we enhance the Subscription Management solution by including the following functionality:

- Accounts Lookup Library
- Account REST API Services

To enable the Accounts Lookup Library function for Integrations with CRMs, complete the following steps:

1. Navigate to the Admin Home page.
2. Select **Commerce Settings** under **Commerce and Documents**. The Commerce Options page displays.
3. Select an Account Lookup BML script from the **Accounts Lookup Library Function** drop-down.

Commerce Options	
Options - Commerce	
Number of Milliseconds to Wait Before Showing the Loading Dialog for Ajax Rules	<input type="text" value="1000"/>
Allow Commerce Processes and Invocations to be Deployed and Undeployed	<input checked="" type="radio"/> Yes <input type="radio"/> No
Allow Commerce Processes to be Cloned and Migrated	<input type="radio"/> Yes <input checked="" type="radio"/> No
Allow the Transaction ID to be included in the Commerce Search	<input checked="" type="radio"/> Yes <input type="radio"/> No
Hide Commerce Invocation Buttons on Add From Catalog	<input type="radio"/> Yes <input checked="" type="radio"/> No
Hide Add To Cart Button on Add From Catalog	<input type="radio"/> Yes <input checked="" type="radio"/> No
Show Select Language Preference menu for Print, Email and Export Actions	<input type="radio"/> Yes <input checked="" type="radio"/> No
Dialog Dimensions - Commerce Search	<input type="text" value="800"/> Width <input type="text" value="600"/> Height
Dialog Dimensions - Quick Links	<input type="text" value="400"/> Width <input type="text" value="900"/> Height
Error Box Behavior	<input checked="" type="radio"/> Default <input type="radio"/> Collapsible
Document Engine - Validate page data on save	<input type="radio"/> Yes <input checked="" type="radio"/> No
Enable Transaction Total for Mobile Layouts	<input checked="" type="radio"/> Yes <input type="radio"/> No
Ignore blank attributes in SOAP API Payload for modify action	<input checked="" type="radio"/> Yes <input type="radio"/> No
Display Part Number on Part Display Quick Key Search Results	<input checked="" type="radio"/> Yes <input type="radio"/> No
Doc Designer - Preserve line feeds for XSL snippets. Documents must be deployed for any change to take effect.	<input type="radio"/> Yes <input checked="" type="radio"/> No
Enable Company Associations on Part Display Search	<input type="radio"/> Yes <input checked="" type="radio"/> No
Commerce Timeout Action	<input type="radio"/> Kill and Log <input checked="" type="radio"/> Log Only <input type="radio"/> No Action <input type="text" value="2"/> minutes
Allow Users to Select Currency for Quotes Created from the Transaction Manager	<input type="radio"/> Yes <input checked="" type="radio"/> No
Transfer Advanced Pricing Profiles JSON to Commerce	<input checked="" type="radio"/> Yes <input type="radio"/> No
Enable sticky header for line item grid	<input type="radio"/> Yes <input checked="" type="radio"/> No
Number of columns to freeze on line item grid	<input type="text" value="0"/>
Disable Transaction Item Count	<input type="radio"/> Yes <input checked="" type="radio"/> No
Accounts Lookup Library Function	queryIntegratedAccounts[_account.queryIntegratedAccounts]
Enable Subscription Ordering for Simple Products	<input checked="" type="radio"/> Yes <input type="radio"/> No
Enable HTML Approval Email	<input type="radio"/> Yes <input checked="" type="radio"/> No

[Back to Top](#)

4. Click **Apply**.

**Note:** The Accounts Lookup Library Function drop-down is only available for sites that are integrated with a CRM. Refer to the Oracle CPQ Administration Online Help for information about the Integration Center.

Administrators need to develop an Account Lookup BML script in order for the lookup feature to know what data to search. A sample BML script is included within the Subscription Management Installation package. The following examples provide the input and output BML format for the accounts lookup script.

```
{
  "fields": ["firstName", "companyName", "customerId"],
  "q": {
    "$and": [{
      "customerId": {
        "$like": "account2%"
      }
    }, {
      "id": {
        "$gte": "21"
      }
    }
  ]
}
```

```

    "companyName": {
      "$like": "account2%",
      "$options": "I"
    }
  }, {
    "firstName": {
      "$exists": true
    }
  }, {
    "customerRep": {
      "$like": "account2%",
      "$options": "I"
    }
  }, {
    "$or": [{
      "lastName": {
        "$exists": true
      }
    }, {
      "supplierId": {
        "$eq": "123"
      }
    }
  ]
}
]
},
"offset": 0,
"limit": 10,
"orderby": ["firstName:ASC", "companyName:DESC"],
"version": "v8"
}

```

Sample Input BML

```

{
  items: [{
    "firstName": "Fname_OSC_account138",
    "lastName": "Lname_OSC_account138",
    "phone": "Phoneno_account138",
    "companyName": "Company_account138",
    "customerId": "account138",
    "id": 4162404,
    "fax": "Fax_account138",
    "email": "first.last@yourcompany.com",
    "_crm_custom_msm": "value1~value2~value3",
    "_crm_custom_ssm": "value1",
    "customerRep": "CR_account138"]
  }, {
    "firstName": "Fname_OSC_account139",
    "lastName": "Lname_OSC_account139",
    "phone": "Phoneno_account139",
    "companyName": "Company_account139",
    "customerId": "account139",
    "id": 4162408,
    "fax": "Fax_account139",
    "email": "first.last@yourcompany.com",
    "_crm_custom_msm": "value2~value3",
    "_crm_custom_ssm": "value1",
  }
}

```



```
    "customerRep": "CR_account139"
  }
]
error: {
  system: true
  errorMessage: "tokenized string",
  errorTokenValues: ["a", "b"]
}
}
```

Sample Output BML

Note the following eRestLayer Query for reference:

KEY	VALUES
fields	customerId, firstName, companyName
totalResults	true
q	{ "\$and": [{"customerId": {"\$like": "account296"}}, {"id": {"id": {"\$gte": "21"}}, {"c...
Offset	0
Limit	10
Orderby	firstName:DESC,companyName

### Account REST API Services

Account REST API services are added to support integrating and querying external system accounts. The Accounts REST API is added to the Integration Catalog. The following services are available:

- Get Account
- Get Accounts

#### Custom Account Attributes

You may need custom account attributes. By default, all customer account deployed custom account attribute types are available in the account object except for the following:

- Single Select Menu with the menu value exceeding 30 characters
- Attribute variable names exceeding 116 characters

All custom account attributes are given the accounts object prefix `_crm_custom`. In the case of native accounts, where CPQ is the source master, Single Select Menu and Multi-select Menu attributes are not searchable and custom attribute filters are required.

The table below defines differences between native and integrated Oracle CPQ Subscription Management custom attribute feature support.

FEATURE	NATIVE (CPQ IS MASTER SOURCE)	INTEGRATED (INTEGRATION IS MASTER SOURCE)
Complex search with more than one level	Supported	Not Supported
Custom attribute filters	Supported	Not Supported
Search on Multi-Select Menu and Single Select Menu	Not Supported	Administrators can extend support in BML
Limit	Maximum 1000 characters	Maximum 500 characters
hasMore	Supported	Supported with some restrictions
Translations	Not Supported	Not Supported

### Get Account

<b>Get Account</b> <span style="background-color: #0070c0; color: white; padding: 2px 5px; border-radius: 3px;">GET</span>			
<b>Description</b>	This action returns the specified account's information.		
<b>URI Endpoint</b>	<code>/rest/v13/accounts/{customerId}</code>		
<b>Endpoint Parameter(s)</b>	<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="width: 30%;"><code>customerId</code></td> <td>The variable name for the account or customer ID number</td> </tr> </table>	<code>customerId</code>	The variable name for the account or customer ID number
<code>customerId</code>	The variable name for the account or customer ID number		
<b>HTTP Method</b>	GET		
<b>Request Body Parameters</b>	None		
<b>Response Body Parameters</b>	JSON data containing the account information for the specified account		

#### URI ENDPOINT SAMPLE

`https://site.oracle.com/rest/v13/accounts/100`

## SAMPLE RESPONSE BODY

```
{
  "firstName": "Jones Pizza",
  "phone": "2135555555",
  "companyName": "Jones Pizza",
  "customerId": "100",
  "email": "ljones@jonespizza.net",
  "links": [{
    "rel": "self",
    "href": "https://sitename.oracle.com/rest/v13/accounts/100"
  }]
}
```

## Get Accounts

Get Accounts <span>GET</span>	
<b>Description</b>	This action returns the list of accounts.
<b>URI Endpoint</b>	/rest/v13/accounts
<b>Endpoint Parameter(s)</b>	None
<b>HTTP Method</b>	GET
<b>Request Body Parameters</b>	None
<b>Response Body Parameters</b>	JSON data containing a list of accounts and the account information for each of the accounts in the list

## URI ENDPOINT SAMPLE

```
https://sitename.oracle.com/rest/v13/accounts
```

SAMPLE RESPONSE BODY

```

{{
  "hasMore": false,
  "links": [{
    "rel": "self",
    "href": "https://sitename.oracle.com/rest/v13/accounts"
  }
],
  "items": [{
    "firstName": "Jones Pizza",
    "phone": "2135555555",
    "companyName": " Jones Pizza",
    "customerId": "100",
    "email": "ljones@jonespizza.net",
    "links": [{
      "rel": "self",
      "href": "https://sitename.oracle.com/rest/v13/accounts/100"
    }
  ]
}, {
  "firstName": "Smith Trucking",
  "phone": "8475555555",
  "companyName": "Smith Trucking Incorporated",
  "customerId": "102",
  "email": "jsmith@smithtruck.com",
  "links": [{
    "rel": "self",
    "href": "https://sitename.oracle.com/rest/v13/accounts/102"
  }
]
}, {
  "firstName": "Jackson Rentals",
  "phone": "1235555555",
  "companyName": "Jackson Rentals",
  "customerId": "103",
  "email": "mwjackson@jacksonrental.com",
  "links": [{
    "rel": "self",
    "href": "https://sitename.oracle.com/rest/v13/accounts/103"
  }
]
}, {
  "firstName": "Morris Foods",
  "phone": "3215555555",
  "companyName": "Morris Foods",
  "customerId": "104",
  "email": "jmorris@morrisfoods.net",
  "links": [{
    "rel": "self",
    "href": "https://sitename.oracle.com/rest/v13/accounts/104"
  }
]}]}}

```

## Reference Accounts Integration

Reference Accounts Integration, for example OSC Integration, use SOAP API to query. The information to retrieve Reference Accounts is obtained through the following information process flow:

- **End point Information** - This is captured using a generic integration.
- **CPQ Account Attribute to OSC Account Attribute Mapping** - This is captured in a data table named 'Account\_attribute'.
- **Understand the Search Request** – To understanding the search that was requested by the user (i.e., browser, API client) review a set of BMLs.
- **Transform the User Requested Search to the External Search Syntax to Query the External System** - CPQ Search Operator to OSC Search Operator mapping is in a data table named 'Account\_Operator'.
- **Convert the OSC Accounts Response using an XSL File** - This is stored in the file manager under TCA folder findOrganizationResultTransformation.XSL.

For additional information, refer to Oracle CPQ to Oracle CX Sales Integration Guide, [Doc ID 2015009.1](#) on [My Oracle Support](#).

## Account Search Data Tables

This section contains the Account Search mapping details.

### ACCOUNT ATTRIBUTE MAPPING

KEY	INDEX	NAME	DESCRIPTION	TYPE	COMMENTS
Yes	Yes	cpq_attribute	Attribute name of account object in CPQ	String	For example: <code>customerId</code> Customer attribute example: <code>_crm_custom_mainContactAddress</code>
No	No	external_attribute	Attribute name of account object in external system (integrated partner)	String	This can be the attribute name of the top-level (root) entity for simple mapping, or can be an attribute of the child entity (first level).  For example: <code>PartyUsageAssignment.AssignmentCode</code> where <code>PartyUsageAssignment</code> is the child entity and <code>AssignmentCode</code> is the attribute of the child entity.  Note: If there are two or more records for the child, they are delimited by a tilde.  You must ensure that there is only one record in case the <code>cpq_attribute</code> is mapped to is a not a Multi Select Menu.
No	No	queryable	Flag to indicate whether the attribute is queryable (search or sortable)	String	Values can be Y for Yes and N for No.  Note: If the <code>external_attribute</code> is referring to the child-entity's attribute name, then <code>queryable</code> must be N because you cannot search and sort on the child entity
No	No	textual	Flag to indicate whether the attribute is a type string	String	Values can be Y for Yes and N for No.  For OSC Accounts, the <code>PartyID</code> is numeric, hence the value is set as N.

## ACCOUNT OPERATOR MAPPING

KEY	INDEX	NAME	DESCRIPTION	TYPE	COMMENTS
Yes	Yes	cpq_operator	The operator conforming to the CPQ Rest Syntax	String	For example: \$like
No	No	external_operator	The operator conforming to external system syntax	String	For example: STARTSWITH

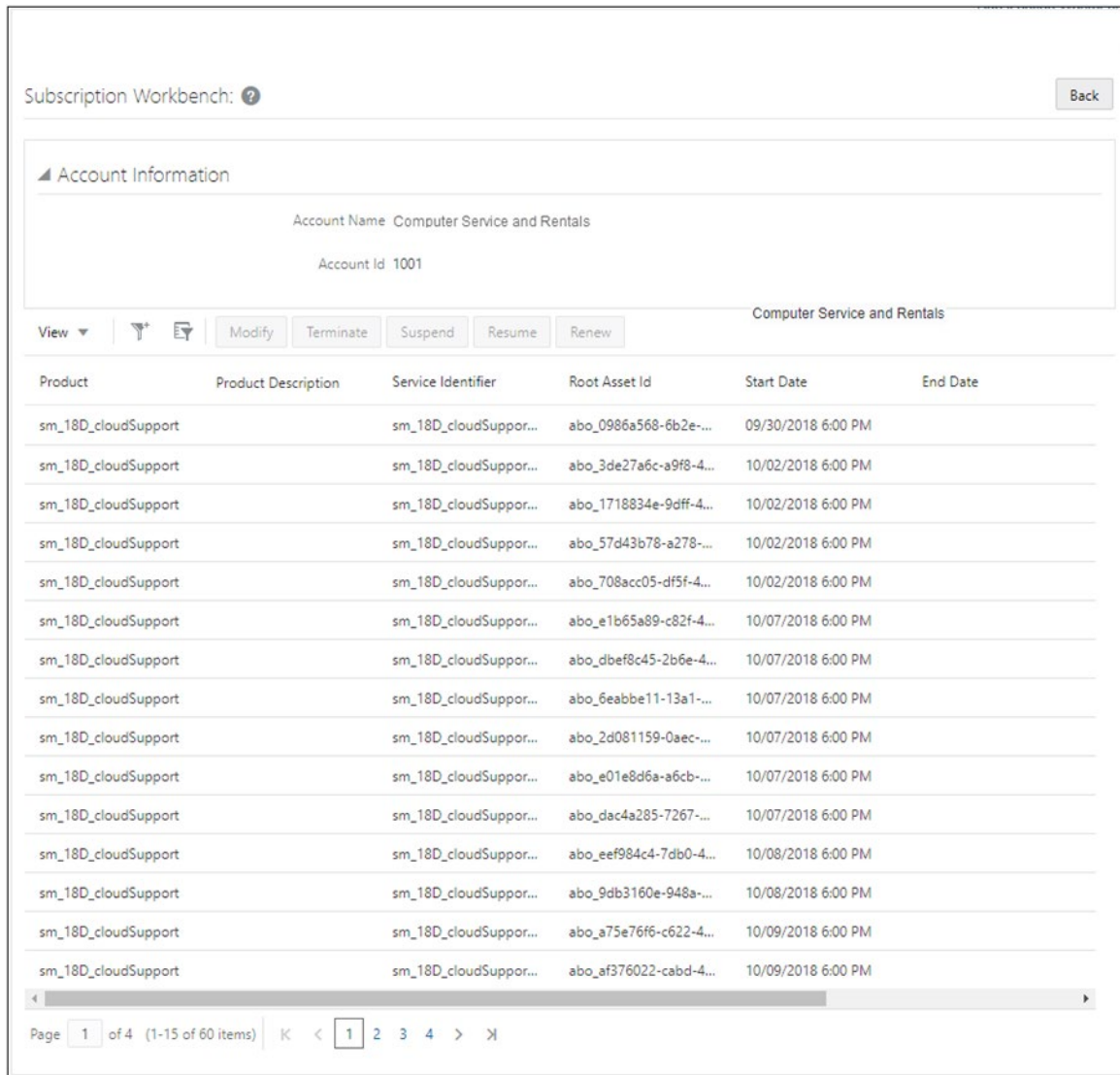
### Notes:

- Only simple search syntax is supported for Accounts data. As such, only one level of query parameters are supported. Nested query parameters are not supported.
- Search on Multi Select, Single Select menu attributes is not supported. Therefore, the query column in the Data Table must be set to No.
- The maximum number of items that can be searched in one time is 500. If more than 500 items are requested for searching, you will not receive an error message but the results will only be returned for the first 500 items.
- The **hasMore** property is not strictly honored in the Response payload. For example, a response of True is always returned if the number of rows is the same as the limit specified in the search.

## SUBSCRIPTION WORKBENCH

Oracle CPQ 19B and later enhances the Customer Assets List page, now known as the Subscription Workbench, to provide sales users with a centralized location to easily access subscription information by account ID or account name. In Oracle CPQ 19A or earlier, customer's leveraging CPQ Subscription Ordering functionality are restricted to a predefined flow for users. To see the list of assets owned by a given account, a sales user was required to access a Transaction first. This enhancement decouples the Subscription Workbench page from the Transaction UI and allows users to view the assets/subscriptions belonging to a given account directly through a navigation link.

To view the Subscription Workbench, a user-defined link must be set up. If the user-defined link is not set up, the Subscription Workbench is only viewable from the Transaction.



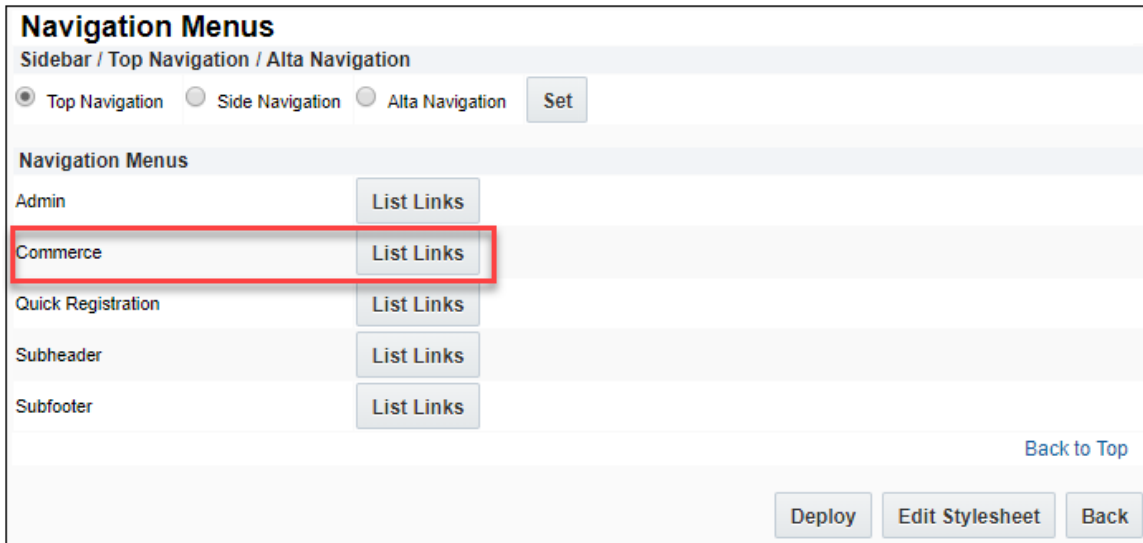
The screenshot displays the Subscription Workbench interface. At the top, it says "Subscription Workbench: ?" with a help icon and a "Back" button. Below this is a section for "Account Information" showing "Account Name: Computer Service and Rentals" and "Account Id: 1001". A table of actions is visible, including "View", "Modify", "Terminate", "Suspend", "Resume", and "Renew". The main part of the interface is a table with columns: Product, Product Description, Service Identifier, Root Asset Id, Start Date, and End Date. The table lists 15 rows of "sm\_18D\_cloudSupport" subscriptions, each with a unique Root Asset Id and a Start Date ranging from 09/30/2018 to 10/09/2018. At the bottom, there is a pagination control showing "Page 1 of 4 (1-15 of 60 items)" and navigation arrows.

Product	Product Description	Service Identifier	Root Asset Id	Start Date	End Date
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_0986a568-6b2e-...	09/30/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_3de27a6c-a9f8-4...	10/02/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_1718834e-9dff-4...	10/02/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_57d43b78-a278-...	10/02/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_708acc05-df5f-4...	10/02/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_e1b65a89-c82f-4...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_dbef8c45-2b6e-4...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_6eabbe11-13a1-...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_2d081159-0aec-...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_e01e8d6a-a6cb-...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_dac4a285-7267-...	10/07/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_eef984c4-7db0-4...	10/08/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_9db3160e-948a-...	10/08/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_a75e76f6-c622-4...	10/09/2018 6:00 PM	
sm_18D_cloudSupport		sm_18D_cloudSuppor...	abo_af376022-cabd-4...	10/09/2018 6:00 PM	

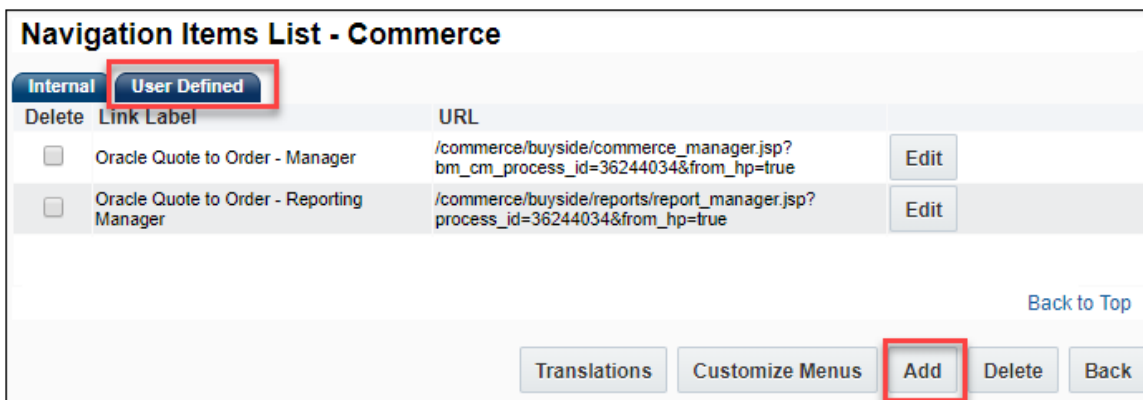
Sample Subscription Workbench

To setup a user-defined Link for accessing the Subscription Workbench, complete the following steps:

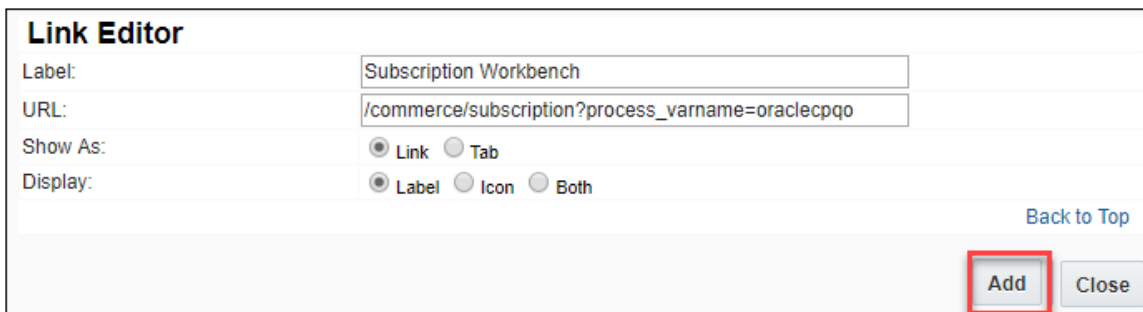
1. Navigate to Admin Home page.
2. Click **Navigation Menus** under **Styles and Templates**. The Navigation Menus page displays



3. Click **List Links** for Commerce. The Navigation Items List – Commerce page displays.
4. Select the **User Defined** tab.



5. Click on **Add**. The Link Editor page displays.



6. Enter Subscription Workbench for the **Label**.
7. Enter a valid **URL**. For example, /commerce/subscription?process\_varname={PROCESS\_VARNAME} where process\_varname is the valid Commerce process.
8. Select **Link** for **Show As**.



- Select **Label** for Display.
- Click **Add**. The Navigation Items List - Commerce page displays.

**Navigation Items List - Commerce**

Internal User Defined

Delete	Link Label	URL	
<input type="checkbox"/>	Oracle Quote to Order - Manager	/commerce/buyside/commerce_manager.jsp?bm_cm_process_id=36244034&from_hp=true	Edit
<input type="checkbox"/>	Oracle Quote to Order - Reporting Manager	/commerce/buyside/reports/report_manager.jsp?process_id=36244034&from_hp=true	Edit
<input type="checkbox"/>	Subscription Workbench	/commerce/subscription?process_varname=oraclecpqo	Edit

Back to Top

Translations Customize Menu Add Delete Back

- Click **Customize Menus** within the **User Defined** tab. The Customize Links for Commerce page displays.

**Customize Links for Commerce**

Company Type: FullAccessWithESales User Type: FullAccess Go

Hidden Links Add/Remove Link

Subscription Workbench  
Home Page (Internal)  
Favorites (Internal)

Left Add Remove Association

Top Navigation Side Navigation

Top Navigation Alignment and Ordering

Column Width: 33% Column Width: 33% Column Width: 33%

Hide Hide Hide

Back to Top

Add Apply Update Back

- Select **Subscription Workbench** from the Hidden Links list and then click **Add**. The Subscription Workbench link is moved to under the Top Navigation Tab.

13. (Optional) Click the arrows to move the Subscription Workbench link to the desired location to display on the UI.

**Customize Links for Commerce**

Company Type: FullAccessWithESales User Type: FullAccess Go

Hidden Links Add/Remove Link

Home Page (Internal)  
Favorites (Internal)

Left Add Remove Association

Top Navigation Side Navigation

Top Navigation Alignment and Ordering

Column Width: 33% Column Width: 33% Column Width: 33%

Oracle Quote to Order - Manager  
Subscription Workbench  
Oracle Quote to Order - Reporting

Hide Hide Hide

Back to Top

Add Apply Update Back

14. Click **Update** to get back to Navigation Items List – Commerce page.

15. Click **Back** to get to the Navigation Menus page.

**Navigation Menus**

Sidebar / Top Navigation / Alta Navigation

Top Navigation Side Navigation Alta Navigation Set

Navigation Menus

Admin	List Links
Commerce	List Links
Quick Registration	List Links
Subheader	List Links
Subfooter	List Links

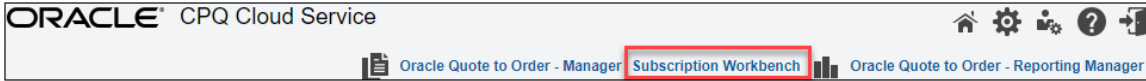
Back to Top

Deploy Edit Stylesheet Back

16. Click **Deploy** to deploy the Subscription Workbench to the User-Defined Navigation Menu UI.

To view the Subscription Workbench, complete the following steps:

1. Log in to CPQ, click **Subscription Workbench**. The Subscription Workbench page displays




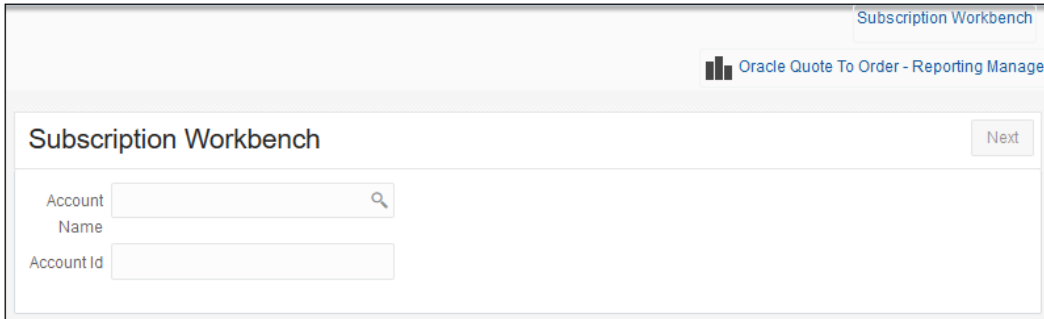
Subscription Workbench

2. Perform one of the following steps:

Enter the account name in the **Account Name** field.

Enter the account ID in the **Account Id** field.

Click on the  Look up Key to select an account.



3. Click **Next**. The Subscription Workbench showing the list of assets/subscriptions for that account displays.

## SUBSCRIPTION PRICING INTEGRATION

OSS generates a billing schedule and bills the subscription in periodic intervals, as configured in the Oracle CPQ Transaction. When customers opt for usage-based pricing for the subscription, OSS invokes the Oracle CPQ Pricing REST API. The subscription is price based on its usage/consumption.

### Enable Subscription Pricing

To enable Subscription Pricing, perform the following steps in Oracle CPQ:

1. Open the Admin Home page.
2. Select **General Site Options** under **General**. The Options – General page opens.
3. Set the Apply only the first matching Pricing profile option to No.
4. Click **Update**. The Admin Home page opens.
5. Select **Commerce Settings** under **Commerce and Documents**. The Commerce Options page opens.
6. Set the **Commerce Pricing Behavior** option to Version 2.

**Note:** If **Commerce Pricing Behavior** is not displayed as a **Commerce Setting**, edit the BM Context to remove the `disabled_pricing_behavior`. The `disabled_pricing_behavior` must be blank in the BM Context to display in the **Commerce Settings** drop-down.

7. Click **Update**.

### Charges

The following three charges are supported by the Subscription Management solution:

- One Time Fee (e.g. Activation Fee)
- Recurring Fee (e.g. Monthly Fee)
- Usage Fee (e.g. Consumption Fee)

**Notes:**

- Pricing for One Time charges and Recurring changes is calculated as unit price x quantity.
- User BOM rules to model the charge structure for products in the Subscription Management solution.
- For the Subscription Management solution to work, product items must be added as parts under the Model.
- Two kinds of pricing are supported for Recurring Usage Fee charges: Non Tier Pricing and Tier Pricing.

### Discounts

Sales users can apply auto discounts and can override discounts on One Time, Recurring, and Usage Fee charges.

- Discounts are queried from the data table and populated in Transaction Line.
- These auto-populated discounts can be overridden by sales user.
- These discounts are sent to OSS upon clicking **Submit Order**.

## Pricing Engine Setup

The following Subscription Pricing-related configuration is deployed by default with the CPQ Subscription Management package. Pricing Definitions are stored in the following data tables.

### ORCL\_PRC\_BASE\_CHGS

This data table stores the charge-related information. The price definition of each products needs to be added to this table.

- Charge\_Id: Unique Id for the charge
- Product: Product name
- AsOfDate: The date the information provided is valid from.
- Block\_Size: Block size for the charge
- Price: Price for the charge
- Allowance: Allowance for the charge
- Charge\_Type: Type of the charge (One Time, Recurring, or Usage)
- Periodicity: Periodicity determines the frequency at which billing invoice is available for each product.
- Tiered: Recurring Usage Charge can be Tiered or Non Tiered (Y or N)
- TierType: If the Recurring Usage Charge is designated as Tiered, it can be set as ORA\_ALL\_TIERS or ORA\_HIGHEST\_TIER. By default ORA\_ALL\_TIERS is supported for the pricing profile. The default can be modified to ORA\_HIGHEST\_TIER pricing profile.

### ORCL\_PRC\_BASE\_TIERS

This data table stores the tier information for the parts which have tiered price set in the price definition. Tier information is only used when Recurring Usage Charge is set as Tiered in the pricing profile.

- Charge\_Id: Charge\_Id of the charge for which tiers are configuring
- Tier\_Min: Tier Starting range
- Tier\_Max: Tier Ending range
- Block\_Size: Block Size for the corresponding Tier
- Price: Price for the corresponding Tier

### ORCL\_PRC\_DISCOUNTS

This data table stores the discount data for each part. Discounts that are defined as part of the price definition are added to this data table.

- Product: Product Name of the charge
- Discount\_Name: Name of the adjustment
- Discount\_Type: Type of the adjustment. Supported values are: Percent Off, Amount Off, and Price Override
- Discount\_Value: Actual value to be adjusted
- Discount\_Effectivity: Effectivity of discounts. Supported value is ORA\_ALL\_TERM.
- Effectivity\_Periods: This column can be used if ORA\_PERIODS\_FROM\_START\_DT or ORA\_PERIODS\_BEFORE\_END\_DT discount effectivity has to be used.
- Start\_Date: Discount is applicable to product if contract start date is same or after date mentioned in this column
- End\_Date: Discount is applicable to product if contract start date is same or before date mentioned in this column
- Discount\_Reason: Reason to apply discount
- Discount\_Desc: Description about the discount

## ORCL\_PRC\_LOOKUP

This data table stores the CPQ and OSS lookups for periodicity and charge type.

- Type: Lookup type. Periodicity and ChargeType
- CPQ\_Code: CPQ code for periodicity and charge type
- OSS\_Code: OSS code for periodicity and charge type

**Note:** This table can be used if lookups for any other type has to be introduced. For demo setup this table is configured with periodicity and charge type.

## PRICING PROFILE CONFIGURATION FOR DEMO DATA

**Profile Name:** *Subscription Charges*. The Subscription Charges pricing profile filters the list of products by pricing profile action. By default, the profile action executes for Crossover Outright Buy, Crossover SUV Subscription, Sports SUV Subscription, Luxury SUB Subscription, Standard Maintenance Plan, Premium Maintenance Plan, Sirius XM Radio, WiFi Service, and SUB Charging Station use.

Below simple condition is used to filter out parts for pricing engine:

Product Type = subscription

Prices of all the parts, whose value for “Product Type” field is set to “subscription”, will be decided by pricing engine.

The screenshot displays the 'Product Pricing' configuration page for the 'Subscription Charges' profile. The 'Profiles' tab is active, showing a table of conditions. The table has columns for '#', 'Attribute Name', 'Operator', and 'Attribute Value'. A single condition is listed with '# 1', 'Attribute Name' 'Product Type', 'Operator' '=', and 'Attribute Value' 'subscription'. Below the table, the 'Row Grouping' is set to 'AND ALL' with a count of '1'. The 'Pricing Formula' is set to 'ADVANCED PRICING'. The 'Discount Type' is 'Advanced'. The interface includes a 'Filter Profiles' search box, an '+ Add Profile' button, and a 'View Linked Rules' button.

#	Attribute Name	Operator	Attribute Value
1	Product Type	=	subscription

**Profile BML:** *Profile Action is associated to Advanced BML*. This BML is associated to the profile action and contains the logic for price calculation. Administrators can customize the BML logic to change the pricing logic.

Profile BML invokes the `oRCL_pRC_oraclePricingSubscriptionBaseProfile` utility BML. This returns JSON calculation information that provides all the charge information.

## PRICING RELATED UTILITY BMLS

The following are the Utility BMLs used by Subscription Pricing. The BML source code is available in [Appendix M: Subscription Pricing Utility BMLs](#).

### ***oRCL\_pRC\_oraclePricingSubscriptionBaseProfile BML***

This utility is associated to the pricing profile and internally calls the following utility BMLs:

- Calculate List Price: Calculates the List Price
- Prepare Tier Info JSON: Prepares the Tier JSON information
- Get Lookups: Provides OSS lookups for periodicity and charge types.

## Utility BML

There are two utility BMLs which populate calculated pricing information for arrays. These BMLs are invoked from the Transaction Line Advanced Default - After Formulas.

- **Populate Charges:** Populates the charge information for the product

The screenshot displays the Oracle CPQ Transaction form. The top navigation bar includes buttons for Save, Submit, Close, Update Sales, Customer Assets, Generate Proposal, Submit Order, Customer Rejection, and Pipeline Viewer. The main form is divided into several sections: Transaction Details, Customer Details, Pricing Details, and Troubleshooting and Support Controls. The Transaction Details section shows fields for Transaction Name (07/26), Transaction Number (CPQ-381-36811360), Version (1), Customer Company Name, Customer First Name, Customer Last Name, and OSC File Attachment (Subscription Id and Subscription Status). The Customer Details section shows Owner (Super User), Created Date (07/26/2021), Last Updated By (Super User), Win/Loss Status (In Progress), and various dates (Default Request Date, Max Request Date, Contract Start Date, Contract End Date). The Pricing Details section shows Current Step Var Name (start\_step) and SubscriptionProfileId (300100172161475). The Troubleshooting and Support Controls section shows a table of charges.

Grp Se...	Product #	updateOldSu...	Action Code	Fulfillment St...	Change Code	Change Reas...	Contract Star...	Contract End ...	Contract Perio...	Discount	Discou
1.0	SUV	True	Add	Created			07/26/2021	07/25/2022			
1.1	Crossover SUV Subscription	True	Add				07/26/2021	07/25/2022	12	10.00	Percen
1.2	Standard Maintenance Plan	True	Add				07/26/2021	07/25/2022	12		

- **Populate Tiers:** Populates the tier information for tier arrays

## APPENDIX A: CREATE SUBSCRIPTION WORKFLOW

OIC is the middleware used to establish an integration between Oracle CPQ and OSS. Once this integration is established, sales users can use Oracle CPQ to create a Transaction and invoke OIC to create a subscription in OSS. To access Oracle CPQ to create a Transaction, sales users directly log in Oracle CPQ or gain access through a Customer Relationship Management (CRM) opportunity.

To create a subscription, perform the following steps in Oracle CPQ:

1. Navigate to Transaction Manager.
2. Click **New Transaction**. The Transaction page opens.
3. Select the date the subscription is to take effect from the **Default Request Date** field, which is used to specify the date when the subscription order should be activated. After approval of the Transaction, the Submit Order button is available on the Transaction page.

The screenshot shows the Oracle CPQ Transaction page. At the top, there are buttons: Save, Submit, Close, Update Sales, Customer Assets, Generate Proposal, **Submit Order** (highlighted with an orange box), Customer Rejection, and Pipeline Viewer. Below the buttons, the page is titled "Transaction" and has tabs for Transaction Details, Customer Details, Pricing Details, and Troubleshooting and Support Controls. The main content area displays transaction details such as Transaction Name (07/26), Transaction Number (CPO-385-36811360), Version (1), Customer Company Name, Customer First Name, and Customer Last Name. It also shows fields for Subscription End Date, Default Request Date (07/26/2021), Current Step Var Name (star\_step), Max Request Date (07/26/2021), Subscription Profile Id (300100172161475), Contract Start Date, and Contract End Date. A section titled "OSS File Attachment" contains fields for Subscription Id and Subscription Status, both highlighted with orange boxes. Below this is a table with columns: Grp Se..., Product #, updateOldSt..., Action Code, Fulfillment St..., Change Code, Change Rees..., Contract Star..., Contract End..., Contract Perio..., Discount, and Discou. The table has three rows: 1.0 SUV (Fulfillment St...: Created), 1.1 Crossover SUV Subscription, and 1.2 Standard Maintenance Plan.

Submit Order Button

### Notes:

- When the Subscription Management package is installed, a Subscription Status field and a Subscription Id field are added. Administrators can add these fields to the layout.
  - OSS requires the generation of a billing schedule as part of subscription creation. If billing or charge-related fields are missing or incorrect in the Create subscription payload, the billing schedule is typically not generated. Administrators can resolve this discrepancy by attempting to activate the subscription in OSS.
4. Click **Submit Order**. Once the subscription is created, the **Subscription Id** is updated.
    - a. If subscription activation is successful, the Subscription Status field updates to "Success".
    - b. If the billing schedule is not generated, a new Subscription id is created, but the Subscription Status field updates to "Failure".



- When the subscription reaches the Subscription Created state, the Update Asset Timer is triggered. Refreshing the Transaction changes the Fulfillment Status from Created to Fulfilled.

#### Fulfillment Status – Fulfilled

#### Notes:

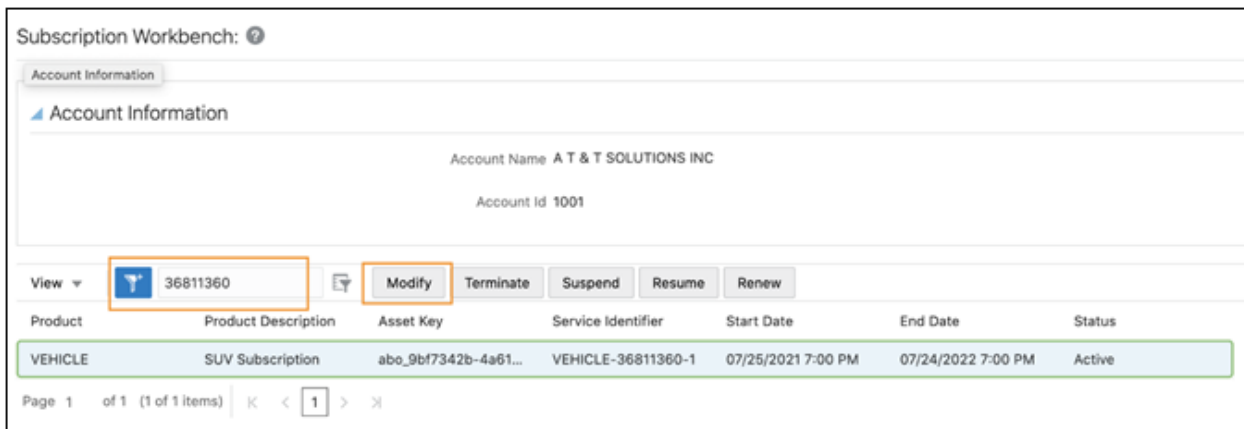
- The Update Asset Timer is configured in the Subscription Created step and monitors the Max Request Date field. The Max Request Date field, which is part of the Subscription Management Installation Package, defaults to the Default Request Date during the creation of a Transaction.
- The Update Asset Timer create assets only when the Contract Start Date is less than or equal to the Max Request Date. After the Update Asset Timer is executed, the Max Request Date field is updated to the next Contract Start Date. The Update Asset Timer will then execute on this date. CPQ Commerce Timer functionality is leveraged to activate the asset as specified on the request date. For details about Timer functionality, refer to the Oracle CPQ Administration Online Help.

## APPENDIX B: AMEND SUBSCRIPTION WORKFLOW

The Subscription Management solution uses Subscription Ordering functionality to support the modification of an existing asset-based subscription. After a subscription is created and fulfilled, sales users can use the Subscription Ordering Modify flow to change the subscription product, quantity, or duration.

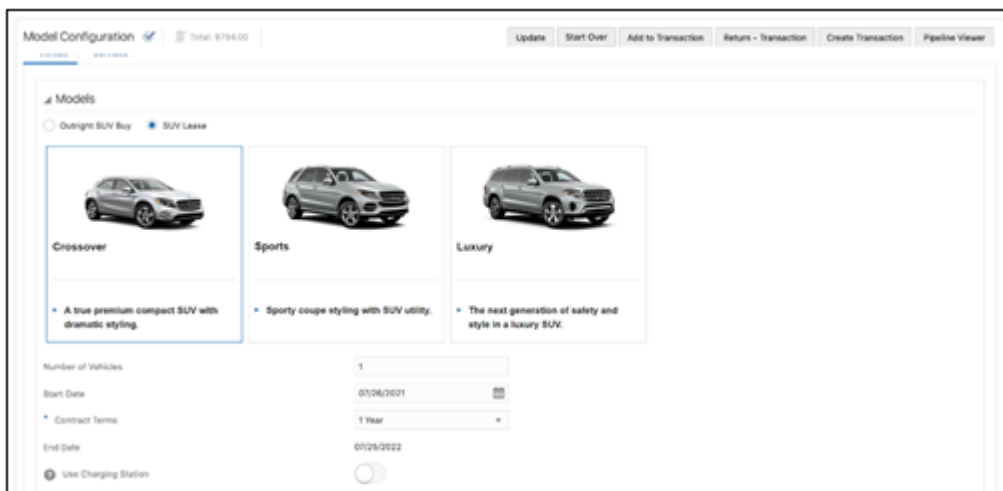
To amend a subscription, perform the following steps in Oracle CPQ:

1. Navigate to Transaction Manager.
2. Click the Transaction Number associated with the asset to amend. The Transaction page opens.
3. Click the **Customer Assets** button. The Customer Assets page opens.
4. Select the asset to modify. Sales users must know the asset key associated with the asset-based subscription to modify.



### Customer Assets

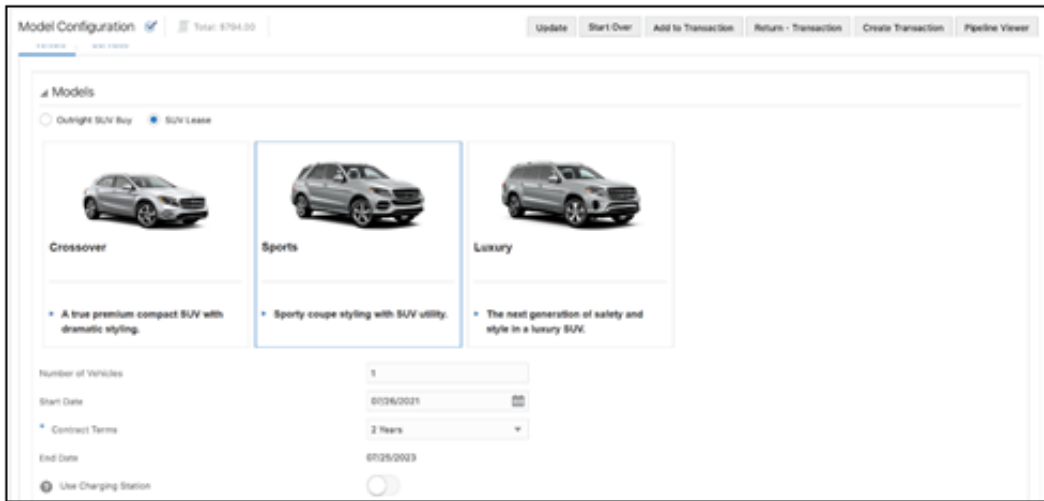
5. Click **Modify**. The Model Configuration page opens.



### Model Configuration Page

6. Modify the subscription service, quantity, or duration.

7. Click **Update**.



Model Configuration Page

8. Click **Create Transaction**. The Transaction page opens.
9. (Optional) Click **Add Line Item** to add one or more additional products.

Click **Update**.

Click Add to Transaction.

The modified subscription details are shown on the Transaction page.

10. Select the date the amended subscription is to take effect from the **Default Request Date** field.
11. Select a **Change Reason**. Select the appropriate Change Reason based on your business need from the following: **Non-Compliant, Breach, Quantity Change, Downgrade, Term Change, and Upgrade**.

If the Change Reason is Upgrade or Downgrade, the Amend Replacement field must be the instance Id of the deleted product.

<input type="checkbox"/>	Grp Se...	Product #	updateOldSu...	Action Code	Fulfillment St...	Change Code	Change Reason	Amend Replacement	Contract Star...	Contra
<input checked="" type="checkbox"/>	1.0	SUV	False	Update	Created				07/26/2021	07/25/...
<input type="checkbox"/>	1.1	Sports SUV Subscription	False	Add		Full	Upgrade	abc_ae76d54c-9c1f-42d3-92c1-736e...	07/26/2021	07/25/...
<input type="checkbox"/>	1.2	Standard Maintenance Plan	False	Update		Full	Term Change		07/26/2021	07/25/...
<input type="checkbox"/>	1.3	Crossover SUV Subscription	False	Delete		Full	Upgrade		07/26/2021	07/25/...

Transaction Page – Change Reason and Change Code Fields

**Note:** The amendment process requires CPQ to send fields like Change Reason, Change Code, Amendment Replacement, Instance ID, etc. to OSS. Administrators can write Commerce rules to set the values for these fields as part of the amendment flow call to OSS.

12. Select a **Change Code**. Select the appropriate Change Code based on your business need from one of the following options: **Full, Prorate without Credit, and Prorate with Credit**.
13. Use the **UpdateOldSubscription** menu to determine whether to apply the subscription amendments to a new subscription or to the existing subscription. In the following example, the amendments to the subscriptions are applied to a new subscription.
  - When the **UpdateOldSubscription** menu is set to **False**, the amended product lines are added to a new subscription in OSS.
  - When the **UpdateOldSubscription** menu is set to **True**, the amended product lines are added the existing subscription in OSS.

14. Submit the transaction to get the required approvals and create the amendment order.

**Notes:**

- When a subscription is amended, the status of both the original subscription and the amended subscription is "Active" in OSS.
- Quality changes are supported at the subscription line level and not at the root model level.
- Sales users cannot amend a subscription with a future-dated termination date. The termination of a subscription is handled by OSS.
- Sales user can use the Update flag with all types of Amend flows to determine whether to apply subscription changes to a new subscription or the existing subscription.

## APPENDIX C: ADD AMENDED LINES TO EXISTING SUBSCRIPTION

Once a subscription is created, sales users can change the subscription and have the amended lines included in the existing subscription. As an example, a sales user subscribes to "Crossover SUV Subscription" in Oracle CPQ and later decides to upgrade to "Sports SUV Subscription". The sales user wants to include the amended lines in the existing subscription instead of creating a new subscription.

To add amended lines to an existing subscription, perform the following steps:

1. Open Oracle CPQ.
2. Navigate to Transaction Manager.
3. Click the Transaction Number associated with the subscription to amend. The Transaction page opens.
4. Amend the subscription by adding, updating, or deleting products.
5. Set the **UpdateOldSubscription** checkbox next to the amended products to True. The amended product lines are added to the existing subscription in OSS.

<input type="checkbox"/>	Grp Se...	Product #	Tiered	updateOldSu...	Action Code	Fulfillment St...	Change Code
<input type="checkbox"/>	1.0	SUV	N	True	Update	Created	
<input type="checkbox"/>	1.1	Sports SUV Subscription	N	True	Add		Full
<input type="checkbox"/>	1.2	Standard Maintenance Plan	N	True	Update		Full
<input checked="" type="checkbox"/>	1.3	Crossover SUV Subscription	N	True	Delete		Full

Subscription Changes Applied to Existing Subscription

### Notes:

- Administrators can also automate the process by setting the "Update" flag through the BOM configuration approach.
- The Populate OSS Charge action allows users to call OSS to get the update the charges for the pre-existing subscription lines from the original subscription contract that are getting amended in the current Transaction.

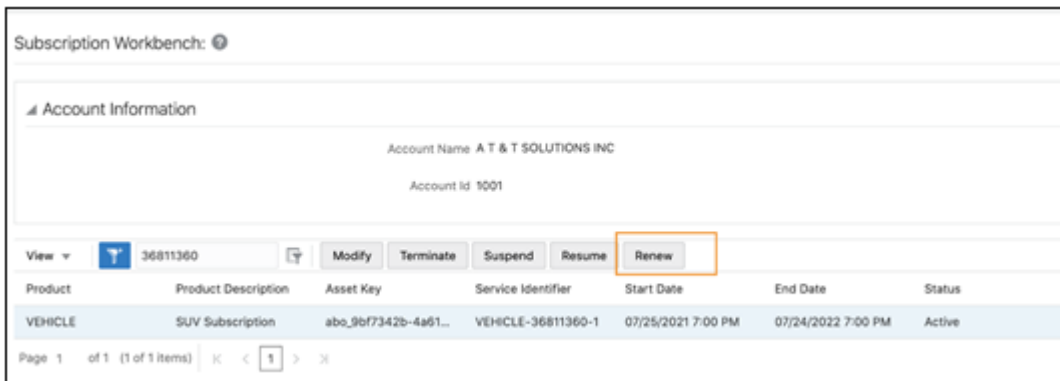
## APPENDIX D: RENEW SUBSCRIPTION WORKFLOW

OSS provides the ability to set the renewal for subscriptions that are managed with Oracle CPQ. When the OSS Renew event is triggered, a new Transaction is created in CPQ with an action code of Renew for all the Transaction Lines. Upon clicking the Submit Order button, a new subscription is created in OSS.

**Note:** If administrators define email notifications for renewals, the subscription owner receives an email notification informing of the subscription renewal.

To renew an active subscription directly from Oracle CPQ, perform the following steps:

1. In Oracle CPQ, create a new Transaction with the customer information updated.
2. Navigate to the Customer Assets page.
3. Select the asset that corresponds to the subscription to renew.



Customer Assets Page

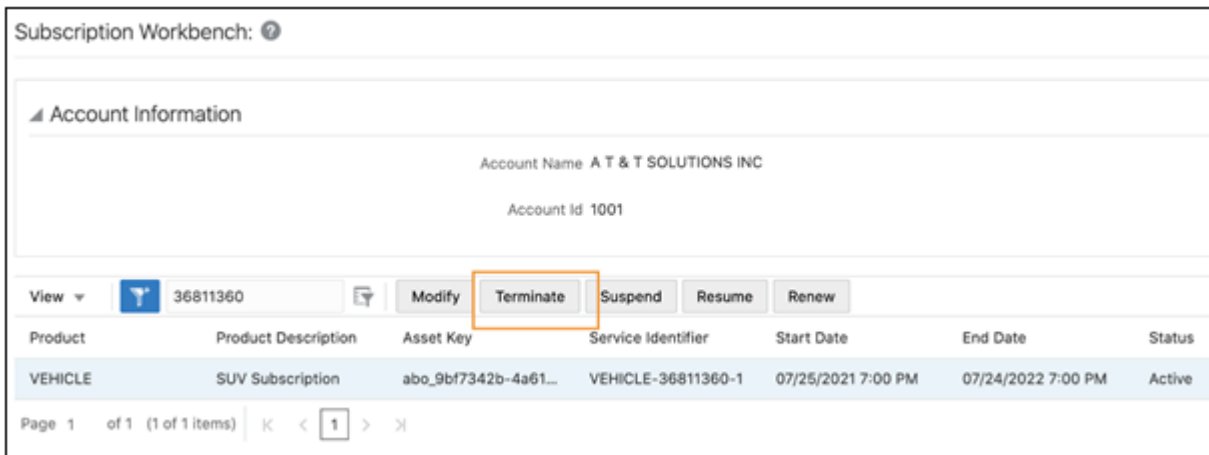
4. Click **Renew**. The Transaction page opens.
5. Click **Save**.
6. Click **Submit Order**.  
Use OSS to verify the renewal of the subscription.  
The product status of the subscription displays as "renewed" in OSS.

## APPENDIX E: TERMINATE SUBSCRIPTION WORKFLOW

Sales users can use the Customer Assets page to terminate subscriptions that are in an active state in OSS. Terminating an active subscription creates a Transaction Line in a new Transaction, where the root line item has an action code of Terminate and all products have an action code of Delete.

To terminate an active subscription, perform the following steps:

1. In Oracle CPQ, create a new Transaction with the customer information updated.
2. Navigate to the Customer Assets page.
3. Select the asset that corresponds to the subscription to terminate.



Customer Assets Page

4. Click **Terminate**. The Transaction page opens.
5. Click **Save**.
6. Click **Submit Order**.  
Use OSS to verify the termination of the subscription.  
The product status of the subscription displays as "closed" in OSS.

## APPENDIX F: COMMERCE ATTRIBUTES

The following table identifies new and existing Commerce attributes used by the Subscription Management solution. The attributes identified in the Description column as “existing” are already available in the Oracle Quote to Order Commerce process.

DOCUMENT	ATTRIBUTE NAME	ATTRIBUTE VARIABLE NAME	TYPE	DESCRIPTION
Transaction	SubscriptionEndDate	subscriptionEndDate_t	Date	This field is added to capture the subscription End date. The subscription Start date is mapped to the Subscription Ordering defaultRequestDate field.
Transaction	maxRequestDate	maxRequestDate_t	Text	A Timer is configured based on this field, which initially copies the content of defaultRequestDate field and is updated based on the Contract.
Transaction	Status	status_t	Menu	This is an existing menu used to store the state of a Transaction. A new menu item, Subscription Created (SUBSCRIPTION_CRATED) is added.
Transaction	Transaction Number	transactionID_t	Text	This is an existing field that is mapped to an OSS subscription number. The subscription number must be unique. The transaction number is appended with transaction Id.
Transaction	SubscriptionId	subscriptionId_t	Text	This field is added for debugging purposes. When subscriptions are successfully created, the field is updated with the new subscription ID.
Transaction	Subscription Status	subscription_Status_t	Text	



DOCUMENT	ATTRIBUTE NAME	ATTRIBUTE VARIABLE NAME	TYPE	DESCRIPTION
Transaction	RenewDraftSubscriptionNumber	renewDraftSubscriptionNumber_t	Text	This field is added for the Renew flow. In the last step of the Renew flow, the temporary subscription created by the OSS Renew event is deleted. The subscription number of the temporary subscription is stored in this variable, so the REST API call from OIC can delete the temporary subscription.
Transaction	SubscriptionProfileId	subscriptionProfileId_t		Holds the OSS subscription Profile Id used when a subscription is created by CPQ Cloud. The value is obtained from OSS.
Transaction	Buld	businessUnitId_t	Text	Fusion applications identify each customer using a Buld. The value is obtained from OSS.
Transaction	OSS Price Info	ossPriceInfo_t	Text	Holds the pricing charges coming from OSS for all the subscription items.
Transaction->Customer Detail	Billing Frequency	oRCL_billingFrequency_t	Menu	Determines the billing frequency for a subscription. The billing frequency option can be obtained using the following REST API call: <a href="https://fuscdrmsmc225-fa-ext.us.oracle.com/crmRestApi/resources/latest/timeCodeUnits">https://fuscdrmsmc225-fa-ext.us.oracle.com/crmRestApi/resources/latest/timeCodeUnits</a> . Conversion rate multiplied by BaseUOMCode provides UserUOMCode.
Transaction->Customer Detail	Customer Company Name	CustomerCompanyName_t	Text	When the Customer Company Name field is populated, Customer Details, partyId, accountId, and BillToSiteUsed are populated for the entered company.
Transaction->Customer Detail	Account Number	accountNumber_t	Text	When the Customer Detail button on the Customer Detail tab is clicked, the customer account number is retrieved based on the entered customer company name.

DOCUMENT	ATTRIBUTE NAME	ATTRIBUTE VARIABLE NAME	TYPE	DESCRIPTION
Transaction->Customer Detail	Bill To Site Use Id	billToSiteUseId_t	Text	When the Customer Detail button on the Customer Detail tab is clicked, the Bill To Site Use Id is retrieved based on the entered customer company name.
Transaction Line Attribute	Party Id	partyId_t	Text	The CDM integration provides the value for this field based on the Customer Company Name
Transaction Line Attribute	Contract Start Date	contractStartDate_l	Date	This is an existing Line Item Grid attribute in the Ref App, modified to have a default current date.
Transaction Line Attribute	Quantity	requestedQuantity_l	Integer	This is an existing field modified to make the field read-only for parts and marking lines (Product and Charge) as mandatory in the BOM definition
Transaction Line Attribute	Change Reason	changeReason_l	Menu	A single select menu to handle all the possible reasons for closing a subscription. Possible values (variable names in parentheses): <ul style="list-style-type: none"> <li>• Breach (ORA_BREACH)</li> <li>• Non Compliant (ORA_NON_COMPLIANCE)</li> <li>• Quantity Change (ORA_CHANGE_QUANTITY)</li> <li>• Downgrade (ORA_DOWNGRADE)</li> <li>• Term Change (ORA_TERM_CHANGE)</li> <li>• Upgrade (ORA_UPGRADE)</li> </ul>
Transaction Line Attribute	Change Code	changeCode_l	Menu	A single select menu to handle all the possible codes to close a subscription. Possible values (variable names in parentheses): <ul style="list-style-type: none"> <li>• Full (ORA_FULL)</li> <li>• Prorate Without Credit (ORA_PRORATE_WITHOUT_CREDIT)</li> <li>• Prorate With Credit (ORA_PRORATE_WITH_CREDIT)</li> </ul>

DOCUMENT	ATTRIBUTE NAME	ATTRIBUTE VARIABLE NAME	TYPE	DESCRIPTION
Transaction Line Attribute	Amend Replacement	amendReplacement_I	Text	This field is used to capture the instance Id for the product getting replaced. This value is used for constructing a relationship in OIC.
Transaction Line Attribute	Tier Sequence	oRCL_pRC_tierInfo.oRCL_pRC_tierSequence	Integer	Used to capture the sequence of the tier with up to a maximum of three tiers. When the tier is single, this holds the single tier.
Transaction Line Attribute	Tier From	oRCL_pRC_tierInfo.oRCL_pRC_tierFrom	Integer	The starting count of tier 1.
Transaction Line Attribute	Tier To	oRCL_pRC_tierInfo.oRCL_pRC_tierTo	Integer	The ending count of tier 1.
Transaction Line Attribute	Tier List Price	oRCL_pRC_tierInfo.oRCL_pRC_tierListPrice	Currency	The list price for the tier.
Transaction Line Attribute	Tier Price Format	oRCL_pRC_tierInfo.oRCL_pRC_tierPriceFormat	Menu	The menu for selecting the price format for the tier. Value sets include: <ul style="list-style-type: none"> <li>Per Unit(ORA_PER_UNIT)</li> <li>Per Block(ORA_PER_BLOCK)</li> </ul>
Transaction Line Attribute	Tier Block Size 1	oRCL_pRC_tierInfo.oRCL_pRC_tierBlockSize	Integer	The block size for tier 1.
Transaction Line Attribute	Amended Replacement Product	amendReplacementProduct	HTML	Displays the list of products that are getting replaced as part of an upgrade or downgrade.
Transaction Line Attribute	Period From	oRCL_chg_periodFrom_I	Integer	Used when Adjustment Effectivity is ORA_SPECIFIC_PERIODS
Transaction Line Attribute	Period To	oRCL_chg_periodTo_I	Integer	Used when Adjustment Effectivity is ORA_SPECIFIC_PERIODS

DOCUMENT	ATTRIBUTE NAME	ATTRIBUTE VARIABLE NAME	TYPE	DESCRIPTION
Transaction Line Attribute	Contract End Date	contractEndDate_I	Date	The date the customer stops receiving the service.
Transaction Line Attribute	External Parent Key	oRCL_pRC_externalParentKey	Text	
Transaction Line Attribute	PriceFormat	oRCL_cHG_priceFormat_I	Menu	Values set as follows: <ul style="list-style-type: none"> <li>PER UNIT (ORA_PER_UNIT)</li> <li>PER BLOCK (ORA_PER_BLOCK)</li> </ul>
Transaction Line Attribute	Discount Effectivity Type	discountEffectivityType_I	Menu	This field holds information about when to apply the adjustment. <ul style="list-style-type: none"> <li>All Term (ORA_ALL_TERM)</li> </ul>
Transaction Line Attribute	Usage Value	usageValue_I	Float	This attribute acts as a quantity for parts of type Usage.
Transaction Line Attribute	Usage Value UoM	usageValueUOM_I	Text	Unit of measure for Usage Value.

## APPENDIX G: UPDATE ASSET TIMER BML

The Subscription Created workflow step triggers an Update Asset Timer when the Max Request Date is reached. The BML for the Update Asset Timer is used to convert an asset and all of its Transaction Lines, or the selected Transaction Lines, to an updated asset.

```
//MainDoc UpdateAsset Action Script
//
//Purpose: Helper to create asset for all or selected lines in the transaction or specified
external lines
//System Variables : _system_buyside_id
//Main doc fields: _transaction_customer_id(_transaction prefix is from maindoc varname) //
(process specific): currency_t, paymentTerms_t
//Line fields:_document_number, requestDate_l, itemInstanceId_l, //oRCL_ABO_ActionCode_l,
_line_bom_parent_id, fulfillmentStatus_l

//step1 common transaction heading initialization customer_id = _transaction_customer_id;
currency = currency_t; paymentTerm = paymentTerms_t;

//abo main logic
FULFILLED = "FULFILLED";
DOCUMENT_NUMBER = "documentNumber";
CUSTOMER = "customer";
TRANSACTION_ID = "transactionId";
CURRENCY_CODE = "currency";
REQUEST_DATE = "requestDate";
ACTION_CODE = "actionCode";
ITEM_INSTANCE_ID = "itemInstanceId";
PAYMENT_TERM = "paymentTerms";
//we won't allow asset creation until you select a customer.
if(customer_id == "" OR isnull(customer_id)){ throwError("Please select an customer.");
}
// since some transaction attribute value are needed to update asset
// we will also collect some transaction info and for internal , we will place them into 2
level hierarchy json txn_json = json(); jsonput(txn_json, CUSTOMER, customer_id);
jsonput(txn_json, CURRENCY_CODE, currency); jsonput(txn_json, PAYMENT_TERM, paymentTerm);
jsonput(txn_json, TRANSACTION_ID, _system_buyside_id);

//processing for internal case.
//now we collect the list of lines need to update-asset
currentDateFmt=strtojavodate(getstrdate(),"MM/dd/yyyy"); successString =""; lineJsonArray =
jsonArray(); for line in transactionLine{ if(line.fulfillmentStatus_l == "FULFILLED" OR
line.fulfillmentStatus_l == "CANCELLED"){ continue;//skip lines already fulfilled or
cancelled.
} if(line._line_bom_parent_id <>"") { //skip non-root line continue;
} if(line.itemInstanceId_l=="") { //skip non abo & non model line

continue;
} contractStartDatFmt=strtojavodate(line.contractStartDate_l,"MM/dd/yyyy");
if (comparedates(contractStartDatFmt,currentDateFmt)==1) { continue;
} lineJson = json();
jsonput(lineJson, DOCUMENT_NUMBER, line._document_number);
//for transaction date we will use db format within abo script, // also for empty date
we treat as today as of processing time transactionDate = line.requestDate_l;
if(transactionDate == "" OR isnull(transactionDate)){ transactionDate =
datetostr(getDate(false), "yyyy-MM-dd HH:mm:ss");
}else{ tranDate = strtojavodate(transactionDate, "MM/dd/yyyy HH:mm:ss");
transactionDate = datetostr(tranDate, "yyyy-MM-dd HH:mm:ss");
```

```

    }    jsonput(lineJson, REQUEST_DATE, transactionDate);    jsonput(lineJson, ACTION_CODE,
line.orcl_abo_actionCode_1);    jsonput(lineJson, ITEM_INSTANCE_ID, line.itemInstanceId_1);
jsonarrayappend(linejsonArray, lineJson);

    //also prepare the return string to update root lines when the updateAsset is successful
if (successString<> ""){    successString = successString + "|";
    }    successString = successString + line._document_number +
"~fulfillmentStatus_1~"+FULFILLED; }

//now invoke utility to load line detail,transfer to bom, and aggregate open order, and
generate delta action and invoke asset syc
//if updateAsset fail, expect the abo_updateAsset to throwerror from inside response =
util._ORCL_ABO.abo_updateAsset(txn_json, lineJsonArray); if (successString<> ""){
successString = successString + "|";
}
    return successString;
//MainDoc UpdateAsset Action Script
//
//Purpose: Helper to create asset for all or selected lines in the transaction or specified
external lines
//System Variables : _system_buyside_id
//Main doc fields: _transaction_customer_id(transaction prefix is from maindco varname) //
(process specific): currency_t, paymentTerms_t
//Line fields:_document_number, requestDate_1, itemInstanceId_1, //orcl_abo_actionCode_1,
_line_bom_parent_id, fulfillmentStatus_1

//step1 common transaction heading initialization customer_id = _transaction_customer_id;
currency = currency_t; paymentTerm = paymentTerms_t;

//abo main logic
FULFILLED = "FULFILLED";
DOCUMENT_NUMBER = "documentNumber";
CUSTOMER = "customer";
TRANSACTION_ID = "transactionId";
CURRENCY_CODE = "currency";
REQUEST_DATE = "requestDate";
ACTION_CODE = "actionCode";
ITEM_INSTANCE_ID = "itemInstanceId";
PAYMENT_TERM = "paymentTerms";
//we won't allow asset creation until you select a customer.
if(customer_id == "" OR isnull(customer_id)){    throwError("Please select a customer."); }
// since some transaction attribute value are needed to update asset
// we will also collect some transaction info and for internal , we will place them into 2
level hierarchy json txn_json = json(); jsonput(txn_json, CUSTOMER, customer_id);
jsonput(txn_json, CURRENCY_CODE, currency); jsonput(txn_json, PAYMENT_TERM, paymentTerm);
jsonput(txn_json, TRANSACTION_ID, _system_buyside_id);

//processing for internal case.
//now we collect the list of lines need to update-asset
currentDateFmt=strtojavodate(getstrdate(),"MM/dd/yyyy"); successString =""; lineJsonArray =
jsonArray(); for line in transactionLine{    if(line.fulfillmentStatus_1 == "FULFILLED" OR
line.fulfillmentStatus_1 == "CANCELLED"){    continue;//skip lines already fulfilled or
cancelled.
    }    if(line._line_bom_parent_id <>"") { //skip non-root line    continue;
    }    if(line.itemInstanceId_1=="") { //skip non abo & non model line    continue;
    }    contractStartDatFmt=strtojavodate(line.contractStartDate_1,"MM/dd/yyyy");
    if (comparedates(contractStartDatFmt,currentDateFmt)==1) {    continue;
    }
    lineJson = json();

```

```

    jsonput(lineJson, DOCUMENT_NUMBER, line._document_number);
    //for transaction date we will use db format within abo script, // also for empty date
we treat as today as of processing time transactionDate = line.requestDate_1;
if(transactionDate == "" OR isnull(transactionDate)){ transactionDate =
datetostr(getDate(false), "yyyy-MM-dd HH:mm:ss");
} else{ tranDate = strtodate(transactionDate, "MM/dd/yyyy HH:mm:ss");
transactionDate = datetostr(tranDate, "yyyy-MM-dd HH:mm:ss"); }
    jsonput(lineJson, REQUEST_DATE, transactionDate); jsonput(lineJson, ACTION_CODE,
line.orcl_abo_actioncode_1); jsonput(lineJson, ITEM_INSTANCE_ID, line.itemInstanceId_1);
jsonarrayappend(linejsonArray, lineJson);

    //also prepare the return string to update root lines when the updateAsset is successful
if (successString<> ""){ successString = successString + "|";
} successString = successString + line._document_number +
"~fulfillmentStatus_1~"+FULFILLED; }

//now invoke utility to load line detail, transfer to bom, and aggregate open order, and
generate delta action and invoke asset sync
//if updateAsset fail, expect the abo_updateAsset to throw error from inside response =
util._ORCL_ABO.abo_updateAsset(txn_json, lineJsonArray); if (successString<> ""){
successString = successString + "|";
}
    return successString;

```

## APPENDIX H: CUSTOMER DETAILS BML

The BML for the Customer Details action is used to support account integration. When sales users enter a customer company name and click Customer Details, the BML retrieves the PrimaryPartyId, BillToAccountId, and BillToSiteUseId fields from the Oracle EBS Customer Data Management application.

```
//1. Get Template Location system="TCA-OrgService"; operation="FindOrg";
organizationSoapRequestLocation = commerce.getTemplateLocation(system, operation);
//payload = commerce.getUserAttributes(system); defaultMessage = "";
organizationSoapRequest=applytemplate(organizationSoapRequestLocation ,dict("string"),
defaultErrorMessage);
organizationSoapResponse = commerce.invokeWebService(system, organizationSoapRequest );
errorString = "Error in TCA Service"; xpaths = string[1]; xpaths[0] = "//ns2:PartyId";
returnPartyId = ""; output = readxmlsingle(organizationSoapResponse, xpaths); if
(containskey(output,xpaths[0])) { returnPartyId = get(output,xpaths[0]); } else
{
    returnPartyId = "Check if customer company name is valid & also TCA
service is up."; returnSiteNumber = "Check if customer company name is valid & also
TCA service is up" ; return
"1~partyId_t~"+returnPartyId+"|"+1~billToSiteUseId_t~"+returnSiteNumber+"|"; }

// Get Template Location system="TCA-AccService"; operation="FindAcc";
customerAccountSoapRequestLocation = commerce.getTemplateLocation(system, operation);
//payload1 = commerce.getUserAttributes(system,returnPartyId); payload1 = dict("string");
put(payload1,"returnPartyId",returnPartyId);
customerAccountSoapRequest=applytemplate(customerAccountSoapRequestLocation,payload1,
defaultErrorMessage);
//print customerAccountSoapRequest;
customerAccountsoapResponse = commerce.invokeWebService("TCA-AccService",
customerAccountSoapRequest);
//print customerAccountsoapResponse ;
xpathsAcct = string[1]; xpathsAcct[0] = "//ns2:CustomerAccountId"; returnAccountNumber =
"" ; outputAcct = readxmlsingle(customerAccountsoapResponse, xpathsAcct); if
(containskey(outputAcct,xpathsAcct[0]))
    { returnAccountNumber = get(outputAcct,xpathsAcct[0]); } else {
        returnAccountNumber = "Check if customer company name is valid & also
TCA service is up"; returnSiteNumber = "Check if customer company name is valid & also
TCA service is up" ; return
"1~accountNumber_t~"+returnAccountNumber+"|"+1~billToSiteUseId_t~"+returnSiteNumber
+"|"; } returnBillToSiteUseId=""; xpathsbilltositeuseid = string[1];
xpathsbilltositeuseid[0] =
"//ns2:Value/ns2:CustomerAccountSite/ns2:CustomerAccountSiteUse[ns2:SiteUseCode='BILL_TO'
and ns2:PrimaryFlag='true']/ns2:SiteUseId"; outputbilltositeuseid =
readxmlsingle(customerAccountsoapResponse, xpathsbilltositeuseid); if
(containskey(outputbilltositeuseid ,xpathsbilltositeuseid[0])) {
returnBillToSiteUseId= get(outputbilltositeuseid,xpathsbilltositeuseid[0]); } else {
returnBillToSiteUseId= "Check if customer company name is valid & also TCA service is up.
";

} xpathCustomerAccountSiteUse = string[1];
// get the XML Element called Customer AccountSiteUse where primary is true and use is BILL
TO xpathCustomerAccountSiteUse[0] =
"//ns2:Value/ns2:CustomerAccountSite/ns2:CustomerAccountSiteUse[ns2:SiteUseCode='BILL_TO'
and ns2:PrimaryFlag='true']";
outputCustomerAccountSiteUse = readxmlsingle(customerAccountsoapResponse,
xpathCustomerAccountSiteUse); returnSiteId =""; if
(containsKey(outputCustomerAccountSiteUse,xpathCustomerAccountSiteUse[0])) {
    CustomerAccountSiteUseXmlFragment =
get(outputCustomerAccountSiteUse,xpathCustomerAccountSiteUse[0]); xpath1 = string[1];
xpath1[0] = "//ns2:CustomerAccountSiteId"; output1 =
readxmlsingle(CustomerAccountSiteUseXmlFragment,xpath1); if
(containsKey(output1,xpath1[0]))
```



```

    {
        returnSiteId = get(output1, xpath1[0]);
    }
    } if (returnSiteId == "") { returnSiteId = errorString;
} partySiteId = ""; xpathCustomerAccountSite = string[1];
// get the XML Element called Customer AccountSite where ID = returnSiteId
xpathCustomerAccountSite[0] =
"//ns2:Value/ns2:CustomerAccountSite[ns2:CustomerAccountSiteId=" + returnSiteId + "];

    outputCustomerAccountSite = readxmlsingle(customerAccountsoapResponse,
xpathCustomerAccountSite);
    if (containsKey(outputCustomerAccountSite, xpathCustomerAccountSite[0]))
    {
        CustomerAccountSiteXmlFragment =
get(outputCustomerAccountSite, xpathCustomerAccountSite[0]);
        xpath2 = string[1];    xpath2[0] = "//ns2:PartySiteId";    output2 =
readxmlsingle(CustomerAccountSiteXmlFragment, xpath2);    if
(containsKey(output2, xpath2[0]))
        {
            partySiteId = get(output2, xpath2[0]);
        } if (partySiteId == "") { partySiteId = errorString;
        }
    } partySiteNumber = ""; xpathPartySiteNumber = string[1];
// get the XML Element called Customer AccountSite where ID = returnSiteId
xpathPartySiteNumber[0] = "//ns2:Value/ns2:PartySite[ns1:PartySiteId=" + partySiteId + "];
outputPartySiteNumber = readxmlsingle(organizationSoapResponse, xpathPartySiteNumber);
if (containsKey(outputPartySiteNumber, xpathPartySiteNumber[0])) {
    PartySiteNumberXmlFragment = get(outputPartySiteNumber, xpathPartySiteNumber[0]);
    xpath3 = string[1];    xpath3[0] = "//ns1:PartySiteNumber";    output3 =
readxmlsingle(PartySiteNumberXmlFragment, xpath3);    if (containsKey(output3, xpath3[0]))
    {
        partySiteNumber = get(output3, xpath3[0]);
    }
} return
"1~partyId_t~"+returnPartyId+"|"+ "1~accountNumber_t~"+returnAccountNumber+"|"+ "1~billToSite
UseId_t~"
+returnBillToSiteUseId +"|"+ "1~customerID_t~"+returnPartyId
+"|"+ "1~_customer_id~"+returnPartyId+"|";
    outputCustomerAccountSite = readxmlsingle(customerAccountsoapResponse,
xpathCustomerAccountSite);
    if (containsKey(outputCustomerAccountSite, xpathCustomerAccountSite[0]))
    {
        CustomerAccountSiteXmlFragment =
get(outputCustomerAccountSite, xpathCustomerAccountSite[0]);
        xpath2 = string[1];    xpath2[0] = "//ns2:PartySiteId";    output2 =
readxmlsingle(CustomerAccountSiteXmlFragment, xpath2);    if
(containsKey(output2, xpath2[0]))
        {
            partySiteId = get(output2, xpath2[0]);
        } if (partySiteId == "") { partySiteId = errorString;
        }
    } partySiteNumber = ""; xpathPartySiteNumber = string[1];
// get the XML Element called Customer AccountSite where ID = returnSiteId
xpathPartySiteNumber[0] = "//ns2:Value/ns2:PartySite[ns1:PartySiteId=" + partySiteId + "];
outputPartySiteNumber = readxmlsingle(organizationSoapResponse, xpathPartySiteNumber);
if (containsKey(outputPartySiteNumber, xpathPartySiteNumber[0])) {
    PartySiteNumberXmlFragment = get(outputPartySiteNumber, xpathPartySiteNumber[0]);
    xpath3 = string[1];    xpath3[0] = "//ns1:PartySiteNumber";    output3 =
readxmlsingle(PartySiteNumberXmlFragment, xpath3);    if (containsKey(output3, xpath3[0]))
    {
        partySiteNumber = get(output3, xpath3[0]);
    }
}
} print "returnPartyId: " + returnPartyId; print "returnAccountNumber : " +
returnAccountNumber ; print "returnSiteId : " + returnSiteId ; print "partySiteId : " +

```

```
partySiteId; print "billToSiteUseId"+ returnBillToSiteUseId ; print "partySiteNumber : " +
partySiteNumber ;

return
"1~partyId_t~"+returnPartyId+"|"+"1~accountNumber_t~"+returnAccountNumber+"|"+"1~billToSite
UseId_t~" +returnBillToSiteUseId +"|" ;
```

## APPENDIX I: OPEN TRANSACTION LINE BML

The BML for the Open Transaction Line action is used to display the details for a specific Transaction Line.

```
returnVal = "";
lineHierInfo = commerce.oRCL_abo_BuildLineItemHierarchy();
returnVal = returnVal + commerce.oRCL_abo_PostDefaultsOnLineItems(lineHierInfo);
returnVal = returnVal+ commerce.oRCL_sm_postDefaultOnLineItem(lineHierInfo);

return returnVal;
```

## APPENDIX J: SAVE BML

### Transaction - Save

The BML for the Save action is used to save the current state of a Transaction.

Below BML is used for Advanced Modify - Before Formulas

```
contractPeriodsStr = stringBuilder();
for line in transactionLine {
    sbappend(contractPeriodsStr, line._document_number, "~contractedPeriods_1~",
        string(commerce.oRCL_sm_calculateContractPeriods(line.contractStartDate_1,
            line.contractEndDate_1, line.priceType_1, line.pricePeriod_1)), "|");
}
return sbtoString(contractPeriodsStr);
```

Below BML is used for Advanced Modify – After Formulas

```
return commerce.calculateRollupRevenues();
```

### Transaction Line – Save

The BML for the Save action is used to save the current state of a Transaction Line.

Below BML is used for Advanced Modify – Before Formulas.

```
//At TransactionLine level save
//System Variable Name      Type      Description
//_system_current_document_number String  Current Document Number

//Variable Name for (Transaction Line)      Type      Description
//transactionLine Collection of Sub Documents
//      _document_number String  Document Number
//      _price_calculation_info String  Calculation Information
//      oRCL_pRC_tierd_1 String  Tiered

//Imported Util Functions
//String _SM.oRCL_pRC_populateCharges(JsonArray charges, String documentNumber)

returnString = "";
for line in transactionLine {
    if(line._document_number == _system_current_document_number){
        if(line.oRCL_pRC_tierd_1 == "N" AND line._price_calculation_info <> "") {
            calcInfo = jsonArrayget(jsonarray(line._price_calculation_info),0,"json");
            charges = jsonget(calcInfo,"charges", "jsonArray");
            returnString = returnString + util._SM.oRCL_pRC_populateCharges(charges,
line._document_number);
        }
    }
}
return returnString;
```

## APPENDIX K: PAYLOAD TEMPLATE FILE CONTENT

The BML associated with the payload template files referenced in [Add Template Dependencies to File Manager](#) follows.

### *findOrganizationPayload.txt*

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body
xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/parties/organizationService/applicationModule/types/">
    <ns1:findOrganization>
      <ns1:findCriteria xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
        <ns2:filter>
          <ns2:conjunction>And</ns2:conjunction>
          <ns2:group>
            <ns2:conjunction>And</ns2:conjunction>
            <ns2:item>
              <ns2:conjunction>And</ns2:conjunction>
              <ns2:attribute>PartyName</ns2:attribute>
              <ns2:operator>=</ns2:operator>
              <ns2:value>{{customerCompanyName_t}}</ns2:value>
            </ns2:item>
          </ns2:group>
        </ns2:filter>
      </ns1:findCriteria>
      <ns1:findControl xmlns:ns3="http://xmlns.oracle.com/adf/svc/types/">
        <ns3:retrieveAllTranslations></ns3:retrieveAllTranslations>
      </ns1:findControl>
    </ns1:findOrganization>
  </soap:Body>
</soap:Envelope>
```

### *customerAccountPayload.txt*

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ns1="http://xmlns.oracle.com/apps/cdm/foundation/parties/customerAccountService/applicationModule/types/" xmlns:ns2="http://xmlns.oracle.com/adf/svc/types/">
  <soapenv:Body>
    <ns1:findCustomerAccount>
      <ns1:findCriteria>
        <ns2:filter>
          <ns2:conjunction>And</ns2:conjunction>
          <ns2:group>
            <ns2:conjunction>And</ns2:conjunction>
            <ns2:item>
              <ns2:conjunction>And</ns2:conjunction>
              <ns2:attribute>PartyId</ns2:attribute>
              <ns2:operator>=</ns2:operator>
              <ns2:value>{{returnPartyId}}</ns2:value>
            </ns2:item>
          </ns2:group>
        </ns2:filter>
      </ns1:findCriteria>
      <ns1:findControl>
        <ns2:retrieveAllTranslations>>false</ns2:retrieveAllTranslations>
      </ns1:findControl>
    </ns1:findCustomerAccount>
  </soapenv:Body>
</soapenv:Envelope>
```

## APPENDIX L: LIBRARY FUNCTION BML

The Oracle CPQ Subscription Management package adds several library functions to the Commerce process. The BML associated with each of the library functions is provided below.

### ***String getTemplateLocation(String system, String operation)***

The code for this library function is provided below for reference.

```
//1. Get Template File
templateUrl = "";
//bmq1 query
resultSet = bmql("Select Template from INT_SYSTEM_TEMPLATES where System = $system and
Operation = $operation");

//loop through the records
for record in resultSet {
    templateUrl = get(record,"Template");
    print templateUrl;
}

temp=split(templateUrl,"image");

return temp[1];
```

### ***String Dictionary getUserAttributes(String system)***

The code for this library function is provided below for reference.

```
//Name: getUserAttributes
//Variable Name: getUserAttributes
//Description: Queries UserName,Password from DataTable INT_SYSTEM_DETAILS and adds it in
the payload.
//Input: Main Doc Attr: orderId
//System Attr: _system_user_first_name
//Return Type: String Dictionary
//Dependency : getPassword
//bmq1 query
resultSet = bmql("Select Username from INT_SYSTEM_DETAILS where System = $system");
payload = dict("string");
//loop through the records
for record in resultSet {
    userName = get(record,"Username");
    password = "Welcome1";
    put(payload, "USERNAME", userName);
    put(payload, "PASSWORD", password);
}

//2. Set the UTC time in header

utcFormatDate = datetostr(getdate(),"yyyy-MM-dd'T'HH:mm:ss.SSS'Z'", "UTC");
put(payload, "UTCTIME", utcFormatDate);

//3. Useful only for Replacing DOO Template
curDate = datetostr(getdate(),"dd-MM-yyyy");
msgId = "BM_ORDER_"+orderId+"_SubmittedBy"+_system_user_first_name+"_ON_"+curDate;

put(payload, "MSGID", msgId);

//4. Return dictionary
return payload;
```

### ***String invokeWebService(String system, String soapReq)***

The code for this library function is provided below for reference.

```
//1. Get webservice endpoint for the system
resultSet = bmql("Select Endpoint,Username,Password from INT_SYSTEM_DETAILS where System =
$system");
endpoint = "";
username = "";
password = "";

//loop through the records
for record in resultSet {
    endpoint = get(record,"Endpoint");
    username = get(record,"Username");
    password = get(record,"Password");
}
//2. Invoke the web service
headerValues = dict("string");
put(headerValues, "Content-Type", "text/xml; charset=utf-8");
encodeCredential = encodebase64(username+":"+password);
auth = "Basic " + encodeCredential;
put(headerValues,"Authorization",auth);

errorString = "Error in "+system+" invocation";

soapResponse= urldatabypost(endPoint , soapReq,errorString,headerValues,true); // sends the
soap call and returns response to variable.
print "going to print soapResponse";
//3. Return the response
return soapResponse;
```

## String Populate Discounts

The code for this library function is provided below for reference.

```
discStDtEndDtFormat = "yyyy-MM-dd";
contractStartDate = strtodate(contractStartDateStr, "yyyy-MM-dd");
returnStrBuilder = stringBuilder("");
adjustmentList = bmql("select Start_Date, End_Date, Discount_Name, Discount_Value,
Discount_Type, Discount_Reason, Discount_Type, Discount_Effectivity, Effectivity_Periods
from ORCL_PRC_DISCOUNTS where Product=$partNumber ORDER BY Start_Date, End_Date");
for adjustment in adjustmentList {
    discFound = false;
    discStDtStr = get(adjustment,"Start_Date");
    discEndDtStr = get(adjustment,"End_Date");

    if((isnull(discStDtStr) or trim(discStDtStr) == "") and (isnull(discEndDtStr) or
trim(discEndDtStr) == "")) {
        discFound = true;
    } elif (trim(discStDtStr) <> "" and (isnull(discEndDtStr) or trim(discEndDtStr) == "")) {

        discStDt = strtodate(get(adjustment,"Start_Date"), discStDtEndDtFormat);
        if(comparedates(discStDt, contractStartDate) <= 0) {
            discFound = true;
        }
    } elif ((isnull(discStDtStr) or trim(discStDtStr) == "") and trim(discEndDtStr) <> "") {
        discEndDt = strtodate(get(adjustment,"End_Date"), discStDtEndDtFormat);
        if(comparedates(discEndDt, contractStartDate) >= 0) {
            discFound = true;
        }
    } else {
        discStDt = strtodate(get(adjustment,"Start_Date"), discStDtEndDtFormat);
        discEndDt = strtodate(get(adjustment,"End_Date"), discStDtEndDtFormat);
        if(comparedates(discStDt, contractStartDate) <= 0 and comparedates(discEndDt,
contractStartDate) >= 0) {
            discFound = true;
        }
    }

    if(discFound) {
        sbappend(returnStrBuilder, "|", lineDocNumber,
"~customDiscountType_1~",get(adjustment,"Discount_Type"));
        sbappend(returnStrBuilder, "|", lineDocNumber,
"~customDiscountValue_1~",get(adjustment,"Discount_Value"));
        sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityType_1~",get(adjustment,"Discount_Effectivity"));
        break;
    }
}
return sbtoString(returnStrBuilder);
```

## String Populate Amend Charge

The code for this library function is provided below for reference.

```
//Name:Populate Amend Charge
//Variable Name:populateAmendCharge
//Description:Populates charge information directly from OSS for AMEND and RENEW flow.
//Return Type:String
```



```

//define constants for action codes noUpdateCode = "NO_UPDATE"; updateCode = "UPDATE";
deleteCode = "DELETE"; addCode = "ADD"; actionCode = "";
ret = ""; chargeMap = dict("string"); for line in transactionLine{      ossStr = "";
if(NOT(len(line._line_bom_parent_id)>0) AND len(line.itemInstanceId_1)>0){
actionCode = line.oRCL_ABO_ActionCode_1;      chargeMap =
commerce.getOldSubscriptionChargeMap();
} else {
    actionCode_1 = line.oRCL_ABO_ActionCode_1;      if((actionCode == noUpdateCode
OR actionCode == updateCode) AND NOT(actionCode_1 == addCode)){
thisJsonStr = get(chargeMap,line.itemInstanceId_1);      thisJson =
json(thisjsonStr);      chargeArray = jsonarray( jsonget(thisJson,"charges") );
subsProdId = jsonget(thisJson,"SubscriptionProductPuid");      charges =
string[jsonarraysize(chargeArray)];      i = 0;      ossArray = jsonarray();
for chrg in charges{      chargeJson = json();      chrgJson =
jsonarrayget(chargeArray,i,"json");      i = i + 1;
jsonput(chargeJson,"chargeName",jsonget(chrgJson,"ChargeName"));
jsonput(chargeJson,"unitPrice",jsonget(chrgJson,"UnitListPrice","float"));
jsonput(chargeJson,"chargeType",jsonget(chrgJson,"PriceType"));
jsonput(chargeJson,"periodicity",jsonget(chrgJson,"PricePeriodicity"));
if(NOT(isnull(jsonget(chrgJson,"Allowance"))){
jsonput(chargeJson,"Allowance",jsonget(chrgJson,"Allowance","integer"));
}      if(NOT(isnull(jsonget(chrgJson,"BlockSize"))){
jsonput(chargeJson,"BlockSize",jsonget(chrgJson,"BlockSize","integer"));
}
jsonput(chargeJson,"UsagePriceLockFlag",jsonget(chrgJson
,"UsagePriceLockFlag","boolean"));      chargePuid = jsonget(chrgJson
,"ChargePuid");      if(jsonget(chrgJson,"TieredFlag")==="true"){
jsonput(chargeJson,"tiered", "Y");
jsonput(chargeJson,"tierType",jsonget(chrgJson,"TierType"));      urlParam
= subsProdId+"/child/charges/"+chargePuid+"?expand=all";      itemStr =
util.invokeOss(json(),"GET", "OSS-Amend", urlParam, false);      item =
json(itemStr);      tierArray = jsonget(item,"chargeTiers","jsonarray");
tier = string[jsonarraysize(tierArray)];      j = 0;
tiers = jsonarray();      for each in tier {      oneTier
= jsonarrayget(tierArray,j,"json");      tierJson = json();
j = j + 1;
jsonput(tierJson,"BlockSize",jsonget(oneTier,"BlockSize","integer"));
jsonput(tierJson,"TierTo",jsonget(oneTier,"TierTo","integer"));
jsonput(tierJson,"TierFrom",jsonget(oneTier,"TierFrom","integer"));
jsonput(tierJson,"ListPrice",jsonget(oneTier,"ListPrice","float"));
jsonput(tierJson,"PriceFormat",jsonget(oneTier,"PriceFormat"));

jsonput(tierJson,"SequenceNumber",jsonget(oneTier,"SequenceNumber","integer"));
jsonarrayappend(tiers,tierJson);
}      jsonput(chargeJson,"tierList", tiers);
}      jsonarrayappend(ossArray,chargeJson);
}      charge = json();      jsonput(charge,"charges",ossArray);
ossChargeArray = jsonarray();      jsonarrayappend(ossChargeArray,charge);
ossStr = jsonarraytostr(ossChargeArray);      if(len(ret)>0){
ret = ret+"|";
}      ret = ret + line._document_number+"~oSSCharge~"+ossStr;
}
}
} return ret;

```

### **Integer Calculate Contract Periods**

This is a Commerce library function that calculates number of periods for given contract start date and end date based on price type and periodicity. The code for this library function is provided below for reference.

```

monthsToAdd = 1; //months counter for monthly and annual parts.
//return 1 for all invalid use cases
if(priceType <> "Recurring" AND priceType <> "Usage"){
    return 1;
}else {
    if(periodicity == "Per Month"){
        monthsToAdd = 1;
    }elseif(periodicity == "Per Year"){
        monthsToAdd = 12;
    }else{
        return 1;
    }
}
returnPeriods = 1;
if(contractStartDate <> "" AND isnull(contractStartDate) <> true AND contractEndDate <> ""
AND isnull(contractEndDate) <> true){
    startDate = strtodate(contractStartDate , "MM/dd/yyyy");
    endDate = strtodate(contractEndDate , "MM/dd/yyyy");
    periodInDays = getdiffindays(startDate, endDate);
    count = Integer[(periodInDays / (30 * monthsToAdd))+1];
    //Keep adding months to contract start date till we reach contract end date. Number of
times we add months will be the final contract periods.
    for i in count{
        nextDate = addmonths(startDate, monthsToAdd * returnPeriods);
        dayOfAddedMonth = datetostr(nextDate, "dd");
        dayOfStartDate = datetostr(startDate, "dd");
        compareDate = nextDate;
        if(atoi(dayOfAddedMonth) == atoi(dayOfStartDate )){
            compareDate = minusdays(nextDate, 1);
        }
        if(comparedates(compareDate, endDate) >= 0 ) {
            break;
        }
        returnPeriods = returnPeriods + 1;//Increment the returnPeriods counter if end
date(nextDate) for the period is before the contract end date.
    }
}
return returnPeriods;

```

## APPENDIX M: SUBSCRIPTION PRICING UTILITY BMLS

The Oracle CPQ Subscription Management package includes Subscription Pricing BML for the following:

- Calculate List Price
- Prepare Tier Pricing
- Populate Charges for Arrays
- Populates Tier Information for Arrays
- Oracle Pricing Subscription Base Profile
- Get Lookups

### ***Calculate List Price (oRCL\_pRC\_calculateListPrice) BML***

The BML for the Calculate List Price action is used to calculate the list price for a product.

```
listPrice = 0;
if(quantity > allowance){
  finalQuantity = quantity - allowance;
}else{
  finalQuantity = 0;
}

if(blockSize > 0){
  finalQuantity = ceil((finalQuantity * 1.0) / blockSize);
}

listPrice = listPrice + price* finalQuantity ;
return listPrice;
```

### ***Prepare Tier Info JSON (oRCL\_pRC\_prepareTierInfoJson) BML***

The BML for the Prepare Tier Info JSON action is used to prepare tier pricing information for a product.

```
tierInfoJson = json();

jsonput(tierInfoJson, "SequenceNumber", tierSeq);
jsonput(tierInfoJson, "TierFrom", tierMin);
jsonput(tierInfoJson, "TierTo", tierMax);
jsonput(tierInfoJson, "ListPrice", tierPrice);

if(tierBlockSize > 1){
  jsonput(tierInfoJson, "PriceFormat", "ORA_PER_BLOCK");
}
else{
  jsonput(tierInfoJson, "PriceFormat", "ORA_PER_UNIT");
}

jsonput(tierInfoJson, "BlockSize", tierBlockSize);

return tierInfoJson;
```

## Populate Charges (oRCL\_pRC\_populateCharges) BML

The BML for the Populate Charges action is used to populate the charge information for a charge array. This BML is invoked from Transaction Line Advanced Default – After Formulas.

```
returnString = "";

chargesSize = jsonarraysize(charges);
itr = string[chargesSize];
chargecount=0;
for i in itr {
    charge = jsonArrayget(charges, chargecount, "json");
    chargecount = chargecount + 1;

    chargeType = jsonget(charge,"chargeType","string");
    periodicity = jsonget(charge,"periodicity","string");
    // Get CPQ codes for charge type and periodicity.
    lookups = util.oRCL_pRC_getLookups(periodicity, chargeType, false);

    returnString = returnString + "|" + documentNumber + "~priceType_l~" + get(lookups,
"ChargeType");

    if(isnull(periodicity) == false ){
        returnString = returnString + "|" + documentNumber + "~pricePeriod_l~" + get(lookups,
"Periodicity");
    }

    if(jsonpathcheck(charge,"$.Allowance")){
        returnString = returnString + "|" + documentNumber + "~oRCL_pRC_blockAllowance~" +
string(jsonget(charge,"Allowance","integer"));
    }

    if(jsonpathcheck(charge,"$.BlockSize")){
        returnString = returnString + "|" + documentNumber + "~oRCL_pRC_blockSize~" +
string(jsonget(charge,"BlockSize","integer"));
    }

    tiered = "N";
    if(NOT(isnull(jsonget(charge,"tiered")))){
        tiered = jsonget(charge,"tiered","string");
    }
    returnString = returnString + "|" + documentNumber + "~oRCL_pRC_tierd_l~" + tiered + "|";

    if(tiered == "Y") {

        //jsonput(chargeJson,"oRCL_pRC_tiertype_l",jsonget(charge,"tierType"));
        returnString = returnString + "|" + documentNumber + "~oRCL_pRC_tiertype_l~" +
jsonget(charge,"tierType","string") + "|";
        tierList = jsonget(charge,"tierList","jsonArray");
        if(isnull(tierList) == false){
            returnString = returnString + "|" + util.oRCL_pRC_populateTiers(tierList,
documentNumber);
        }
    }
}
return returnString;
```

### **Populate Tiers (oRCL\_PRC\_populateTiers) BML**

The BML for the Populate Tiers action is used to populate tier information for a tier array. This BML is invoked from Transaction Line Advanced Default – After Formulas.

```
returnString = ""; tiers = jsonArray(); tierListSize = jsonarraysize(tierList); itr =
string[tierListSize]; tierCount=0; for j in itr { tier = json(); tierInfo =
jsonarrayget(tierList ,tierCount,"json"); tierCount = tierCount + 1; if(isnull(tierInfo )
== false) {      jsonput(tier, "oRCL_PRC_tierSequence", jsonget(tierInfo
,"SequenceNumber","integer"));      jsonput(tier, "oRCL_PRC_tierFrom", jsonget(tierInfo
,"TierFrom","integer"));      jsonput(tier, "oRCL_PRC_tierTo", jsonget(tierInfo
,"TierTo","integer"));      jsonput(tier, "oRCL_PRC_tierBlockSize", jsonget(tierInfo
,"BlockSize","integer"));      jsonput(tier, "oRCL_PRC_tierListPrice", jsonget(tierInfo
,"ListPrice","float"));      jsonput(tier, "oRCL_PRC_tierPriceFormat", jsonget(tierInfo
,"PriceFormat","string"));
}
jsonarrayappend(tiers , tier);
} tiersSize = jsonarraysize(tiers); if(tiersSize > 0) { returnString = returnString +
documentNumber + "~oRCL_PRC_tierInfo~" + jsonArraytostr(tiers)+"|";
} return returnString;
```

### **Oracle Pricing Subscription Base Profile (oRCL\_PRC\_oraclePricingSubscriptionBaseProfile) BML**

The BML for Oracle Pricing Subscription Base Profile action is used to calculate the price based on the pricing model.

```
// This utility BML will calculate the price based on pricing model(Single or Tiered).
// Name: Oracle Pricing Subscription Base Profile
// Variable Name: oRCL_PRC_oraclePricingSubscriptionBaseProfile
// Input:
// partNumber(String) - This is used to look up rows from the ORCL_PRC_BASE_CHGS table
// quantity(Integer) - Used for calculating the price.
// chargeName(String)
// lockedAllowance(Integer)
// lockedBlockSize(Integer)
// lockedListPrice(Float)
// tierFrom(Integer[])
// tierTo (Integer[])
// tierListPrice(Float[])
// tierBlockSize(Integer[])
// lockUsage(Boolean)
// Output:
// JSON - Includes UnitPrice and other JSON elements for Subscription Service.
// Dependency : Calculate List Price, Prepare Tier Info Json
//
// Note : Input parameter chargeName is no more used in calculation but keeping here so
that OSS doesn't have to modify their // payload.

lang = dict("string");
fields = dict("string");
where = "";

put(fields, "$field1", partNumber);
where = "Product= $field1";

// Get the results from the ORCL_PRC_BASE_CHGS to calculate the price.
charges = bmql("select Charge_Id, Price, Block_Size, Allowance, Charge_Type, Periodicity,
Tiered, TierType from ORCL_PRC_BASE_CHGS where $where",lang,fields);
chargeList = jsonArray();
returnPayload = json();
errorInPricing = false;
```

```

errorMessagesJsonArr = jsonarray();
unitPriceEach = 0.0;
for charge in charges { // Iterating through charges queried from data table
ORCL_PRC_BASE_CHGS.
    listPrice = 0.0;
    unitPrice = 0.0;
    unitListPrice = 0.0;

    tierList = jsonarray();
    // Reading the each charge details
    tiered = get(charge, "Tiered");
    chargeId = getInt(charge, "Charge_Id");
    chargeType = get(charge, "Charge_Type");
    quantityForCalc = 0.0;
    if(chargeType == "Usage") {
        quantityForCalc = usageValue;
    } else {
        quantityForCalc = quantity;
    }

    periodicity = get(charge, "Periodicity");
    tierType = get(charge, "TierType");
    allowance = getInt(charge, "Allowance");
    chargeJson= json();
    if(tiered == "N"){ // Non tiered
        // Checking charge is of type lockable, if yes and locked then calculating the price
based on locked values
        if(chargeType == "Usage" AND lockUsage) {
            price = lockedListPrice;
            blockSize = lockedBlockSize;
            allowance = lockedAllowance;
        }else{
            price = getfloat(charge, "Price");
            blockSize = getInt(charge, "Block_Size");
        }

        // Price from data table
        unitListPrice = price;

        // Calculating List Price
        listPrice = util.ORCL_PRC_calculateListPrice(quantityForCalc, allowance, blockSize,
price);

        // Calculating Unit Price
        if(quantityForCalc <> 0){
            unitPrice = listPrice / quantityForCalc;
        }

    }else{
        if(tierType == "ORA_ALL_TIERS"){
            prevMax = 0;
            tierSeq = 1;
            // Checking charge is of type lockable, if yes and locked then calculating the
price based on locked values
            if(chargeType == "Usage" AND lockUsage) {
                tierSize= sizeofarray(tierFrom);
                tierLoopArr = range(tierSize);
                for tierNum in tierLoopArr{
                    tierMin = tierFrom[tierNum];
                    tierMax = tierTo[tierNum];

```

```

        blockSize = tierBlockSize[tierNum];
        tierPrice = tierListPrice[tierNum];
        if(tierMax == 0 OR (tierMax >= quantityForCalc)){ // True if we are at the
last row of the tier.
            iterationquantityForCalc = quantityForCalc - prevMax; // Get the
quantityForCalc to charge for this tier.
            if(iterationquantityForCalc < 0){
                iterationquantityForCalc = 0;
            }
        }else{ // True if the tier still has more rows after first tier.
            iterationquantityForCalc = tierMax - prevMax; // Get the
quantityForCalc to charge for this row.
        }

        listPrice = listPrice +
util.orcl_prc_calculateListPrice(iterationquantityForCalc, allowance, blockSize,
tierPrice);

        if(quantityForCalc <> 0){
            unitPrice = listPrice / quantityForCalc;
        }

        // Preparing tier info
        tierInfoJson = util.orcl_prc_prepareTierInfoJson(tierSeq, tierMin, tierMax,
tierPrice, blockSize);

        //Adding tier Info to tier list
        jsonArrayappend(tierList, tierInfoJson);

        prevMax = tierMax; // Set the previous max for the if statement above on
the next loop.
        tierSeq = tierSeq + 1; // Increment the tier sequence number for the next
iteration of the loop.
        if(tierMax <= 0){ // stop processing tiers if <= 0 since that is used as a
max identifier
            break;
        }
    }
}
}else{
    // If tiered pricing then querying ORCL_PRC_BASE_TIERS data table to get the
tier info.
    tiers = bmql("select Tier_Min, Tier_Max, Block_Size, Price from
ORCL_PRC_BASE_TIERS where Charge_Id= $chargeId ORDER BY Tier_Min ASC");

    for tier in tiers { // Iterating through tiers
        // Reading the tier info
        tierMin = getint(tier, "Tier_Min");
        tierMax = getint(tier, "Tier_Max");
        blockSize = getint(tier, "Block_Size");
        tierPrice = getfloat(tier, "Price");
        if(tierMax == 0 OR (tierMax >= quantityForCalc)){ // True if we are at the
last row of the tier.
            iterationquantityForCalc = quantityForCalc - prevMax; // Get the
quantityForCalc to charge for this tier.
            if(iterationquantityForCalc < 0){
                iterationquantityForCalc = 0;
            }
        }else{ // True if the tier still has more rows after first tier.
            iterationquantityForCalc = tierMax - prevMax; // Get the
quantityForCalc to charge for this row.

```

```

    }

    listPrice = listPrice +
util.oRCL_pRC_calculateListPrice(iterationquantityForCalc , allowance, blockSize ,
tierPrice);

    if(quantityForCalc <> 0){
        unitPrice = listPrice / quantityForCalc;
    }
    // Preparing tier info
    tierInfoJson = util.oRCL_pRC_prepareTierInfoJson(tierSeq, tierMin, tierMax,
tierPrice, blockSize);

    //Adding tier Info to tier list
    jsonarrayappend(tierList, tierInfoJson);

    prevMax = tierMax; // Set the previous max for the if statement above on
the next loop.
    tierSeq = tierSeq + 1; // Increment the tier sequence number for the next
iteration of the loop.
    if(tierMax <= 0){ // stop processing tiers if <= 0 since that is used as a
max identifier
        break;
    }
}

}
}
}
if(tierType == "ORA_HIGHEST_TIER"){

    tierSeq = 1;
    // Checking charge is of type lockable, if yes and locked then calculating the
price based on locked values
    if(chargeType == "Usage" AND lockUsage) {
        tierSize= sizeofarray(tierFrom);
        tierLoopArr = range(tierSize);
        for tierNum in tierLoopArr{
            tierMin = tierFrom[tierNum];
            tierMax = tierTo[tierNum];
            blockSize = tierBlockSize[tierNum];
            tierPrice = tierListPrice[tierNum];

            if(tierMin <= quantityForCalc AND (tierMax >= quantityForCalc OR tierMax ==
0)){ // True if we are at the last row of the tier.
                listPrice = listPrice + util.oRCL_pRC_calculateListPrice(quantityForCalc,
allowance, blockSize, tierPrice);

                if(quantityForCalc <> 0){
                    unitPrice = listPrice / quantityForCalc;
                }
            }

            // Preparing tier info
            tierInfoJson = util.oRCL_pRC_prepareTierInfoJson(tierSeq, tierMin, tierMax,
tierPrice, blockSize);

            //Adding tier Info to tier list
            jsonarrayappend(tierList, tierInfoJson);
            tierSeq = tierSeq + 1; // Increment the tier sequence number for the next
iteration of the loop.

```



```

    }
    }else{
        // If tiered pricing then querying ORCL_PRC_BASE_TIERS data table to get the
tier info.
        tiers = bmql("select Tier_Min, Tier_Max, Block_Size, Price from
ORCL_PRC_BASE_TIERS where Charge_Id = $chargeId ORDER BY Tier_Min ASC");

        for tier in tiers { // Iterating through tiers
            // Reading the tier info
            tierMin = getint(tier, "Tier_Min");
            tierMax = getint(tier, "Tier_Max");
            blockSize = getint(tier, "Block_Size");
            tierPrice = getfloat(tier, "Price");
            if(tierMin <= quantityForCalc AND (tierMax >= quantityForCalc OR tierMax ==
0)){ // True if we are at the last row of the tier.
                listPrice = listPrice + util.oRCL_pRC_calculateListPrice(quantityForCalc,
allowance, blockSize, tierPrice);

                if(quantityForCalc <> 0){
                    unitPrice = listPrice / quantityForCalc;
                }
            }
            // Preparing tier info
            tierInfoJson = util.oRCL_pRC_prepareTierInfoJson(tierSeq, tierMin, tierMax,
tierPrice, blockSize);

            //Adding tier Info to tier list
            jsonarrayappend(tierList, tierInfoJson);

            tierSeq = tierSeq + 1; // Increment the tier sequence number for the next
iteration of the loop.

        }

    }

}

}

}

// Preparing the charge details
jsonput(chargeJson, "chargeName", chargeName);
jsonput(chargeJson, "unitPrice", unitPrice);
jsonput(chargeJson, "listPrice", listPrice);
jsonput(chargeJson, "unitListPrice", unitListPrice);

// Get OSS codes for charge type and periodicity.
lookups = util.oRCL_pRC_getLookups(periodicity, chargeType, true);
jsonput(chargeJson, "chargeType", get(lookups, "ChargeType"));
jsonput(chargeJson, "periodicity", get(lookups, "Periodicity"));
if(tiered == "Y"){ // Include Tier information if tiered pricing.
    jsonput(chargeJson, "tiered", "Y");
    jsonput(chargeJson, "tierList", tierList);
    jsonput(chargeJson, "tierType", tierType);
}else{
    jsonput(chargeJson, "Allowance", allowance );
    jsonput(chargeJson, "BlockSize", blockSize );
}
// Adding charge details to charge list

```

```

jsonarrayappend(chargeList , chargeJson);
if(chargeType == "Usage"){
    unitPriceEach = listPrice;
} elif (quantityForCalc <> 0){
    unitPriceEach = unitPrice;
}else {
    unitPriceEach =unitListPrice;
}
}

calculationInfoPayload = json();

jsonput(returnPayload, "unitPrice", unitPriceEach);
jsonput(calculationInfoPayload , "charges", chargeList);

jsonput(calculationInfoPayload, "hasErrors", errorInPricing); // Adding the hasErrors item
if(errorInPricing){ // Including Error information if there are errors
    jsonput(calculationInfoPayload, "errorInfo", errorMessagesJsonArr);
}
jsonput(returnPayload, "calculationInfo", calculationInfoPayload ); // Adding the
calculationInfo child to the payload.
return returnPayload;

```

### **Get Lookups (oRCL\_pRC\_getLookups) BML**

This BML is used to find out matching OSS or CPQ code for periodicity and charge type.

```

lookupDict = dict("string");

if(cpqCode) {
    ossLookups = bmql("select Type, OSS_Code from ORCL_PRC_LOOKUP where CPQ_Code =
$chargeTypeCode or CPQ_Code = $periodicityCode");
    for ossLookup in ossLookups {
        lookupType = get(ossLookup, "Type");
        if(lookupType == "Periodicity") {
            put(lookupDict, "Periodicity", get(ossLookup, "OSS_Code"));
        } elif (lookupType == "ChargeType") {
            put(lookupDict, "ChargeType", get(ossLookup, "OSS_Code"));
        }
    }
} else {
    cpqLookups = bmql("select Type, CPQ_Code from ORCL_PRC_LOOKUP where OSS_Code =
$chargeTypeCode or OSS_Code = $periodicityCode");
    for cpqLookup in cpqLookups {
        lookupType = get(cpqLookup, "Type");
        if(lookupType == "Periodicity") {
            put(lookupDict, "Periodicity", get(cpqLookup, "CPQ_Code"));
        } elif (lookupType == "ChargeType") {
            put(lookupDict, "ChargeType", get(cpqLookup, "CPQ_Code"));
        }
    }
}

return lookupDict;

```

## APPENDIX N: IMPLEMENTATION FOR DISCOUNT EFFECTIVITY TYPES

The Oracle CPQ Subscription Management package includes discount effectivity of type All Term [ORA\_ALL\_TERM]. Follow below changes if other types of effectivity has to be included:

### Commerce Attributes

Make below changes to Transaction Line attributes:

- Update Discount Effectivity Type (discountEffectivityType\_1)
  - Add below menu entries
    - Starting Period (ORA\_PERIODS\_FROM\_START\_DT)
    - Ending Period (ORA\_PERIODS\_BEFORE\_END\_DT)
    - Specific Periods (ORA\_SPECIFIC\_PERIODS)
- Add new attribute Discount Effectivity Periods (discountEffectivityPeriods\_1)
  - Type : Text
  - This attribute should be editable so that user can override default value.

### Commerce Library Functions

Make below changes to commerce library function:

- Edit Apply Discounts (oRCL\_sm\_applyDiscounts)
- Replace existing script with the script below.

```
discStDtEndDtFormat = "yyyy-MM-dd";
contractStartDate = strtodate(contractStartDateStr, contractStartDateFormat);
returnStrBuilder = stringBuilder("");
adjustmentList = bmql("select Start_Date, End_Date, Discount_Name, Discount_Value,
Discount_Type, Discount_Reason, Discount_Type, Discount_Effectivity, Effectivity_Periods
from ORCL_PRC_DISCOUNTS where Product=$partNumber ORDER BY Start_Date, End_Date");
for adjustment in adjustmentList {
    discFound = false;
    discStDtStr = get(adjustment,"Start_Date");
    discEndDtStr = get(adjustment,"End_Date");

    if((isnull(discStDtStr) or trim(discStDtStr) == "") and (isnull(discEndDtStr ) or
trim(discEndDtStr) == "")) {
        discFound = true;
    } elif (trim(discStDtStr) <> "" and (isnull(discEndDtStr) or trim(discEndDtStr) == ""))
{
        discStDt = strtodate(get(adjustment,"Start_Date"), discStDtEndDtFormat);
        if(comparedates(discStDt, contractStartDate) <= 0) {
            discFound = true;
        }
    } elif ((isnull(discStDtStr) or trim(discStDtStr) == "") and trim(discEndDtStr) <> "")
{
        discEndDt = strtodate(get(adjustment,"End_Date"), discStDtEndDtFormat);
        if(comparedates(discEndDt, contractStartDate) >= 0) {
            discFound = true;
        }
    } else {
        discStDt = strtodate(get(adjustment,"Start_Date"), discStDtEndDtFormat);
        discEndDt = strtodate(get(adjustment,"End_Date"), discStDtEndDtFormat);
        if(comparedates(discStDt, contractStartDate) <= 0 and comparedates(discEndDt,
contractStartDate) >= 0) {
            discFound = true;
        }
    }
}

if(discFound) {
```

```

        validDiscEntry = false;
        discEffectivityType = get(adjustment,"Discount_Effectivity");
        discEffectivityPrdsStr = get(adjustment,"Effectivity_Periods");
        if(discEffectivityType == "ORA_ALL_TERM") {
            sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityPeriods_1~", string(contractPeriods));
            validDiscEntry = true;
        } elif ((discEffectivityType == "ORA_PERIODS_FROM_START_DT" or discEffectivityType
== "ORA_PERIODS_BEFORE_END_DT") and isnumber(discEffectivityPrdsStr) and priceType <> "One
Time") {
            discEffectivityPrds = atoi(discEffectivityPrdsStr);
            if(discEffectivityPrds < contractPeriods) {
                sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityPeriods_1~",discEffectivityPrdsStr);
            } else {
                sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityPeriods_1~", string(contractPeriods));
            }
            validDiscEntry = true;
        } elif (discEffectivityType == "ORA_SPECIFIC_PERIODS" and priceType <> "One Time")
{
            effPrdsArray = split(discEffectivityPrdsStr, ",");
            if(isnull(effPrdsArray) == false and sizeofarray(effPrdsArray)==2){
                if(isnumber(trim(effPrdsArray[0])) and
isnumber(trim(effPrdsArray[1]))){
                    startPrd = atoi(effPrdsArray[0]);
                    endPrd = atoi(effPrdsArray[1]);
                    if(contractPeriods >= startPrd and contractPeriods >= endPrd) {
                        sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityPeriods_1~",discEffectivityPrdsStr);
                        validDiscEntry = true;
                    } elif(contractPeriods >= startPrd and contractPeriods < endPrd) {
                        sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityPeriods_1~",sbtostring(stringbuilder(string(startPrd), ",",
string(contractPeriods))));
                        validDiscEntry = true;
                    }
                }
            }
        }

        if(validDiscEntry) {
            sbappend(returnStrBuilder, "|", lineDocNumber,
"~customDiscountType_1~",get(adjustment,"Discount_Type"));
            sbappend(returnStrBuilder, "|", lineDocNumber,
"~customDiscountValue_1~",get(adjustment,"Discount_Value"));
            sbappend(returnStrBuilder, "|", lineDocNumber,
"~discountEffectivityType_1~",discEffectivityType);
        }
        break;
    }
}
return sbtostring(returnStrBuilder);

```

- Add new commerce library function to calculate contract values
  - Function name and variable name: Calculate Contract Values (calculateContractValues)
  - Transaction Line input parameters:
    - priceType\_l
    - discountAmount\_l
    - contractedPeriods\_l
    - discountEffectivityType\_l
    - discountEffectivityPeriods\_l
    - \_document\_number
    - listAmount\_l
  - Add the library function code below.

```

returnStrBuilder = stringBuilder("");

totalContractDiscVal = 0.0;
totalContractListVal = 0.0;
totalContractValue = 0.0;

for line in transactionLine {
  discContractVal = 0.0;
  listContractVal = 0.0;
  netContractVal = 0.0;
  if(line.priceType_l == "One Time"){
    discContractVal = line.discountAmount_l;
    listContractVal = line.listAmount_l;
    netContractVal = listContractVal - discContractVal;
  } elif (line.priceType_l == "Recurring" or line.priceType_l == "Usage") {
    listContractVal = line.listAmount_l * line.contractedPeriods_l;
    if(line.discountEffectivityType_l == "ORA_ALL_TERM") {
      discContractVal = line.discountAmount_l * line.contractedPeriods_l;
    } elif((line.discountEffectivityType_l == "ORA_PERIODS_FROM_START_DT" or
line.discountEffectivityType_l == "ORA_PERIODS_BEFORE_END_DT") and
isnumber(line.discountEffectivityPeriods_l)) {
      discEffPrdInt = atoi(line.discountEffectivityPeriods_l);
      if(discEffPrdInt > line.contractedPeriods_l){
        discContractVal = line.discountAmount_l * line.contractedPeriods_l;
      } else {
        discContractVal = line.discountAmount_l * discEffPrdInt;
      }
    }elif(line.discountEffectivityType_l == "ORA_SPECIFIC_PERIODS"){
      strArray = split(line.discountEffectivityPeriods_l, ",");
      if(sizeofarray(strArray)==2){
        if(isnumber(strArray[0]) and isnumber(strArray[1])){
          startPrd = atoi(strArray[0]);
          endPrd = atoi(strArray[1]);
          bol = startPrd >= line.contractedPeriods_l;
          if(startPrd <= endPrd and startPrd > 0 and endPrd > 0 and startPrd
<= line.contractedPeriods_l and endPrd <= line.contractedPeriods_l){
            noOfPrds = endPrd - startPrd + 1;
            discContractVal = line.discountAmount_l * noOfPrds;
          }
        }
      }
    }
  }
  netContractVal = listContractVal - discContractVal;
}
totalContractDiscVal = totalContractDiscVal + discContractVal;
totalContractListVal = totalContractListVal + listContractVal;
totalContractValue = totalContractValue + netContractVal;

```

```

    sbappend(returnStrBuilder, "|", line._document_number,
"~contractDiscount_l~",string(discContractVal));
    sbappend(returnStrBuilder, "|", line._document_number,
"~contractListValue_l~",string(listContractVal));
    sbappend(returnStrBuilder, "|", line._document_number,
"~contractValue_l~",string(netContractVal));
}
sbappend(returnStrBuilder, "|1", "~totalContractDiscount_t~",string(totalContractDiscVal));
sbappend(returnStrBuilder, "|1",
"~totalContractListValue_t~",string(totalContractListVal));
sbappend(returnStrBuilder, "|1", "~totalContractValue_t~",string(totalContractValue));
return sbtoString(returnStrBuilder);

```

- Call this function from the Save action Advanced Modify – Before Formulas > Define Advanced Modify – Before Formulas.

## APPENDIX O: CALCULATE PRICE API

This operation calculates the price of items using the Oracle CPQ pricing engine.

url: <https://<hostname>.com/rest/v13/pricing/actions/calculatePrice>

PAYLOAD ATTRIBUTE	DATA TYPE	COMMENTS
customerId	string	Customer ID of the customer (account).
currencyCode	string	Code for the currency.
priceBookVarName	string	Variable name of the CPQ price book.
headerAttributeValues	object	Values of pricing attributes at the header level.
oRCL_pRC_priceAsOfDate	string	Price as of date.
partNumber	string	Part Number of the CPQ product.
itemAttributeValues	object	Values of pricing attributes at the item level for a specific item.
oRCL_pRC_partNumber	string	Part Number
oRCL_pRC_blockSize	integer	Block Size
oRCL_pRC_tierTo	array	Array of integer providing 'Tier To' numbers.
oRCL_pRC_tierFrom	array	Array of integer providing 'Tier From' numbers.
oRCL_pRC_tierListPrice	array	Array of number providing list price for each tier.
oRCL_pRC_quantity	integer	Quantity
oRCL_pRC_allowance	integer	Allowance
oRCL_pRC_tierBlockSize	array	Array of integer providing block sizes.
oRCL_pRC_externalParentKey	string	External parent key

PAYLOAD ATTRIBUTE	DATA TYPE	COMMENTS
oRCL_pRC_usageValue	number	Usage value for products of type usage. Use this attribute to map quantity in usage rating for parts of type "Usage".
oRCL_pRC_productType	string	Product Type
oRCL_pRC_listPrice	number	Product list price.
oRCL_pRC_contractStartDate	string	Contract start date

Below is the sample payload:

```
{
  "customerId": "ATT",
  "currencyCode": "USD",
  "priceBookVarName": "_default_price_book",
  "headerAttributeValues": {
    "oRCL_pRC_priceAsOfDate": "2018-04-23"
  },
  "items": [
    {
      "itemIdentifier": "1",
      "partNumber": "SUV Charging Station use",
      "itemAttributeValues": {
        "chargeName": "Consumption Fee",
        "oRCL_pRC_contractStartDate": "2018-04-23",
        "oRCL_pRC_useUsageLock": true,
        "oRCL_pRC_usageValue": 10,
        "oRCL_pRC_tierBlockSize": [
          25,
          1
        ],
        "oRCL_pRC_tierFrom": [
          0,
          50
        ],
        "oRCL_pRC_tierTo": [
          51,
          0
        ],
        "oRCL_pRC_tierListPrice": [
          20,
          16
        ]
      }
    }
  ]
}
```



## APPENDIX P: TROUBLESHOOTING

The troubleshooting information provided in this appendix contains workarounds for common scenarios administrators may encounter with the Oracle CPQ - Subscription Management solution.

### Manually Add BOM Parts to Oracle CPQ

If the BOM Item Tree Administration page shows some parts in red, this indicates that not all BOM parts from the BOM package were added to the Oracle CPQ site. Administrators must manually add these parts to their Oracle CPQ site.

To manually add BOM parts to Oracle CPQ, perform the following steps:

1. Open the Admin Home page.
2. Select **Parts** under **Products**. The Product Administration page opens.
3. Click **Add New Parts**. The Part Editor opens.
4. Enter the part number shown in red, as it displays on the BOM Item Tree Administration page.
5. If the part corresponds to a subscription product, populate the **Product Type** with **subscription** in the Extended Information section.
6. Click **Add**.
7. Repeat the above steps for all of the missing parts that display in red on the BOM Item Tree Administration page.

### Warning Message with Initial Install of Subscription Management Package

When installing the Subscription Management package for the first time, and the OIC site does not have Subscription Management-related integration installed yet, the migration succeeds and Commerce migration displays a warning message similar to the following:

```
"WARNING
0 out of 2 Integrations were updated successfully.
Please update the process integrations to work properly"
```

This warning indicates the OIC flow doesn't exist yet, so migration could not update the OIC integration endpoint. To address the situation, after importing OIC flows, go to the integration center to disable and re-enable the OIC integration, then redeploy the Commerce process.

### Resolve Issues with Submit Order Action

If installing the Subscription Ordering package results in issues with the Submit Order action, a workaround is available.

To resolve issues with the Create Subscription action, perform the following steps:

1. Open Oracle CPQ.
2. Select **Integration Center** under **Integration Platform**. The Integration Center opens.
3. Select the **ICS integration** from the left pane.
4. Unmark the **Enable Integration** check box to disable the integration.
5. Click **Save**.
6. Re-enable the ICS integration.
7. Click **Save**.

## Enable the OSS Renew Event in the OIC Environment

When sites do not have valid certificates, the OSS Renew event does not work in OIC. Administrators can enable the Renew event in OIC by beginning the OIC integration URLs with “https” and uploading the OIC certificates to Oracle CX Sales.

## CONNECT WITH US

Call +1.800.ORACLE1 or visit [oracle.com](http://oracle.com).

Outside North America, find your local office at [oracle.com/contact](http://oracle.com/contact).

 [blogs.oracle.com](http://blogs.oracle.com)

 [facebook.com/oracle](https://facebook.com/oracle)

 [twitter.com/oracle](https://twitter.com/oracle)

Copyright © 2019, 2023, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

**U.S. GOVERNMENT END USERS:** Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

