



Oracle CPQ Cloud

What's New in 2016 R1

August 2016

TABLE OF CONTENTS

REVISION HISTORY	4
OVERVIEW.....	5
RELEASE FEATURE SUMMARY	6
ENTERPRISE EXCELLENCE.....	7
<i>Bill of Material (BOM) Mapping</i>	<i>7</i>
BOM Mapping Overview	8
BOM Mapping Tables	10
BOM Administration.....	12
Bills of Materials Page	13
Declare Util Function	17
BOM Mapping Rules.....	18
Reference Information	19
<i>Subscription Ordering</i>	<i>29</i>
Assets	29
Asset Creation	31
Asset Modification.....	31
Reconfigure	32
Follow-On Orders	32
Asset Termination	32
<i>Contract Negotiation</i>	<i>33</i>
Enable Contract Negotiation	33
Create Contract From Single-Language Template.....	34
Capture Versions of the Contract	34
Generate List of Differences Between Document Versions	34
Merge Approved Changes	35
<i>Secure Data Table Columns</i>	<i>36</i>
Adding a Secure Column to a Data Table	38
EASY ADMINISTRATION.....	40
<i>Single Select Pick Lists in Configuration</i>	<i>40</i>
Apply Hiding Rules to a Single Select Pick List Attribute.....	40
Use Related Rules Tab to Find References Made from a Single Select Pick List	41
Display a Single Select Pick List as an Image Grid	42
Use Array Set as Source of Single Select Pick List Data.....	44
<i>UI Designer</i>	<i>47</i>
Layouts List Page	47
UI Designer Screen Layout.....	48
Layout.....	49
Attributes	49
Layout Settings	50
Panel Settings	52
Table Settings	53
Column Settings.....	54
Button Settings.....	55
<i>Performance Logs Page</i>	<i>57</i>
Query by Example (QBE)	57
<i>BML Enhancements</i>	<i>58</i>
JSON Related Functions.....	59
JSON Array Related Functions	59
JSON Path Related Functions.....	60
Functions to Support Remote Approvals.....	65
Generate Unique IDs Function	69
URL Access Function	69
Same Server Authentication.....	70

User Session Functions	70
Global Dictionary Functions	72
Throw Error	75
Apply Template	76
BML Print Log	77
<i>Document Designer Enhancements</i>	78
Performance Enhancements	78
Formatting and Style Enhancements.....	78
Contract Negotiation Enhancements	86
<i>Long Running Thread Diagnostics</i>	88
View Logs.....	88
Timeout Action Settings	91
INTEGRATION	93
<i>Salesforce1 Integration</i>	93
<i>CPQ Cloud – eBusiness Suite (EBS) BOM Reference Integration</i>	95
<i>Process Cloud Service (PCS) Integration</i>	97
Approvals Overview.....	98
Approval Sequence Selection	99
Remote Approval Process Functions	99
<i>Rest APIs</i>	101
REST APIs for Contract Negotiations	103
REST APIs for Subscription Ordering.....	104
Enhancements to Query Parameters	115
<i>Platform as a Service (PaaS) Integration Sample Applications</i>	118
XLS to CSV Converter Sample Application	118
Quote Statistics Sample Application.....	118
PRE-UPGRADE CONSIDERATIONS	120
Known Functionality	120
Resolved Known Issues.....	121
Translation Status	121
POST-UPGRADE CONSIDERATIONS	122
Browser Support.....	122
Salesforce Managed Package Support	124
Training.....	124
Additional Information	124

REVISION HISTORY

This document will continue to evolve as existing sections change and new information is added. All updates are logged below, with the most recent updates at the top.

Date	What's Changed	Notes
24 AUG 2016		Initial Document Creation

OVERVIEW

This guide outlines information about new or improved functionality in Oracle Configure, Price, and Quote (CPQ) Cloud 2016 Release 1 (2016 R1). Each section includes a brief description of the feature, the steps you need to take to enable or begin using the feature, any tips or considerations to keep in mind, and the resources available to help you.

GIVE US FEEDBACK

We welcome your comments and suggestions to help us improve this document. Send your feedback to CPQ_Cloud_documentation_us_grp@oracle.com.

RELEASE FEATURE SUMMARY

Some of the new CPQ Cloud 2016 Release 1 features are automatically available to users after the upgrade and some require action from the company administrator or Oracle.

The following table offers a quick view of the actions required to enable each of the features.

Feature	Action Required to Enable Feature		
	Automatically Available	Administrator Action Required	Oracle Service Request Required
Enterprise Excellence			
BOM Mapping	✓		
Subscription Ordering		✓	
Contract Negotiation		✓	
Secure Data Table Columns	✓		
Easy Administration			
Single Select Pick Lists in Configuration	✓		
UI Designer	✓		
Performance Logs	✓		
BML Enhancements	✓		
Document Designer Enhancements	✓		
Long Running Thread Diagnostics	✓	✓	
Integration			
Salesforce 1 Integration		✓	
CPQ Cloud - EBS BOM Reference Integration		✓	
PCS Integration		✓	
Rest APIs	✓		
PaaS Integration Sample Applications		✓	
Upgrade Considerations			
Pre-Upgrade Considerations		✓	
Post-Upgrade Considerations		✓	

Oracle CPQ Cloud enables companies to streamline the entire opportunity-to-quote-to-order process, including product selection, configuration, pricing, quoting, ordering, and approval workflows. CPQ Cloud provides a flexible, scalable, enterprise-ready CPQ Cloud solution ideal for companies of all sizes that sell products and services across direct, indirect, and e-Commerce sales channels.

Oracle is continually upgrading the functionality of CPQ Cloud to meet the needs of customers who must conform to a variety of regulatory and compliance paradigms. New features that reinforce enterprise excellence include:

- [Bill of Material \(BOM\) Mapping](#)
- [Subscription Ordering](#)
- [Contract Negotiation](#)
- [Secure Data Table Columns](#)

BILL OF MATERIAL (BOM) MAPPING

The new BOM Mapping feature, available in CPQ Cloud 2016 R1, allows administrators to import multi-level BOM product structures for use in CPQ Configuration, Commerce transactions, and downstream integration of orders to an Enterprise Resource Planning (ERP) system. This data-driven solution significantly reduces the amount of time needed to set up and maintain integrations of BOM structures with ERP systems using new BOM tables and a new BOM Mapping rule type.

The BOM Mapping feature delivers the ability to:

- Capture BOM item definitions in CPQ Data Tables. Use existing Data Table migration or upload features to import BOM item definitions from fulfillment systems.
- Map Configuration attributes to the resulting BOM items. Use simple Table-Based rules, advanced BML-Based Rules, or both to accomplish BOM Mapping.
- Create and reconfigure transaction lines from Configuration selections using BOM Mapping rules.
- Generate Sales or Manufacturing BOMs for integration with downstream fulfillment systems.

NOTE: Implementing BOM Mapping requires advanced knowledge of CPQ Configuration and fulfillment system BOM integration.

The BOM Mapping section consists of the following topics:

- [BOM Mapping Overview](#) - describes BOM Mapping, BOM Mapping tables, hierarchical relationships, and BOM table relationships.
- [BOM Administration](#) - examines the new BOM Administration Platform and Configuration rules added for BOM Mapping.
- [Reference Information](#) - covers uses cases, system attributes, BML functions, and Web Service changes to support BOM Mapping.

BOM MAPPING OVERVIEW

Fulfillment systems often maintain bills of material (BOMs) containing complex, multi-level part structures that differ from the Configuration attributes used in CPQ Cloud when sales users configure products. The BOM Mapping feature provides a data-driven mechanism for mapping these differing product views.

As illustrated in Figure 1: BOM Mapping Overview, BOM Mapping enables administrators to associate their fulfillment system BOMs to an Oracle CPQ Cloud Configuration. BOM Mapping captures these product structures in BOM definition tables. Administrators perform item mapping and setup BOM Configuration rules to link the BOM definitions to the CPQ Cloud Configuration attribute selections.

When a customer generates a quote, CPQ uses BOM Mapping rules to create a BOM instance. The BOM instance represents a hierarchy of Commerce transaction lines, containing the BOM items and attributes associated with Configuration selections. The new 'get BOM' function generates a sales or manufacturing BOM using the BOM Mapping rules. CPQ can then send the sales or manufacturing BOM to an ERP system for order fulfillment.

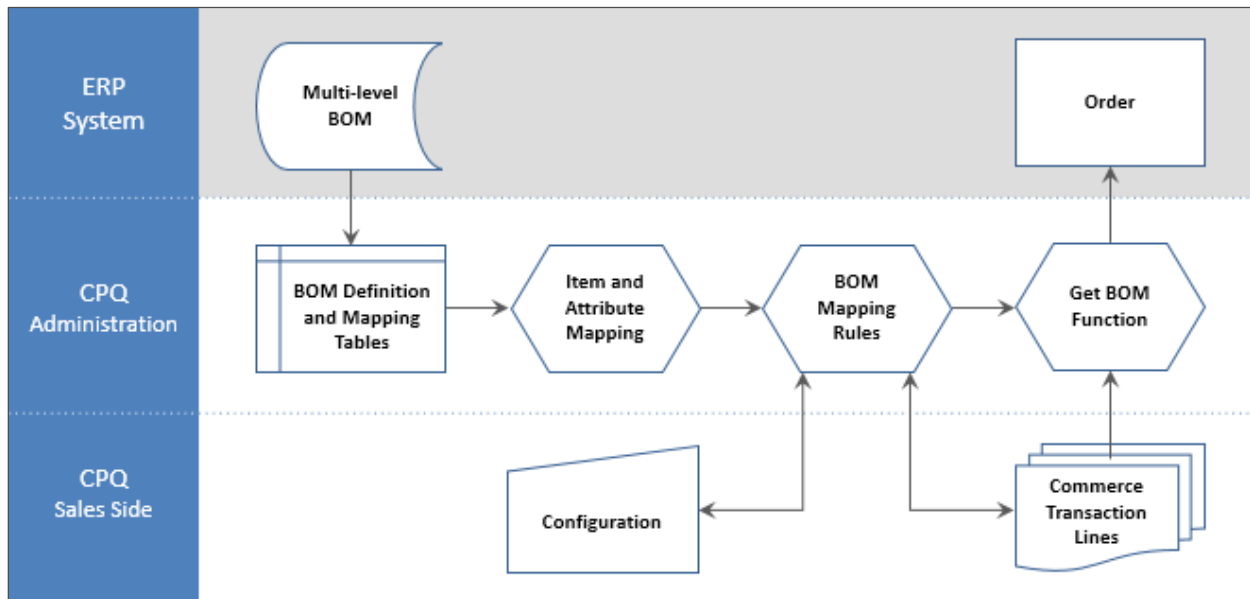


Figure 1: BOM Mapping Overview

HIERARCHICAL RELATIONSHIPS IN BOM MAPPING

A central component of the BOM Mapping feature is the capture of BOM product structure hierarchies in CPQ Cloud for reference by the BOM Mapping rules. The following example illustrates how these hierarchical parent-child relationships are stored.

The fulfillment BOM tree shown in Figure 2 contains root part LP94777 (i.e. a CPQ model) and four parent parts (LAPPRO1101, LAPPRO1109, BP3000, and EXT1000). Parent part BP3000 has three children (BP3025, BP3050, and BP3075), and EXT1000 has two children (EXT2000 and EXT3000) and four grandchildren (EXT2001, EXT2002, EXT3001, and EXT3002).

The CPQ BOM Data Table represents this tree using columns that store the item variable name, the parent item variable name, and the root item variable name.

- For LAPPRO1101, the parent item and root item are both LP94777.
- For BP3025, the parent item is BP3000. The root item is LP94777.
- For EXT2001, the parent item is EXT2000. The root item is still LP94777.

NOTE: Continue this pattern to support unlimited levels of parent-child relationships, allowing the representation of very complex multi-level BOM structures.

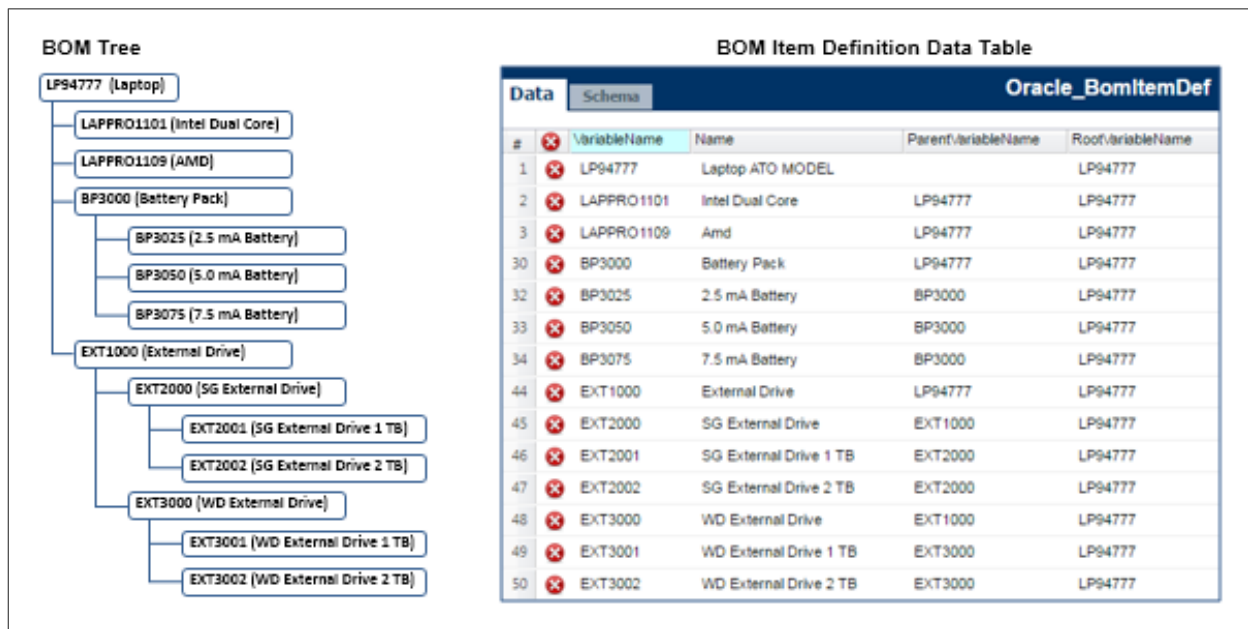


Figure 2: BOM Tree Example

The next section provides an overview of the tables used to store and map these BOM hierarchies to Configuration attributes.

BOM MAPPING TABLES

In 2016 R1, CPQ Cloud introduces five BOM Mapping platform tables to support the full BOM Mapping solution: BOM Item Definition, BOM Item Mapping, BOM Attribute Definition, BOM Attribute Mapping, and BOM Attribute Translation. Many customers may only require one or two of these tables to implement their use cases.

The BOM Mapping platform tables contain the schema for associating BOM structures to Configuration attribute values. Customer-specific mapping details are stored in CPQ Cloud Data Tables. The combination of these two sets of tables enables users to create simple Table-Based Configuration rules to associate fulfillment system BOMs, CPQ Configuration attributes, and Commerce transaction lines without the need for BML or other logic.

Administrators upload or migrate BOM structures to CPQ Data Tables using CPQ Cloud's standard importing features. Data Tables can be linked to the corresponding BOM Mapping platform tables for use in BOM Mapping rules, as shown in the illustration below.

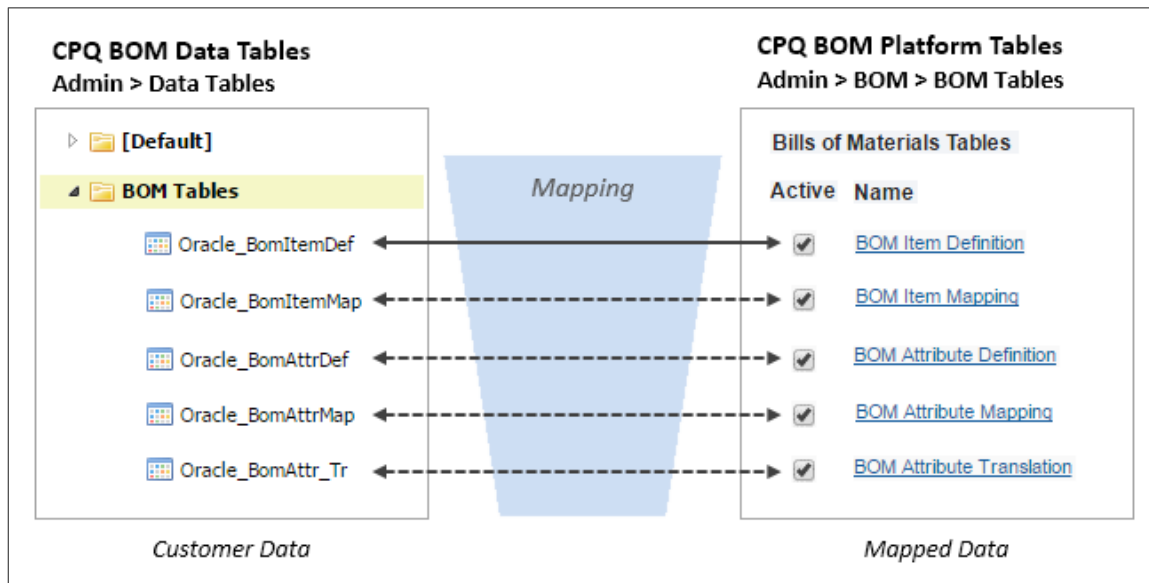


Figure 3: BOM Data Table to BOM Platform Table Mapping

CPQ Cloud provides standard, downloadable Data Table definitions that can be used to create implementation-specific tables, which map automatically to the BOM Mapping platform tables. Alternatively, customers can reuse existing Data Tables containing BOM structure details by mapping the columns in their tables to the BOM Mapping platform tables.

NOTE: Administrators must activate the required BOM Mapping platform tables, and populate BOM Mapping Data Tables prior to BOM Mapping.

BOM ITEM DEFINITION TABLE

The BOM Item Definition Table stores the BOM hierarchical relationships used in the fulfillment system, along with item variable references, which recursively link child items to parent items. In addition to the hierarchical information, BOM definition tables also store other information from the fulfillment BOM, such as:

- Fulfillment system IDs
- Default quantity
- Whether an item is optional, a sales item, or a manufacturing item
- BOM item effective dates

NOTE: The BOM Item Definition table is the key component of BOM Mapping and is the only required table for all use cases. For examples of when these tables are used, refer to [BOM Mapping Use Cases](#).

BOM ITEM MAPPING TABLE

The BOM Item Mapping Table associates BOM items to Configuration attributes. Activating this table enables simple Table-Based BOM Mapping Configuration rules. If this table is not active, administrators would use advance BML-Based Rules to establish the association between BOM items and Configuration attributes.

BOM ATTRIBUTE DEFINITION TABLE

The BOM Attribute Definition Table stores attribute definitions and attribute effective dates from the fulfillment system. These attributes can define options, such as color or size. BOM item variable references associate the fulfillment system attributes with the applicable BOM items.

BOM ATTRIBUTE MAPPING TABLE

The BOM Attribute Mapping Table stores associations between BOM attributes, Configuration attributes, Commerce transaction line attributes, and quantity values. Setting up this table enables simple Table-Based BOM Mapping Configuration rules for associating BOM attributes to CPQ Configuration and Commerce.

BOM ATTRIBUTE TRANSLATION TABLE

The BOM Attribute Translation Table associates translations for the applicable attributes. BOM attribute variable references associate the fulfillment system translation with the applicable attributes.

The next section illustrates the key relationships for BOM tables.

BOM TABLE RELATIONSHIPS

BOM Mapping uses variable names as references to capture hierarchical relationships. BOM Mapping also uses variable names to identify relationships between BOM tables. Figure 4: BOM Table Relationships illustrates these relationships.

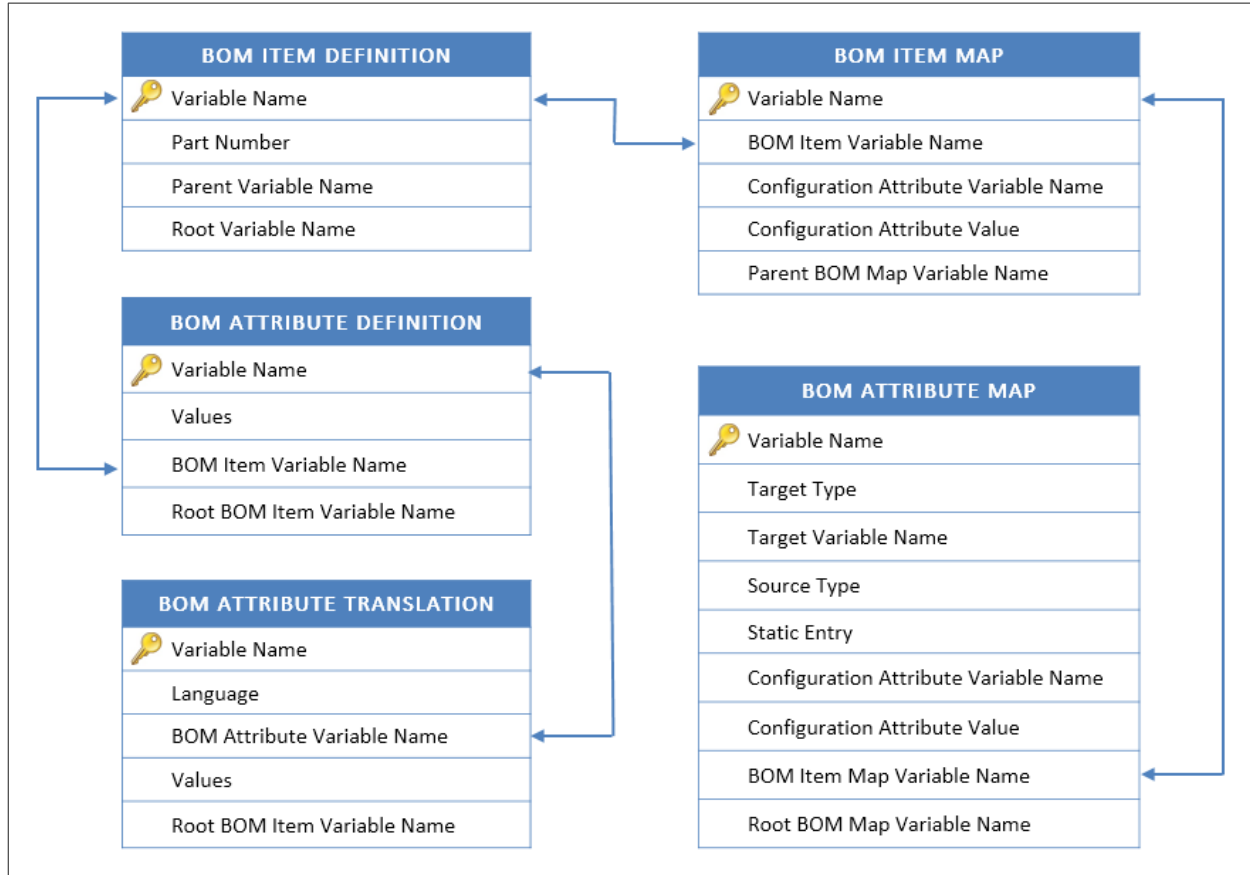


Figure 4: BOM Table Relationships

The next section describes the new CPQ pages, functions, and rules added to support BOM Mapping.

BOM ADMINISTRATION

CPQ Cloud 2016 R1 introduces several new pages to support BOM Mapping implementation and maintenance. This release also provides new functions for product administration to support BOM Mapping Configuration rules.

BOM ADMINISTRATION PLATFORM

The **BOM Administration Platform** page provides administrator access to BOM functions and objects to set up and maintain BOM Mapping Implementations.

- **BOM Tables** - provides access to the **Bills of Materials Tables** page, where administrators can access, activate, and map BOM tables.
- **BOM Root Items List** - enables administrators to inspect root items, and then recursively select components to view all items, attributes, and translations associated with the BOM Item Tree. Administrators can also use these pages to isolate any errors in the BOM Item Tree.
- **Declare Util Function** - allows administrators to select a BML utility function invoked during the Configuration Save event to support Subscription Ordering.

Access the **BOM Mapping Administration Platform** from **Admin > Products > BOM**.

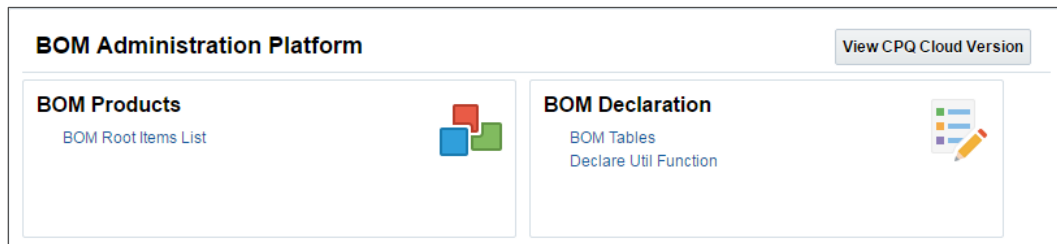


Figure 5: BOM Administration Platform

BILLS OF MATERIALS PAGE

The **Bills of Materials Tables** page provides access to BOM Mapping platform tables, shows mapping status, and provides the ability to activate tables. Administrators only need to activate and map the BOM Mapping platform tables required to meet their business needs.

Bills of Materials Tables			
Bills of Materials Tables			
Active	Name	Mapping Status	Description
<input checked="" type="checkbox"/>	BOM Item Definition	Complete	This table stores the definitions of bills of materials imported from ERP systems.
<input checked="" type="checkbox"/>	BOM Item Mapping	Complete	This table stores the mapping between configuration attributes and BOM items.
<input checked="" type="checkbox"/>	BOM Attribute Definition	Complete	This table stores the attribute definitions of BOM item definitions.
<input checked="" type="checkbox"/>	BOM Attribute Mapping	Complete	This table stores the mapping between configuration attributes and BOM attributes or commerce line attributes.
<input checked="" type="checkbox"/>	BOM Attribute Translation	Complete	This table stores the translations of BOM attribute definitions.
<input type="checkbox"/> Select All			

Figure 6: Bills of Materials Tables Page

The table **Name** link provides access to the **Edit BOM Table Definition** page.

EDIT BOM TABLE DEFINITION PAGE

The **Edit BOM Table Definition** page allows the administrator to map CPQ Data Tables populated with customer BOM data to CPQ platform BOM Mapping Tables. Each BOM table has a corresponding **Edit BOM Table Definition** page to enable mapping. If the Data Table column names and data types match the default columns, column mapping is automatic. If not, the administrator must map each column manually. CPQ Cloud provides sample table definitions for each BOM table, which define the required schema. Administrators can acquire sample tables from the **Download Sample** link.

Edit BOM Table Definition

BOM Table Definition

Name: BOM Item Definition

Table Name: ▼

[Download Sample](#)

Column Name	Data Type	Description	Column Mapping
Variable Name	String	The natural key column of this table.	VariableName ▼
Sequence Number	Integer	BOM item sequence	SequenceNum ▼
Item ID	String	BOM item ID	ItemId ▼
Name	String	Display name	Name ▼
Item Type	String	BOM item type	ItemType ▼
Part Number	String	The part number of the BOM item	PartNumber ▼
Default Quantity	Float	The default quantity of the BOM item	DefaultQuantity ▼
Optional	String	Whether the BOM item is optional. Valid values: Y or N.	Optional ▼
Sales Item	String	Whether the item is a sales item. Valid values: Y or N.	SalesItem ▼
Parent Variable Name	String	The variable name of the hierarchical parent BOM item	ParentVariableName ▼
Root Variable Name	String	The variable name of the root BOM item	RootVariableName ▼
Effective From	String	The effective from date	EffectiveFrom ▼
Effective To	String	The effective to date	EffectiveTo ▼
Manufacturing Item	String	Whether the item is a manufacturing item or not. Valid values: Y or N.	ManufacturingItem ▼

Figure 7: Edit BOM Table Definition Page for BOM Item Definition Table

BOM ROOT ITEMS ADMINISTRATION LIST PAGE

After implementing BOM Mapping, the **BOM Root Items Administration List** page allows administrators to validate the BOM Item Tree containing BOM items, BOM item attributes, and BOM attribute translations. **The BOM Root Items Administration List** page displays the BOM root item names, part numbers, and item IDs. The root item is the highest level in the BOM tree and is equivalent to a CPQ Cloud model.

Access the **BOM Root Items Administration List** page from **Admin > Products > BOM > BOM Root Items List**.

Variable Name	Name	Part Number	Item ID
LP94777	Laptop ATO MODEL	LP94777	1000
DS92777	Custom Sentinel Desktop	DS92777	2000

Figure 8: BOM Root Items List Page

Variable Name links provide access to the **BOM Item Tree Administration** page.

BOM ITEM TREE ADMINISTRATION PAGE

The **BOM Item Tree Administration** page displays the expanded hierarchy and BOM definition information for a root BOM item, child items, and grandchild items. If there are any mapping errors, this page displays error indicators. To view validation errors refer to [BOM Item Tree with Validation Errors](#).

BOM Item Tree Administration										BOM Root Item:LP94777
Order	Variable Name	Name	Part Number	Item ID	Item Type	Sales Item	Manufacturing Item	Optional	Effective From	Effective To
1000	LP94777	Laptop ATO MODEL	LP94777	1000	Standard Item	Y	Y	N		
1002	LAPPRO1101	Intel Dual Core	PRO1101	1101	Standard Item	Y	Y	N		
1003	LAPPRO1109	Amd	PRO1109	1109	Standard Item	Y	Y	N		
1005	SCR1211	11 inch screen	SCR1211	1211	Standard Item	Y	Y	N		
1006	SCR1215	15 inch screen	SCR1215	1215	Standard Item	Y	Y	N		
1007	SCR1217	17 inch screen	SCR1217	1217	Standard Item	Y	Y	N		
4000	ET4000	Etching Service	ET4000	4000	Standard Item	Y	Y	N		
4001	ET4001	CPQ Etching Service	ET4001	4001	Standard Item	Y	Y	N		
4001	ET4002	BOM Etching Service	ET4002	4001	Standard Item	Y	Y	N		
4001	ET4003	ZOC Etching Service	ET4003	4001	Standard Item	Y	Y	N		

Figure 9: BOM Item Tree Administration Page

Variable Name links provide access to the **BOM Item Administration** page.

BOM ITEM ADMINISTRATION PAGE

The **BOM Item Administration** page displays definition information for the selected BOM item. It also displays associated attributes, if attribute definitions are provided.

BOM Item Administration		BOM Root Item:LP94777			
BOM Item					
*Variable Name:	BP3050				
*Name:	5.0 mA Battery				
*Part Number:	BP3050				
*Item ID:	3050				
Sequence Number:	2021				
Item Type:	Standard Item				
Default Quantity:	1.0				
Optional	<input type="checkbox"/>				
Sales Item	<input checked="" type="checkbox"/>				
Manufacturing Item	<input checked="" type="checkbox"/>				
Parent Variable Name:	BP3000				
Effective From:					
Effective To:					
BOM Attribute List					
Variable Name	Name	Data Type	Values	Effective From	Effective To
BSN3094	BatterySerialNumber	Integer			
BSO3902	BatteryOrientation	String	H~V		

Figure 10: BOM Item Administration Page

If the BOM Attribute Definition Table is active and the selected BOM item has attributes, the **Variable Name** links provide access to the **BOM Attribute Administration** page to view attribute details.

BOM ATTRIBUTE ADMINISTRATION PAGE

The **BOM Attribute Administration** page displays the attribute values for the selected BOM attribute.

BOM Attribute Administration **BOM Item:BP3050**

BOM Attribute

*Variable Name: BSO3902
Name: BatteryOrientation
Data Type: String
Values: H~V
Display Values: Horizontal~Vertical
Effective From:
Effective To:

[Back to Top](#)

Figure 11: BOM Attribute Administration Page

If the BOM Attribute Translation Table is active and the selected BOM attribute has translation, click **Translations** to access the attribute translations.

BOM ATTRIBUTE TRANSLATION ADMINISTRATION PAGE

The **BOM Attribute Translation Administration** page shows the language, the translated attribute name and display values.


BOM Attribute Translation Administration **BOM Attribute:BatteryOrientation**

Language	Name	Display Values
English	BatteryOrientation	Horizontal~Vertical
Chinese (Simplified) [China]	电池方向	H~V
Japanese [Japan]	バッテリーの向き	H~V
German	Batterie- Orientierung	H~V

Figure 12: BOM Attribute Translation Administration Page

The next section provides information on BOM item validation errors. The error indicators allow administrators to isolate BOM definition errors.

BOM ITEM TREE WITH VALIDATION ERRORS

If there are any validation errors in the BOM Item Tree, the **BOM Item Tree Administration** page displays an error message, at the top of the page. Items preceded by an error indicator  denote errors exist for the BOM item, its child items, or grandchild items. Administrators can examine descending elements of the BOM item definition to see error details.


BOM Item Tree Administration										BOM Root Item:LP94777	
Order	Variable Name	Name	Part Number	Item ID	Item Type	Sales Item	Manufacturing Item	Optional	Effective From	Effective To	
1000	LP94777	Laptop ATO MODEL	LP94777	1000	Standard Item	Y	Y	N			
1002	LAPPRO1101	Intel Dual Core	PRO1101	1101	Standard Item	Y	Y	N			
1003	LAPPRO1109	Amd	PRO1109	1109	Standard Item	Y	Y	N			
1005	SCR1211	11 inch screen	SCR1211	1211	Standard Item	Y	Y	N			
1006	SCR1215	15 inch screen	SCR1215	1215	Standard Item	Y	Y	N			
1007	SCR1217	17 inch screen	SCR1217	1217	Standard Item	Y	Y	N			
4000	ET4000	Etching Service	ET4000	4000	Standard Item	Y	Y	N			
4001	ET4001	CPQ Etching Service	ET4001	4001	Standard Item	Y	Y	N			
	ET4002	BOM Etching Service	ET4002	4001	Standard Item	Y	Y	N			
4001	ET4003	ZOC Etching Service	ET4003	4001	Standard Item	Y	Y	N			

Figure 13: BOM Item Tree with Validation Errors

NOTE: Errors must be resolved before setting up BOM Mapping Configuration rules.

DECLARE UTIL FUNCTION

To utilize BOM Mapping with Subscription Ordering, administrators select a BML utility function invoked during the Configuration Save event. The recommended utility function compares a Configuration with projected assets and saves the results to Commerce transaction lines. For additional information, refer to the Asset-Based Ordering Implementation Guide.

Declare BML Util Function

BML Util Function: Please select the BML Util Function which needs to be executed on Saving Configuration

Figure 14: Declare BML Util Function

BOM MAPPING RULES

Administrators create BOM Mapping Configuration rules at the Model level. BOM Mapping rules associate Configuration attributes to the BOM items. Use simple Table-Based Rules, advanced BML-Based Rules, or both for BOM Mapping. A new BOM Mapping rule type is available from the **Navigation** drop-down menu on the **Model Administration List** page.

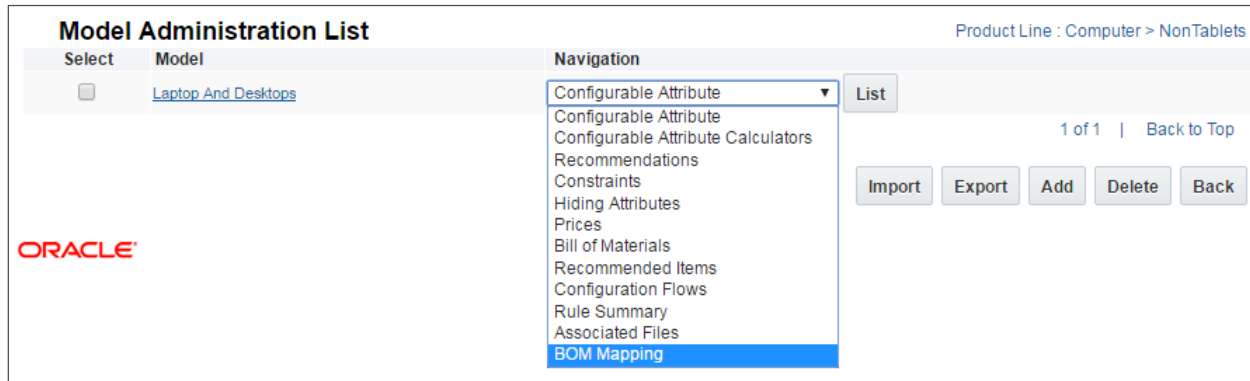


Figure 15: Model Administration - BOM Mapping

Selecting **Navigation > BOM Mapping** and **List** displays the **BOM Mapping: Rules List** page.

This page shows the order the rules actuate, the Name, and Status of BOM Mapping rules. Administrators can add and manage BOM Mapping rules as well as edit their translation.

The screenshot shows the 'BOM Mapping: Rules List' page. At the top right, it says 'Model : Computer > NonTablets > Laptop And Desktops'. Below the title, there is a table with the following columns: 'Select', 'Order', 'Name (Variable Name)', 'Status', and 'Overall Status'. The table contains five rows of data. Below the table, there are buttons for 'Translations', 'Add', 'Update', 'Delete', and 'Back'. At the bottom right, it says 'Back to Top'.

Select	Order	Name (Variable Name)	Status	Overall Status
<input type="checkbox"/>	1	Laptop BOM Mapping (laptopBOM)	Active	Active
<input type="checkbox"/>	2	Desktop BOM (desktopBOM)	Active	Active
<input type="checkbox"/>	3	BML Laptop Mapping C2B (bMLLaptopMappingC2B)	Active	Active
<input type="checkbox"/>	4	BML laptop B2C (bMLLaptopB2C)	Active	Active
<input type="checkbox"/>	5	BML Two Way Mapping (bMLTwoWayMapping)	Active	Active

Figure 16: BOM Mapping: Rules List

Selecting the **Variable Name** link of a Model opens the associated **BOM Mapping Rule** page.

This page displays the rule Name, Variable Name, Status, effective dates, Condition Type, Rule Type, the Target BOM, and Target Commerce Process.

Simple Table-Based BOM Mapping rules use the BOM Mapping tables to declaratively associate the Configuration attributes to BOM items. BOM Mapping requires the Target BOM action. Select one of the Root BOM items as the Target BOM to create based upon Configuration selections. Specify the Target Commerce Process if more than one is implemented.

The screenshot displays the 'BOM Mapping: Laptop BOM Mapping' configuration page. It is organized into several sections:

- BOM Mapping: Laptop BOM Mapping**: This section contains fields for 'Name' (Laptop BOM Mapping), 'Variable Name' (laptopBOM), and 'Description'. To the right, there is a 'Status' section with radio buttons for 'Active' (selected), 'Internal', and 'Inactive'. A link for 'Edit Start/End Dates' is also present.
- Condition**: This section shows 'Condition Type' set to 'Always True'.
- Action: (Define BOM Mapping)**: This section includes a descriptive text: 'BOM Mapping Rule enables you to map between BOM items and buyer-configured values.' Below this, 'Rule Type' is set to 'Simple (Table-based)'. 'Target BOM' is set to 'LP94777' and 'Target Commerce Process' is set to 'Quotes'. A 'Save and View Details' button is located at the bottom of this section.

At the bottom of the page, there is a row of utility buttons: 'Printer Friendly Version', 'Email a Copy', 'Translations', 'Apply', 'Update and Back', and 'Back'.

Figure 17: BOM Mapping Rule Page

REFERENCE INFORMATION

This section provides BOM Mapping use cases, system attributes, BML functions, and web service updates.

BOM MAPPING USE CASES

The versatility of the BOM Mapping feature allows administrators to choose among a variety of options for implementation. Administrators must activate and map only those BOM Mapping tables needed to support their requirements. The table below summarizes several approaches to leverage the BOM Mapping feature.

Usage Example	Usage Explanation	Active Tables
Basic BOM Integration	Capture complex BOM structures and fulfillment fields. Identify associations between BOM items and Configuration attributes using only advanced BML-Based Rules.	BOM Item Definition
BOM Mapping Integration	Integrate with fulfillment systems, such as Oracle EBS, that do not use BOM item attributes. Identify BOM item associations using simple Table-Based Rules. Advanced BML-Based Rules are optional and can further refine Configuration attributes.	BOM Item Definition BOM Item Mapping
BOM Mapping with Attributes Appended in String Variables	In addition to the BOM Mapping integration, this use case adds BOM attributes to BOM item lines. Identify BOM item associations using simple Table-Based Rules, and BOM attributes using advanced BML-Based Rules.	BOM Item Definition BOM Item Mapping BOM Attribute Definition
BOM Mapping using Attributes to Set Transaction Line Attributes	This scenario uses BOM Attribute Mapping to set the values of Commerce line attributes, and does not use the BOM Attribute Definition table. Identify BOM item associations using simple Table-Based Rules. Advanced BML-Based Rules are optional and can further refine Configuration attributes.	BOM Item Definition BOM Item Mapping BOM Attribute Mapping
Full-Service BOM Mapping without Attribute Translations	Integrate with fulfillment systems, such as Siebel, that support BOM item attributes. Identify BOM item and attribute associations using simple Table-Based Rules. Advanced BML-Based Rules are optional and can further refine Configuration attributes.	BOM Item Definition BOM Item Mapping BOM Attribute Definition BOM Attribute Mapping
Full-Service BOM Mapping	Integrate with fulfillment systems that support BOM item attributes with translations. Identify BOM item, attribute, and attribute translation associations using simple Table-Based Rules. Advanced BML-Based Rules are optional and can further refine Configuration attributes.	BOM Item Definition BOM Item Mapping BOM Attribute Definition BOM Attribute Mapping BOM Attribute Translation

BOM MAPPING SYSTEM ATTRIBUTES

Several new system attributes support the BOM Mapping feature. Administrators can use these attributes to display the BOM hierarchy or hierarchy relationships in the Commerce Transaction Line Item Grid user interface. For more information on these attributes and their role in BOM Mapping and Subscription Ordering, refer to the BOM Mapping Implementation Guide and the Asset-Based Ordering Implementation Guide.

Name	Variable Name	Type	Description
Line Item BOM ID	<code>_line_bom_id</code>	Text	The BOM item instance id.
Line Item BOM ID	<code>_line_bom_parent_id</code>	Text	The parent BOM item instance id.
Line BOM Part Number	<code>_line_bom_part_number</code>	Text	The part number of the BOM item. Only applicable to the model line.
Line Item BOM Attributes	<code>_line_bom_attributes</code>	Text	BOM attributes, stored as a JSON string.
Line BOM Item Quantity	<code>_line_bom_item_quantity</code>	Integer	The BOM item line quantity. This is the unexploded line quantity, whereas <code>_price_quantity</code> stores the exploded quantity.
Line BOM Level	<code>_line_bom_level</code>	Integer	The BOM item depth (level) in the quote. The value is 0 for the root BOM item.
Line BOM Effective Date	<code>_line_bom_effective_date</code>	Date	BOM Effective Date. If null, it is interpreted as the current time.

BML FUNCTIONS FOR BOM MAPPING

In release 2016 R1, CPQ Cloud delivers the following BML functions to support the BOM Mapping feature: `getbom`, `savebom`, `convertbomtoflat`, and `convertbomtohier`.

GET BOM FUNCTION

getbom			
Description	<p>For fulfillment system integrations, the <code>getbom</code> function retrieves the saved sales BOM or manufacturing BOM from a quote, to submit to the fulfillment system for order fulfillment.</p> <p>For Subscription Ordering, the <code>getbom</code> function retrieves the saved sales BOM from open orders.</p>		
Parameters	<code>bsId</code>	Integer	Use this parameter to specify the Commerce Transaction ID.
	<code>lineNumber</code>	Integer	Use this parameter to specify the document number of the model line. The line number also represents the root BOM line in the quote.
	<code>lineFields</code>	String	Use this parameter to identify additional line attributes fetched from the quote line, then stored in the returned BOM instance.
			Optional, the default value is null if not provided.
	<code>validateBomModel</code>	Boolean	Use this parameter to validate the returned BOM against the latest BOM item definition. Validation will:
			<ul style="list-style-type: none"> Verify the BOM instance tree (parts and hierarchy) against the BOM item definition. Populate the BOM item variable names. Correct the BOM instance hierarchy according to the latest definition. Exclude items that no longer exist in the latest definition.
	<code>flattenChildItems</code>	Boolean	Use this parameter to flatten child items and return all descendant BOM items as direct children of the root BOM item.
Optional, the default value is false if not provided.			
<code>isSalesBom</code>	Boolean	This parameter returns a sales BOM if true, and a manufacturing BOM if false.	

getbom		
		Optional, the default is true if not provided.
Syntax	<pre>Json getbom(Integer bsId, Integer lineNumber [, String[] lineFields [, Boolean validateBomModel [, Boolean flattenChildItems [, Boolean isSalesBom]]]])</pre>	
Sample Input	bsId	18430319
	lineNumber	2
Sample Return	<pre>{ "partNumber": "part49", "quantity": 10, "id": "BOM_root", "parentId": "", "attributes": {}, "fields": { "_line_bom_level": "0" }, "explodedQuantity": 10, "category": "sales", "variableName": "root", "definition": { "SequenceNum": 814, "ItemId": "814", "ItemType": "Standard Item", "Optional": "Y" }, "children": [{ "partNumber": "part50", "quantity": 5, "id": "BOM_text_bom", "parentId": "BOM_root", "attributes": {}, "fields": { "_line_bom_level": "1" }, "explodedQuantity": 50, "variableName": "text_bom", "definition": { "SequenceNum": 815, "ItemId": "815", "ItemType": "Standard Item", "Optional": "Y" } }] }</pre>	

SAVE BOM FUNCTION

savebom			
Description	<p>The savebom function saves a BOM into a quote without Configuration attributes and returns the document number of the saved quote. For Subscription Ordering, the savebom function saves discontinued assets into a quote.</p> <p>Do not invoke this function from the quote modify action; as the modify action and savebom function will compete to update the same quote. Use reconfigure for the saved BOM instance.</p>		
Parameters	bsID	Integer	Use this parameter to specify the Commerce Transaction ID.
	bomJson	JSON	Use this parameter to hold the BOM instance JSON data.
Syntax	Integer savebom(Integer bsId, Json bomJson)		
Sample Input	bsId	18430319	
	bomJson	<pre>{ "partNumber": "part49", "id": "BOM_root", "quantity": 10, "parentId": "", "attributes": {}, "fields": {"_line_bom_level": "0"}, "explodedQuantity": 10, "category": "sales", "variableName": "root", "definition": { "SequenceNum": 814, "ItemId": "814", "ItemType": "Standard Item", "Optional": "Y" }, "children": [{ "partNumber": "part50", "quantity": 5, "id": "BOM_text_bom", "parentId": "BOM_root", "attributes": {}, "fields": {"_line_bom_level": "1"}, "explodedQuantity": 50, "variableName": "text_bom", "definition": { "SequenceNum": 815, "ItemId": "815", "ItemType": "Standard Item", "Optional": "Y" } } }] }</pre>	
Sample Return	5		

CONVERT A HIERARCHICAL BOM INTO A FLATTENED BOM FUNCTION

convertbomtoflat			
Description	The convertbomtoflat function converts a hierarchical BOM into a flattened BOM. A flat BOM stores all descendants as direct children, including children, grandchildren, etc. Flattened BOMs are easier to process.		
Parameter	bomJson	JSON	Use this parameter to hold the JSON target.
Syntax	Json convertbomtoflat(Json bomJson)		
Sample Input	bomJson	<pre>{ "partNumber": "part1", "quantity": 1, "id": "Bom1", "parentId": "", "children": [{ "partNumber": "part2", "quantity": 2, "id": "Bom2", "parentId": "", "children": [{ "partNumber": "part4", "quantity": 4, "id": "Bom4", "parentId": "" }, { "partNumber": "part5", "quantity": 5, "id": "Bom5", "parentId": "" }] }, { "partNumber": "part3", "quantity": 3, "id": "Bom3", "parentId": "" }] }</pre>	
Sample Return	<pre>{ "partNumber": "part1", "quantity": 1, "id": "Bom1", "parentId": "", "children": [{ "partNumber": "part2", "quantity": 2, "id": "Bom2", "parentId": "Bom1" }, { "partNumber": "part3", "quantity": 3, "id": "Bom3", "parentId": "Bom1" }, { "partNumber": "part4", "quantity": 4, "id": "Bom4", "parentId": "Bom2" }, { "partNumber": "part5", "quantity": 5, "id": "Bom5", "parentId": "Bom2" }] }</pre>		

CONVERT A FLATTENED BOM TO HIERARCHICAL BOM FUNCTION

convertbomtohier			
Description	The convertbomtohier function converts a flattened BOM into a hierarchical BOM. Occasionally, administrators flatten hierarchical BOMs for easier processing; this function returns the processed flattened BOM back into a hierarchical BOM.		
Parameter	bomJson	JSON data type	Use this parameter to hold the JSON target.
Syntax	Json convertbomtohier(Json bomJson)		
Sample Input	bomJson	<pre>{ "partNumber": "part1", "quantity": 1, "id": "Bom1", "parentId": "", "children": [{ "partNumber": "part2", "quantity": 2, "id": "Bom2", "parentId": "Bom1", }, { "partNumber": "part3", "quantity": 3, "id": "Bom3", "parentId": "Bom1", }, { "partNumber": "part4", "quantity": 4, "id": "Bom4", "parentId": "Bom2", }, { "partNumber": "part5", "quantity": 5, "id": "Bom5", "parentId": "Bom2" }] }</pre>	
Sample Return	<pre>{ "partNumber": "part1", "quantity": 1, "id": "Bom1", "parentId": "", "children": [{ "partNumber": "part2", "quantity": 2, "id": "Bom2", "parentId": "", "children": [{ "partNumber": "part4", "quantity": 4, "id": "Bom4", "parentId": "" }, { "partNumber": "part5", "quantity": 5, "id": "Bom5", "parentId": "" }] }, { "partNumber": "part3", "quantity": 3, "id": "Bom3", "parentId": "" }] }</pre>		

CONFIGURATION WEB SERVICE TO SUPPORT BOM MAPPING

This release introduces the optional `bomprice` element to the Configuration web service (v1 and v2). The response includes the BOM price for sites using BOM Mapping if the BOM price is not zero.

Sample Response:

```
<bm:price>
  <bm:bomPrice>$16.0000</bm:bomPrice>
  <bm:totalPrice>$45.0000</bm:totalPrice>
</bm:price>
```

The web service WSDL, upon regeneration, includes the newly introduced "bomPrice" optional element:

```
<xsd:complexType name=" ">
  <xsd:sequence>
    ...
    <xsd:element maxOccurs="1" minOccurs="0" name="bomPrice" nillable="true"
type="xsd:string"/>
    <xsd:element maxOccurs="1" minOccurs="1" name="totalPrice" nillable="false"
type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

STEPS TO ENABLE

BOM Mapping is available on all 2016 R1 sites. Refer to the BOM Implementation Guide, available on [My Oracle Support](#), for detailed instructions.

The following items outline setup activities:

1. Capture BOM item definitions in CPQ Data Tables.
2. Activate BOM tables per business requirements.
3. Map BOM Data Tables to BOM platform tables.
4. Resolve validation errors that occur during set up.
5. Build Configuration rules to associate BOM items and attributes to CPQ Configuration and Commerce.

TIPS AND CONSIDERATIONS

- Administrators can use Data Table Import or Bulk Upload to import BOM Data Tables.
- Favorites and Shopping Cart features do not support BOM Mapping.

NOTE: BOM Mapping integrates with CPQ Configuration to employ and merge BOM items and attributes. Therefore, CPQ Configuration setup is required prior to BOM Mapping Implementation.

NOTE: This document does not cover fulfillment system integration, such as with the Oracle eBusiness Suite (EBS). Refer to the CPQ-EBS Integration Implementation Guide for implementation instructions.

KEY RESOURCES

Refer to the following resources for additional information:

- [BOM Mapping Implementation Guide](#): Provides detailed information about how to implement the BOM Mapping feature available in the 2016 R1.
- CPQ Cloud Online Help: Refer to the BOM Mapping topics.
- [CPQ-EBS Integration Implementation Guide](#): Provides detailed information for integrating Oracle eBusiness Suite (EBS) to CPQ Cloud.

SUBSCRIPTION ORDERING

Companies use Subscription Ordering, sometimes known as asset-based ordering, to sell tangible assets or subscriptions for services delivered over a period of time. The Subscription Ordering feature in CPQ Cloud 2016 R1 supports subscription and asset-based products by recording fulfilled lines as assets and providing customers with robust subscription management tools to address future-dated modifications and generation of follow-on orders.

With the following features, administrators can provide sales users with the ability to review, modify, and renew subscription or asset-based products. When used together, these features provide the functionality and guidance needed to successfully implement a CPQ Cloud 2016 R1 Subscription Ordering solution.

- **BOM Mapping Rules:** Enable Subscription Ordering by setting up BOM Mapping Rules to associate Bills of Material with CPQ Configuration attributes.
- **REST APIs:** Create, query, or modify assets using REST APIs. For additional information, refer to the [REST APIs](#) section of this document.
- **Customer Assets Page:** Display, search, and administer assets based on the order request date.
- **Local Asset Repository:** Store asset information locally in the CPQ Cloud repository.
- **Asset-Based Ordering Package and Implementation Guide:** Use the Asset-Based Ordering (ABO) Package and the steps outlined in the Asset-Based Ordering Implementation Guide as reference materials for implementing a complete subscription management solution.

ASSETS

With the implementation of Subscription Ordering CPQ Cloud creates and submits orders for the fulfillment of subscription products. When these products are fulfilled, the fulfillment system can invoke a CPQ Asset API to create an asset in the CPQ asset repository. Once created, assets can be viewed by sales users on the new **Customer Assets** page. Sales users can also modify or terminate assets and services, by placing subsequent Commerce transaction orders to modify or terminate the asset.

Add Subscription Ordering attributes, functions, and other elements to your Commerce Process by migrating the Asset-Based Ordering Package or manually adding them as described in the CPQ Cloud Asset-Based Ordering Implementation Guide.

- **Instance Id:** Provides the universal asset instance identifier generated during asset creation.
- **Instance Name:** Displays the user-friendly name or identifier for an asset, such as a cell phone number.
- **Action:** Identifies the action to be performed on the asset for the transaction line. The action options include Add, Delete, Update, Terminate, and “–” (represents no change). Action codes can be displayed to sales users for their confirmation of the changes to be made.
- **Request Date:** Provides the requested date for the transaction line. Customers can enter future dates. If a request date is not provided, the current date is used.

- **Fulfillment Status:** Identifies the fulfillment status for the transaction line. The status options include Created, Being Fulfilled, Fulfilled, and Cancelled.
- **Attributes Summary:** Stores BOM attribute name-value pairs for the current transaction line during asset creation and updates.

NOTE: The Asset-Based Ordering package contains the attributes required to implement Subscription Ordering. Administrators are responsible for adding the attributes to their Commerce UI Layouts.

CUSTOMER ASSETS PAGE

The **Customer Assets** page is a new pre-defined page that displays to sales users the assets associated with a transaction’s Customer Account ID. The **Customer Assets** page provides the ability to view purchased, modified, and terminated assets; and to search, filter, and sort assets. The new UI Designer feature introduced in CPQ Cloud 2016 R1 can be used to customize the **Customer Assets** page.

To access the **Customer Assets** page, open a quote, select a customer, and click **Customer Assets**.



Product	Service Identifier	Start Date	End Date	Status	Date Added	Date Modified
MP-1000	MP-1000-36421093-1	9/30/2016	9/30/2017	Active	08/16/2016...	08/16/2016 12:4...
MP-1000	MP-1000-36417174-1	8/31/2016	2/28/2017	Active	08/09/2016...	08/09/2016 9:03...
MP-1000	MP-1000-36418023-1	8/31/2016	8/31/2017	Active	08/10/2016...	08/10/2016 12:2...
MP-1000	MP-1000-36419070-1	8/30/2016		Active	08/16/2016...	08/16/2016 5:01...
MP-1000	MP-1000-36417642-1	8/29/2016	8/29/2019	Active	08/10/2016...	08/10/2016 12:0...

Figure 18: Customer Assets Page

NOTE: The **Customer Assets** page displays active assets based on the request date associated with the current order. The page does not display assets scheduled to start on a later date.

For example: Assume a customer adds a sports network to their cable service on Aug. 1. This creates *Asset 1* with a future request date of Aug 1. The customer then calls back the next day and adds a cooking network to their cable service on July 1, which is order 2. The **Customer Assets** page does not display *Asset 1* on the second order, which has a request date of July 1. The **Customer Assets** page only displays *Asset 1* for orders with Aug. 1 or later as the request date.

ASSET CREATION

The Update Assets action provided in the ABO Package supports the creation and modification of an asset. Asset creation generates a traceable item that integrates with your fulfillment system. After asset creation, customers can view and maintain subscription services through transactions.

For example: When a Sales Representative uses Commerce to add relevant products to a transaction, the transaction lines display a status of 'Created'. When the transaction is submitted as an order to the fulfillment system, the status of the transaction line changes from "Created" to "Being Fulfilled". When the order is fulfilled in the back-end system, the fulfillment system notifies CPQ Cloud in an integration flow. The transaction lines status then changes from "Being Fulfilled" to "Fulfilled" and the asset is recorded in the CPQ local asset repository.

	Fulfill Status	Request Date	Instance Name	Action	Attribute	Instance Id
☰ ☐	Created ▼	06/01/2016	Root Item	Add ▼		001
☰ ▶ ☐	▼		Parent 1	Add ▼		002
☰ ▶ ☐	▼		Child 1.1	Add ▼		003
☰ ▶ ☐	▼		Child 1.2	Add ▼		004

Figure 19: Sample Order for Asset Creation

ASSET MODIFICATION

Asset modification allows customers to modify fulfilled assets, with the option to modify at a future date.

Complete the following steps to modify an asset:

Create a transaction and select the request date for the modification.

1. Navigate to the **Customer Assets** page.
 - Open the CPQ Commerce Transaction UI.
 - Click **Customer Assets**.
2. Select the appropriate asset, and click **Modify**.
The **Configuration** page appears.
3. Make the appropriate revisions and update the current transaction.

The following asset modification options are available from the **Customer Assets** page:

- **Add** a new component to the projected asset, excluding requests with future start dates.
- **Delete** a component from an asset.
- **Update** an asset component through actions such as BOM attribute changes.

NOTE: A hyphen (–) displays for lines without updates.

RECONFIGURE

Use the Reconfigure action to update a quote prior to fulfillment. Reconfigure compares the projected asset with a reconfigure order line to reflect user-intended net changes in a subscription or asset. Pending update order lines that occur before the reconfigure requested date are included in the comparison. Pending order lines have one of the following conditions:

- The item is “Being Fulfilled” in another order.
- The item is “Being Fulfilled” or “Created” in the current order.

FOLLOW-ON ORDERS

A Follow-On Order is a change to an existing order that has not yet been fulfilled. When a follow-on order is created, Subscription Ordering automatically creates the action codes for each transaction line based upon the difference between the expected state of the asset on the request date and the new configuration.

ASSET TERMINATION

Use the Terminate action on the **Customer Assets** page to end a subscription. For example: Assume a customer terminates a cable television subscription. The Sales Representative can create a Commerce transaction for the customer and specify the request date for the termination. When CPQ Cloud displays the **Customer Assets** page, the sales user can select the asset to be terminated, and click the Terminate action. The transaction is displayed in Commerce with the appropriate action codes to terminate the subscription. When the Terminate action completes, the end date of the asset becomes the date the customer requested the subscription termination.

STEPS TO ENABLE

Refer to the Asset-Based Ordering Implementation Guide for detailed instructions on how to implement this feature.

KEY RESOURCES

Refer to the following resources for additional information:

- [BOM Mapping Implementation Guide](#): Provides detailed information about how to implement the BOM Mapping feature available in the 2016 R1.
- CPQ Cloud Online Help: Refer to the Subscription Ordering and BOM Mapping topics.
- [Asset-Based Ordering Implementation Guide](#): Provides detailed information about how to use and implement Subscription Ordering.
- Asset-Based Ordering Package

CONTRACT NEGOTIATION

The Contract Negotiation feature available in CPQ Cloud 2016 R1 supports the negotiation of a contract with a customer by integrating CPQ Cloud document handling with the track changes features of Microsoft Word. Enhancements to CPQ Document Designer and new REST services provide the capability to compare and merge contracts, highlight differences in two versions of a contract, and accept or reject specific changes made by customers or internal organizations such as legal. Administrators can leverage these capabilities to implement a highly customizable Contract Negotiation solution.

The 2016 R1 Contract Negotiation functionality is described below:

- Use a Single-Language Document Designer template to create a contract.
- Capture versions of the contract that have been modified by customers or internal reviewers.
- Generate a list of differences between documents created from the same Document Designer template using the [DOCX Compare REST API](#).
- Filter the list of differences between document versions and merge approved changes into a new document version using the [DOCX Merge REST API](#).

NOTE: Available in 2016 R1, administrators can use Microsoft Word to track changes in .DOCX output files and integrate with DocuSign to allow customers to electronically sign and approve documents. For additional information, refer to the Document Designer Enhancements addressed later in this document.

ENABLE CONTRACT NEGOTIATION

By default, the Contract Negotiation feature in CPQ Cloud 2016 R1 is disabled. When the feature is enabled, a Contract checkbox is available when creating new Document Designer templates as well as in the Document Properties pane of existing templates.

Complete the following steps to enable Contract Negotiation:

1. Click **Admin** to go to the Admin Home page.
2. Click General Site Options in the General section.
The **Options-General** page appears.
3. Set the Enable Contract Generation in Document Designer option to Yes.

CREATE CONTRACT FROM SINGLE-LANGUAGE TEMPLATE

Begin using the 2016 R1 Contract Negotiation feature by selecting a single-language Document Designer template to use as a contract template.

Complete the following steps:

1. Navigate to Document Designer.
 - Click Admin to go to the Admin Home page.
 - Click **Document Designer** in the **Commerce and Documents** section.
2. From the **Document Designer Templates** page, select a single-language Document Designer template to use as a contract template.
3. From the Document Properties panel, select the **Contract** checkbox.

With the exception of the spacer element, the page element, and the column break element, all Document Designer elements are identified in the .DOCX output file as clauses.

NOTE: The name associated with each element becomes the clause name, which is later used to identify differences between document versions.

CAPTURE VERSIONS OF THE CONTRACT

Administrators can define CPQ Commerce file attachment attributes to capture each version of a contract required to support their specific workflow. Administrators can link these attributes to actions to compare and merge contract versions and expose these actions to the sales user in CPQ Cloud Commerce workflows.

GENERATE LIST OF DIFFERENCES BETWEEN DOCUMENT VERSIONS

The ability to generate a list of differences between document versions is supported by the DOCX Compare REST API. The API identifies differences in all elements except headers, footers, and heading styles.

For example: When a Sales Representative sends a contract to a customer, the customer reviews the contract and can either approve the contract or make modifications to the contract. When the latter occurs and the customer sends the modified contract back to the Sales Representative, the DOCX Compare REST API compares the clause tags in the original contract and the modified contract. During this comparison, the DOCX Compare REST API identifies updated content within clause tags and deleted content within clause tags and returns a list of differences.

NOTE: Generating a list of differences between two .DOCX files created from different single-language Document Designer templates is not supported by the DOCX Compare REST API. For additional information about the REST API, refer to the [REST API](#) section of this document.

MERGE APPROVED CHANGES

The ability to merge approved changes into a .DOCX contract document is supported by the DOCX Merge REST API.

For example: When a contract modified by a customer is uploaded into CPQ Cloud by a Sales Representative, the Sales Representative can view the edited contract, approve and reject changes made by the customer, and then merge and apply the approved changes to the finalized contract. The DOCX Merge REST API supports this activity by merging the approved changes into a target document.

NOTE: For additional information about the DOCX Merge REST API, refer to the [REST API](#) section of this document.

TIPS AND CONSIDERATIONS

Consider the following tips when using the new Contract Negotiation feature:

- Verify that the types of changes made to contracts in your standard workflow will be correctly handled in your test or development environment, before enabling the Contract Negotiation feature in production. Modifications may be necessary to contract Document Designer templates to refine your contract negotiation process and assure that it will address your requirements.
- Multi-language Document Designer templates are not supported as contract templates. The Contract checkbox in the Document Properties pane of Document Designer is only available for single-language templates.
- Clause names are hidden tags that do not appear when users preview or print a .DOCX output file. To view clause names when previewing or printing a .DOCX output file, un-check the Hide option in Microsoft Word.
- Background images and header and footer changes are not considered when comparing or merging documents.

KEY RESOURCES

Refer to the following resources for additional information:

- [Contract Negotiation Implementation Guide](#): Provides detailed information about how to use the two Contract Negotiation APIs available in the 2016 R1 release to create a customized Contract Negotiation solution.
- CPQ Cloud Online Help: Refer to the Document Designer and REST API topics.

SECURE DATA TABLE COLUMNS

In this release, Oracle CPQ Cloud is introducing a secure data type option for new columns in both new and existing Data Tables. Confidential client credentials are required to connect to PaaS applications and other external systems. Secure Data Table Columns provide a method for securely storing confidential credentials in CPQ Cloud. Secure columns always store the encrypted form of the data in the Data Table. The only way to access this data in its original, decrypted form is through BMQL.

The content in non-secure columns is in a decrypted (not encrypted) format. Data in secure columns is decrypted, masked, or encrypted dependent upon the context. The following examples display how “Password” appears for the different formats:

- Figure 20 displays a CSV file for import with decrypted data in SecureColumn.

	E	G	P
2	Name	PartNumber	SecureColumn
6	Laptop ATO MODEL	LP94777	Password

Figure 20: Decrypted Data

- Figure 21 displays a Data Table with masked data in SecureColumn.

Data		Schema		
Page Length:		50	▼	
#	⊗	Name	PartNumber	SecureColumn
1	⊗	Laptop ATO MODEL	LP94777	*****

Figure 21: Masked Data

- Figure 22 displays an exported CSV file with encrypted data in SecureColumn.

	E	G	P
2	Name	PartNumber	SecureColumn
6	Laptop ATO MODEL	LP94777	3HTiFSUa6n4dw1oDijQpUQ==

Figure 22: Encrypted Data

INCOMING DATA FORMAT FOR SECURE COLUMNS

The following table lists the expected incoming data format to create or update data in secure columns.

Incoming Data Format	Encrypted	Decrypted
Admin UI Table Editor		✓
Import via Admin UI		✓
*Bulk Upload (see note)		✓
Web Services		✓
Package Upload	✓	

OUTPUT DATA FORMAT FOR SECURE COLUMNS

The following table shows the secure column data output for various CPQ functions.

Output Data Format	Masked	Encrypted	Decrypted
Admin UI Table Editor	✓		
Export via Admin UI		✓	
*Bulk Download (see note)		✓	
Web Services Get API		✓	
Web Services Add/Update APIs		✓	
Package Download		✓	
Jython		✓	
Objects		✓	
bmql() select fields			✓

*** NOTE:** When a Data Table with a secure column is downloaded using a bulk download and the same content is then uploaded using a bulk upload, values are double encoded and result in changed values.

SECURE COLUMN DATA USAGE

Secure column data is not available for the following uses:

- Formulas: query() conditions
- Formulas: query() return column
- Table-Based rules: filter column
- Table-Based rules: Constraint message
- Table-Based rules: Constraint value
- Table-Based rules: Recommendation message
- Table-Based rules: Recommendation value
- Table-Based rules: Recommended item properties

If secure columns are referenced, the following BML statements generate errors:

- gettabledata() select fields
- gettabledata() where fields
- bmql() order by fields (dynamic)
- bmql() where fields (dynamic)

If a BML script contains one of the following BMQL statements with secure column references, errors are generated during a save:

- bmql() order by fields (static)
- bmql() where fields (static)

Web services criteria for Get and Delete APIs return a SOAP fault.

ADDING A SECURE COLUMN TO A DATA TABLE

1. Navigate to the Data Table Administration page.
 - Click **Admin** to go to the Admin Home page.
 - Click **Data Tables** in the **Developer Tools** section.
2. Open an existing Data Table. For instructions on creating a new Data Table, refer to the Data Tables “Manually Adding a Table” topic in CPQ Online Help.
3. Click the **Schema** tab in the right panel of the **Data Table Administration** page.

NOTE: The **Secure Data Type** is only available for new columns. The **Secure Data Type** option is not available when modifying existing columns.

4. Click **Add Column** to add a new Data Table column.

The **New Column** dialog box appears.

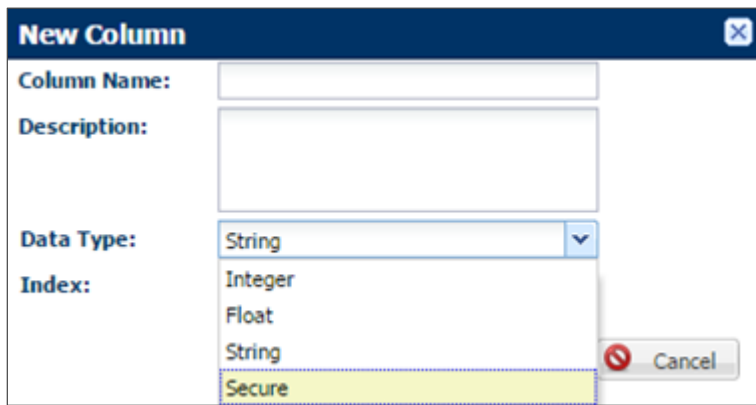


Figure 23: Data Table Column Secure Data Type

5. Enter a Column Name and Description.

NOTE: Column names can only contain alphanumeric characters and underscores. White space characters are not allowed.

6. Select the **Secure** from the **Data Type** drop-down menu.
7. Click **Add Column** to save the column.

STEPS TO ENABLE

Secure Data Table Columns are automatically available on all 2016 R1 sites.

TIPS AND CONSIDERATIONS

Consider the following tips when using the Secure Data Table Column feature:

- Data Tables can only have 50 string (text) columns. Secure columns count against that number. If a Data Table already has 50 text columns, administrators cannot add a secure column, without removing one of the text columns.
- Querying Secure Data Table Columns for Dynamic Pick Lists returns encrypted data.

KEY RESOURCES

CPQ Cloud Online Help: Refer to the Data Table topics.

EASY ADMINISTRATION

Administrators are the main force behind keeping a CPQ Cloud site up-to-date. Designed to make administration even easier, the CPQ Cloud 2016 R1 release includes new functionality in the following areas:

- [Single Select Pick Lists in Configuration](#)
- [UI Designer](#)
- [BML Enhancements](#)
- [Document Designer Enhancements](#)
- [Long Running Thread Diagnostics](#)

SINGLE SELECT PICK LISTS IN CONFIGURATION

As part of continuous improvements to Single Select Pick Lists, the following new features are available in CPQ Cloud 2016 R1.

- Apply hiding rules to a Single Select Pick List attribute
- Use the Related Rules tab to find references made from a Single Select Pick List
- Display a Single Select Pick List as an image grid
- Use an array set as the source of Single Select Pick List data

APPLY HIDING RULES TO A SINGLE SELECT PICK LIST ATTRIBUTE

In CPQ Cloud 2016 R1, administrators can apply hiding rules to Single Select Pick List attributes. Hiding rules consist of a condition and an action and are used in CPQ Cloud to hide select attributes when a pre-defined condition is met. Administrators can use Single Select Pick List attributes as the action for a hiding rule. However, Single Select Pick List attributes cannot be used as a condition for a hiding rule.

Use the **Hiding Rule: New Rule** page to define a hiding rule that uses Single Select Pick List attributes as the action. When the condition specified by the administrator is met, this triggers the hiding of the Single Select Pick List attributes (i.e. the action attributes) selected by the administrator.

Figure 24: Hiding Rule: New Rule Page

NOTE: Single Select Pick List attributes are only available for selection in the Action: (Define Attributes To Hide) area of the **Hiding Rule: New Rule** page. They are not available for selection in the Condition area of the page. If a hiding rule contains a Single Select Pick List, the Add Associated Recommendation Rule functionality is not available.

USE RELATED RULES TAB TO FIND REFERENCES MADE FROM A SINGLE SELECT PICK LIST

The **Related Rules** tab is used to view all of the rules that reference a particular attribute. The tab displays the name of the rule, the level of the rule within the hierarchy, the rule type, and the rule components. As shown below in Figure 25: Related Rules Tab, administrators can now use the **Related Rules** tab to find references made to the current attribute from a Single Select Pick List.

Label	Level	Rule Type	Components
All PF Level	All Products (All Products)	Recommendation	Action Attribute
New Hiding Rule	All Products (All Products)	Hiding Attribute	Hidden
Sample	All Products (All Products)	Single Select Pick List	Action Attribute

Figure 25: Related Rules Tab

Descriptions of each of the columns in the Related Rules tab are provided below:

- **Label:** The list of rules with which the attribute is associated.
- **Level:** The level of the rule within the product hierarchy (e.g. All Products, Product Family, Product Line, Model).
- **Rule Type:** The type of rule. The Single Select Pick List rule type is new in CPQ Cloud 2016 R1 and identifies references to the attribute from a Single Select Pick List.
- **Components:** The rule’s component, which uses this particular attribute. When Single Select Pick List is the Rule Type, this column identifies whether the reference is part of a filter for the Single Select Pick List or a pick map from the Single Select Pick List.

DISPLAY A SINGLE SELECT PICK LIST AS AN IMAGE GRID

In CPQ Cloud 2016 R1, an administrator can display a Single Select Pick List attribute as an image grid. In prior releases, Drop-down Menu was the only available display type for Single Select Pick List attributes.

To display Single Select Pick List attributes as an image grid:

1. Navigate to the Configuration Attributes Administration List page.
 - Click **Admin** to go to the Admin Home Page.
 - Click Catalog Definition in the Products section.
The **Supported Products** page appears.
 - In the **Navigation** column, select **Configuration Attribute** and then click **List**.
The Configuration Attributes Administration List page appears.
2. Click the name of an attribute that has Single Select Pick List in the **Attribute Type** column, or create a new attribute.
The Menu Attribute Editor page appears.
3. Set the **Image Menu** option to **Yes**.
4. Select **Grid** as the **Display Type**.

The screenshot shows the 'Menu Attribute Editor: Sample Attribute' interface. It is divided into several sections:

- Main Information:** Fields for Name (Sample Attribute), Variable Name (sampleAttribute), Data Type (Text), Image Menu (radio buttons for No and Yes, with Yes selected), Display Type (dropdown menu set to Grid), Array Type (No), Display Order (10), and Description (text area). An 'Edit HTML' button is present.
- Properties:** Fields for Set Type (Set), Auto Lock (checkbox), Required (checkbox), Hidden (checkbox), Auto Update (checkbox), Hide In Transaction (checkbox), Status (Active), and a link for [Show Start/End Dates].
- Image Menu Properties:** Fields for Label Location (Below Image), Columns (4), Rows (0 for auto) (3), Fix Dimension (Height), and Image Size (75 px).
- Single Select Pick List:** Fields for Domain (HardDrives), *Variable (HardDrive), *Display (HardDrive), and *Image (HardDrive).

Figure 26: Menu Attribute Editor

The following example shows how Single Select Pick List attributes as image grids display to the end user.



Figure 27: Single Select Pick List Attributes Displayed as Image Grids

Use the **Filter** text box on the **Menu Attribute Editor** to apply a filter to a Single Select Pick List attribute. When a filter is applied to a Single Select Pick List attribute and the filter value changes, the associated Single Select Pick List Image Grid dynamically changes based on the filter value.

NOTE: The size of the Filter text box now supports 4000 characters. In previous releases, the Filter text box supported 128 characters. For information about the syntax to use when applying a filter, refer to the Filters topic in the CPQ Cloud Online Help.

A screenshot of the 'Single Select Pick List' configuration interface. At the top, it shows 'Domain: Laptop_Selection', '*Variable: OperatingSystem', and '*Display: ScreenSize'. Below this is a 'Filter:' label followed by a large, empty text input box with a red border. Underneath the filter box are 'Picker Attribute:' and 'Model Attribute:' dropdown menus, an 'Add' button, and a 'Delete' button. A large empty rectangular area is at the bottom of the interface.

Figure 28: Filter Text Box on the Menu Attribute Editor

USE ARRAY SET AS SOURCE OF SINGLE SELECT PICK LIST DATA

Administrators can now select an array set as the source of Single Select Pick List data. In prior releases, Data Tables were the only available source of Single Select Pick List data. By selecting an array set as the domain, the array attributes are used as the Single Select Pick List options. In Figure 29: Configurable Array Set Editor, **Search Flow Array** is the name of the array set and the attributes shown in the **Associated Attributes** field are the array attributes.

Configurable Array Set Editor

Main Information

*Name: Search Flow Array

*Variable Name: searchFlowArray

Description:

Size Attribute: Search Flow Array Control

Display Type: Horizontal

Label Option: Display Only First Label

Prefix:

Suffix:

Set Association Information

Set Association:

Unassociated Attributes

Associated Attributes

- Search Flow Array Text Field
- Search Flow Array Float Field
- Search Flow Array Integer Field
- Search Flow Array Date Field
- Search Flow Array Currency Field
- Search Flow Array Boolean

Figure 29: Configurable Array Set Editor

Use the **Domain** field in a Single Select Pick List attribute's editor to select an array set or a Data Table to use as the source of the data. As shown in Figure 30: Attribute Editor, the **Domain** drop-down menu contains a list of Data Tables and array sets. The array sets are listed prior to the Data Tables, each of which is shown in alphabetical order. There is no displayed separator between the two collections of data sources.

NOTE: Data Tables in the **Domain** field now display as the actual name of the Data Table. In prior releases, the name of each Data Table contained the prefix "custom" followed by the name of the Data Table.

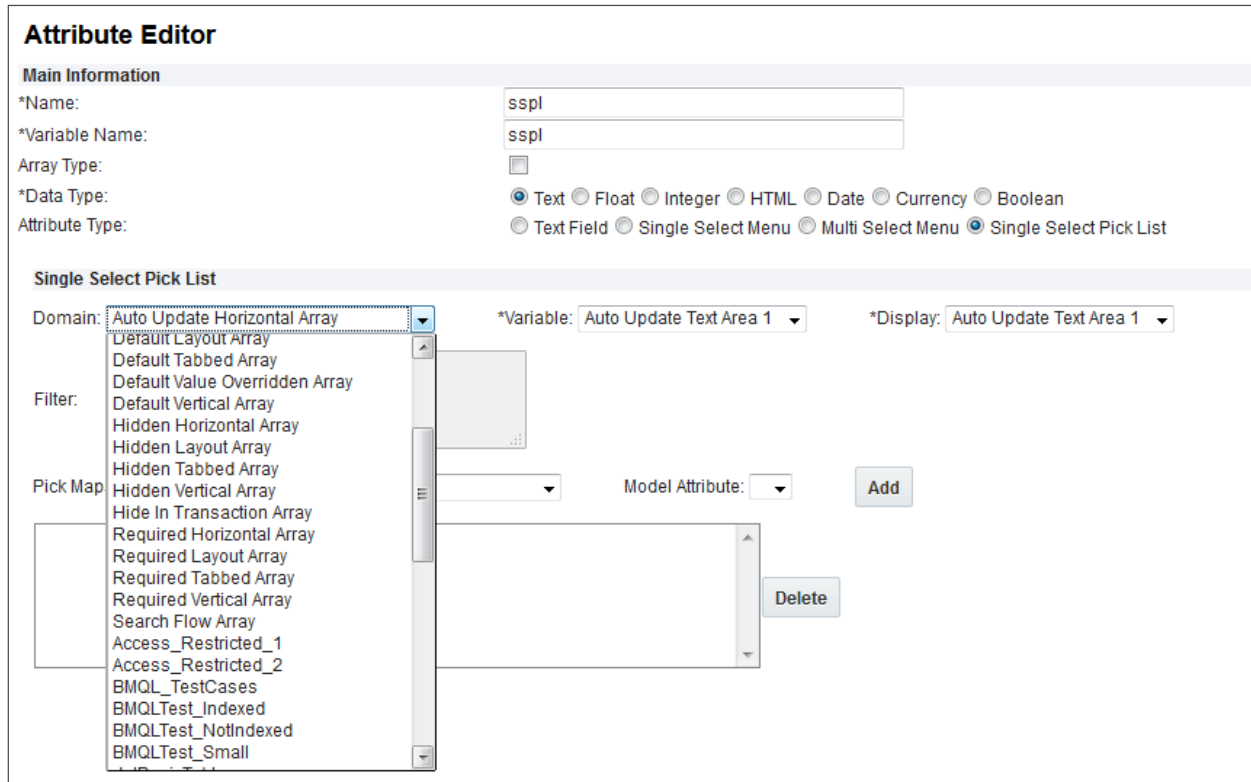


Figure 30: Attribute Editor

When an array set is selected from the **Domain** drop-down menu, the **Variable**, **Display**, and **Picker Attribute** fields automatically populate with all of the array attributes in the array set. The options in the **Model Attribute** field are automatically populated after the administrator selects an array attribute from the **Picker Attribute** drop-down menu.

NOTE: Filtering is not supported for array set Domains. Upon selecting an array set as the Domain, the filter is cleared and is not recoverable.

Figure 31: Attribute Editor

TIPS AND CONSIDERATIONS

Consider the following tips when using the new Single Select Pick List features:

- If the data in the Display or Variable name column is empty or specified as “null” in the Data Table, then pick maps do not work correctly. Single Select Pick maps will display “null” for empty values.
- When a Data Table contains a column name of “class” or “null”, an administrator can use the Data Table but cannot use the column. On the user side, Data Table columns named “class” or “null” break in Configuration.
- Numeric Configuration attributes used in a Single Select Pick List filter are now validated when the attribute focus is lost, which occurs when a user clicks away or uses the Tab key to go to the next attribute on the page. In previous releases, numeric Configuration attributes (e.g. integer or float) were not validated until the Configuration was saved or updated.
- In previous releases, Single Select Pick List Configuration attributes only showed the Display value in the pipeline viewer while Commerce showed both the Display value and the Variable name “Label[varname]” for similar attributes. In 2016 R1, Configuration now follows the same pattern as Commerce for its pipeline viewer for all Configuration attribute types.

KEY RESOURCES

CPQ Cloud Online Help: Refer to the Single Select Pick List topic.

UI DESIGNER

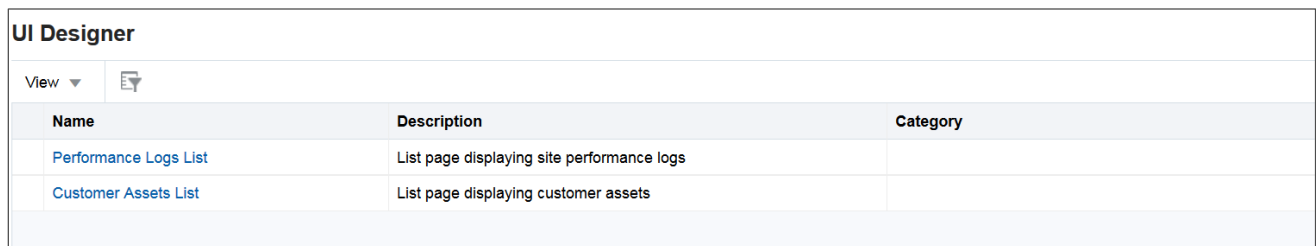
UI Designer is a new drag and drop editor that provides administrators with a simple and intuitive way to edit the new Oracle Application Development Framework (ADF) pages within CPQ Cloud. Oracle ADF is an end-to-end framework that simplifies application development by providing out-of-the-box infrastructure services and a faster and simpler development experience.

Introduced in CPQ Cloud 2016 R1, UI Designer provides a way to administer pre-defined layouts using the following functionality:

- [Layouts List Page](#)
- [UI Designer Screen Layout](#)
- [Layout](#)
- [Attributes](#)
- [Layout Settings](#)
- [Panel Settings](#)
- [Table Settings](#)
- [Column Settings](#)
- [Button Settings](#)

LAYOUTS LIST PAGE

The **Layouts List** page contains the two pre-generated ADF user interface layouts available in CPQ Cloud 2016 R1 (**Customer Assets List** and **Performance Logs List**). Administrators can customize these layouts to meet the needs of their organization.




UI Designer		
View ▾		
Name	Description	Category
Performance Logs List	List page displaying site performance logs	
Customer Assets List	List page displaying customer assets	

Figure 32: Layouts List Page

Administrators can access the **Layouts List** page from the Admin Home Page by selecting **UI Designer** in the **General Settings** section. Use the **Layouts List** page to click the name of the pre-generated layout you want to customize. This opens the layout in **UI Designer**.

UI DESIGNER SCREEN LAYOUT

UI Designer contains a drag and drop interface that allows administrators to easily customize the two pre-generated ADF layouts available in CPQ Cloud 2016 R1. These layouts are shown below in Figure 33: Customer Assets List and Figure 34: Performance Logs List.

By clicking content in the layout such as the column or panel heading outlined in red in Figure 33: Customer Assets List, administrators can view and modify the associated settings. Tooltips with useful hints about the settings are viewable by hovering over the settings with your mouse.

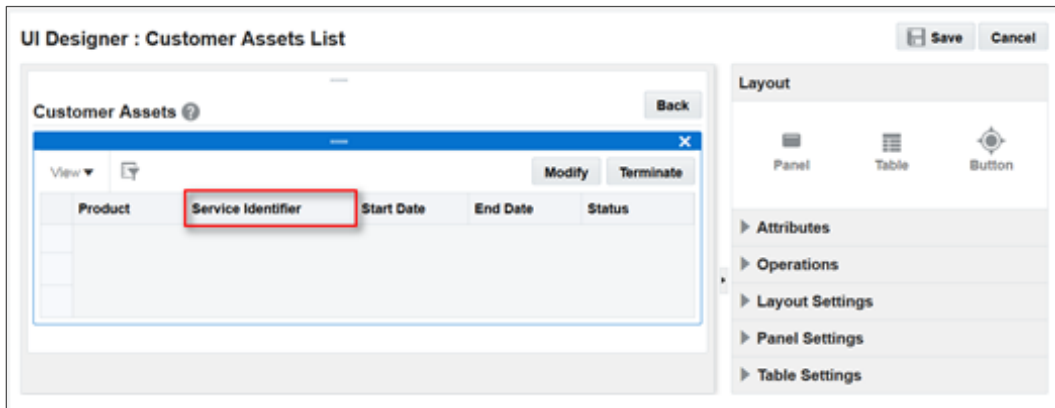


Figure 33: Customer Assets List

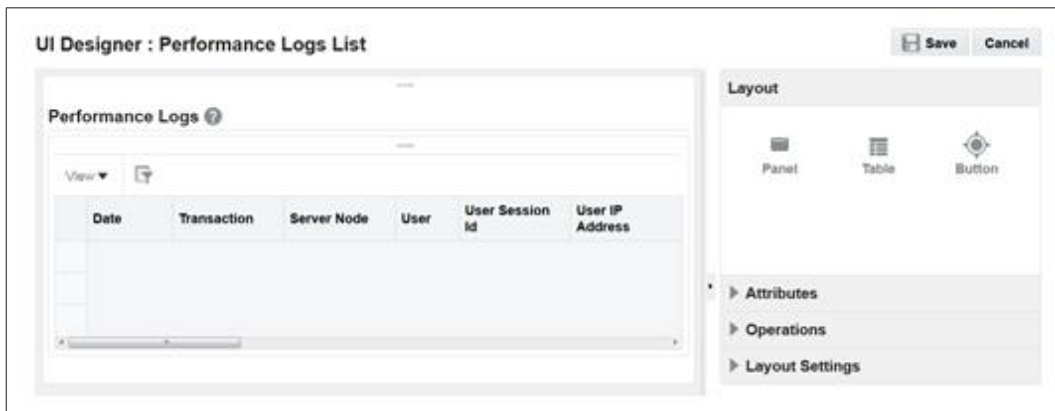


Figure 34: Performance Logs List

LAYOUT

The **Layout** panel contains basic components used to create the layout. Drag and drop the desired components onto the content area to place the component in the layout.

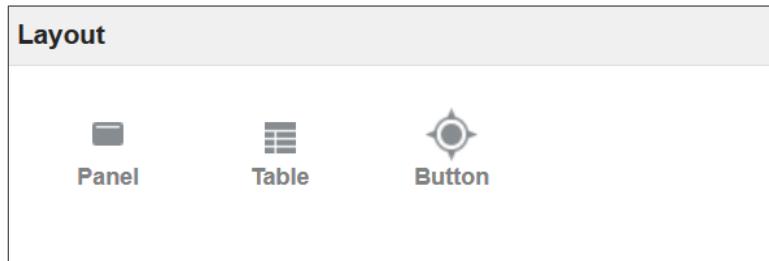


Figure 35: Layout Panel

NOTE: In CPQ Cloud 2016 R1, only one panel and one table is supported per layout.

ATTRIBUTES

The **Attributes** panel contains a list of object attributes that are available to add into the selection in the content area. Drag and drop an attribute onto the table to add it as a column.

NOTE: The attributes in the **Attributes** panel are only available when the table is selected. If the table is not selected, no attributes display in the panel.

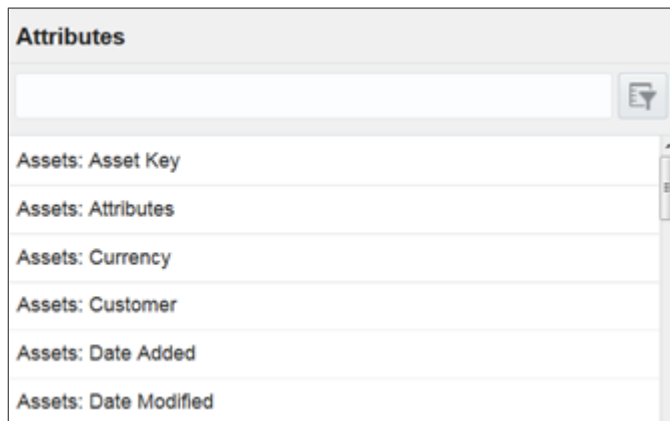


Figure 36: Attributes Panel

To remove a column, select the column, and click the remove icon located to the immediate right of the icon. In Figure 37: Remove a Column, the remove icon is outlined in red.

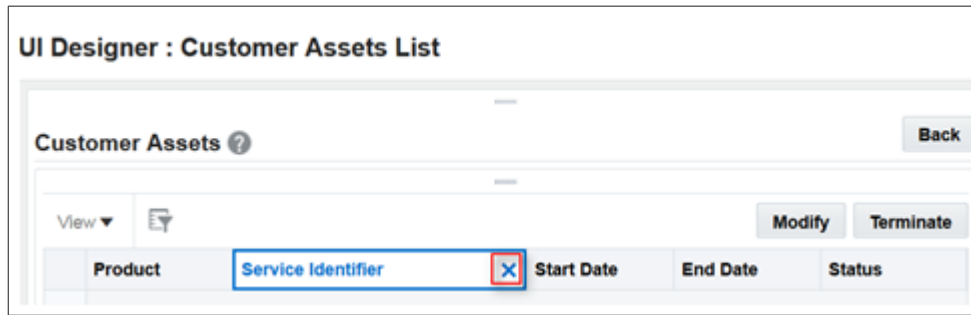


Figure 37: Remove a Column

LAYOUT SETTINGS

Use **Layout Settings** to customize the **Name**, **Description**, **Parameters**, and **Category** associated with a layout. Instructions for each of the **Layout Settings** are provided below and are available as Tooltips by hovering over the settings with your mouse.

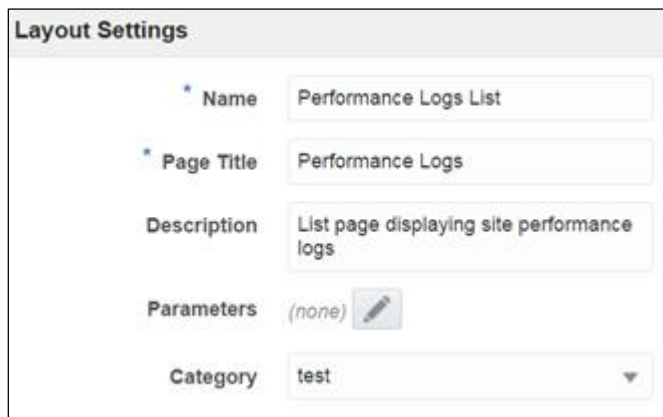


Figure 38: Layout Settings

NOTE: The **Name**, **Description**, and **Category** values are viewable from the **Layouts List** page.

- Name: Enter a unique identifier for the layout.
- Title: Enter a title to display in the browser tab or header.
- Description: Enter a description of the content or purpose of the layout.

- Parameters: Use the Layout Parameters dialog to add, modify, or delete the parameters associated with a layout. The Layout Parameters dialog can contain default values that are already defined, but they are primarily defined through parameters that are passed through a URL.

Parameter	Data Type	Default	Required	Internal
<input type="checkbox"/> backUrl	Text		<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> commerceProcess	Text		<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> customer	Text		<input checked="" type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> result	Json		<input type="checkbox"/>	<input checked="" type="checkbox"/>
<input type="checkbox"/> transactionId	Integer		<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> bnDate	Date Time		<input checked="" type="checkbox"/>	<input type="checkbox"/>

Figure 38: Layout Parameters

- Category:** Select a category or create a new category for organizing the layout on the **Layouts List** page.

Layout Settings

Name: Performance Logs List

Page Title: Performance Logs

Description: List page displaying site performance logs

Parameters: (none)

Category: test

Figure 39: Layout Settings

PANEL SETTINGS

Use **Panel Settings** to customize the **Title**, **Help Text**, **Help URL**, **Tooltip**, or **Icon Set** associated with a panel. A sample panel is outlined in red and shown below.



Figure 40: Sample Panel

Descriptions of each of the **Panel Settings** are provided below.

- **Title:** The title displayed in the upper-left of the panel. In Figure 40: Sample Panel, Customer Assets is the title.
- **Help Text:** The instructional text that displays upon hovering over the Help icon with your mouse.
- **Help URL:** The URL of the page that opens upon clicking the Help icon.

NOTE: If no content is entered in the **Help Text** or **Help URL** fields, a Help icon does not display on the panel.

- **Tooltip:** The short message that displays upon hovering over any part of the panel.
- **Icon Set:** The icon that displays in the upper-left of the panel next to the title. An icon set on a panel only displays in the Enabled state. The behavior of an icon set on a panel is static in comparison to how an icon set functions on a button.

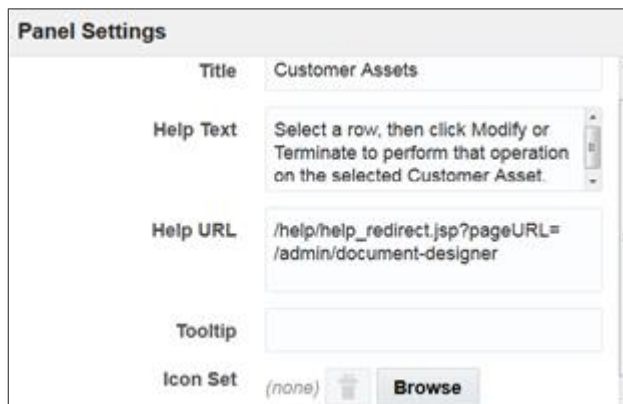


Figure 41: Panel Settings

TABLE SETTINGS

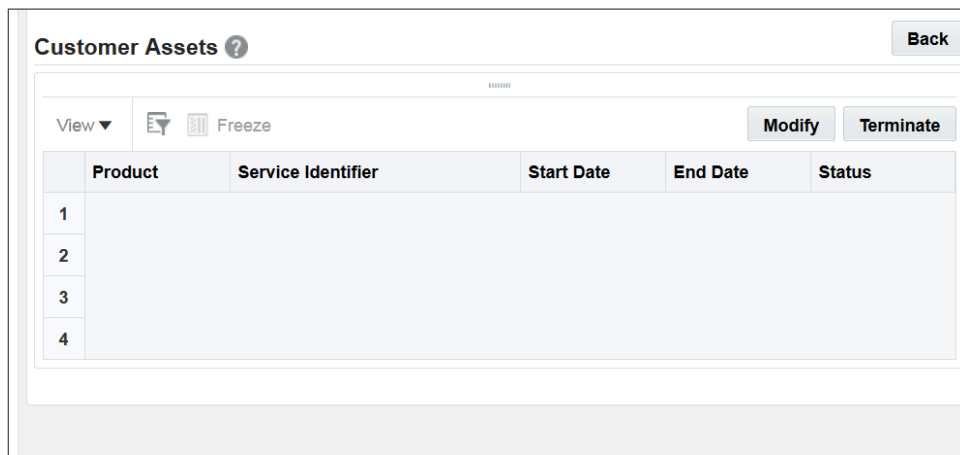
The **Customer Assets List** layout and the **Performance Logs List** layout are both displayed in a table format. Use **Table Settings** to make customizations to these tables, including enabling the freeze column feature on the table, adding row numbering, and changing the default column width.

Descriptions of each of the Table Settings are provided below.

- **Summary:** Enter a summary of the contents of the table. This is an important step for creating accessible layouts.
- **Resource:** Select the object to use for defining the contents of the table. The available options include Assets and Performance Logs.

NOTE: For additional information about assets, refer to the Subscription Ordering section of this document. For additional information about performance logs, refer to the CPQ Cloud Online Help.

- **Resource Query Filter:** Define a filter to apply to the selected object.
- **Tooltip:** Enter a short message to display upon hovering over the table.
- **Expanding Column:** Select a column to fill extra space in the table.
- **Height (In Rows):** Specify the number of rows to display in the table for the height. As the user scrolls down, additional rows will display.
- **Allow Freeze Columns:** Check the **Allow Freeze Columns** checkbox to display a **Freeze** button in the table's toolbar, so users can freeze or un-freeze columns. Selecting and freezing a column creates a horizontal scroll bar to the right of the frozen column.
- **Row Numbering:** Check the **Row Numbering** checkbox to display row numbering for each row in the table. As the user scrolls, additional rows become available.



The screenshot shows a web interface for 'Customer Assets'. At the top right is a 'Back' button. Below the title is a toolbar with a 'View' dropdown, a 'Freeze' button (with a vertical line icon), and 'Modify' and 'Terminate' buttons. The table has five columns: 'Product', 'Service Identifier', 'Start Date', 'End Date', and 'Status'. The first column is numbered 1 through 4. A vertical line is visible between 'Product' and 'Service Identifier', indicating that the 'Service Identifier' column is frozen.

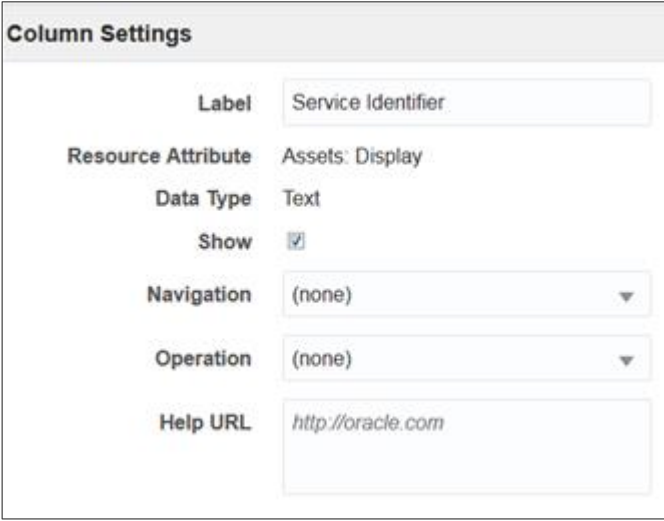
	Product	Service Identifier	Start Date	End Date	Status
1					
2					
3					
4					

Figure 42: Table with Row Numbering and Column Freeze Enabled

COLUMN SETTINGS

Use **Column Settings** to modify a column label, show or hide a column, and add or modify the label, URL, and any specified parameters associated with a column. Descriptions of each of the **Column Settings** are provided below.

- **Label:** The column label to display in the column header.
- **Resource Attribute:** Displays the object attribute associated with the column (e.g. the attribute that was dragged from the **Attributes** panel to a table to create the column).
- **Data Type:** Displays the data type associated with the object attribute.
- **Show:** Select the checkbox to display the table column to end users by default. End users then have the option to hide or display the column.
- **Navigation:** Select a URL to associate with the column.
- **Operation:** Select the operation to assign to the column.
- **Help URL:** Enter the text or the URL to display upon clicking the Help icon.



The image shows a dialog box titled "Column Settings" with the following fields:

Label	Service Identifier
Resource Attribute	Assets: Display
Data Type	Text
Show	<input checked="" type="checkbox"/>
Navigation	(none) ▼
Operation	(none) ▼
Help URL	http://oracle.com

Figure 43: Column Settings

BUTTON SETTINGS

Use **Button Settings** to specify a **Label**, **Navigation** option, **Operation**, **Tooltip**, and **Icon Set** for a button. Descriptions of each of the **Button Settings** are provided below.

- **Label:** Enter a label to display on the button.
- **Navigation:** Select a URL to associate with a button. When the button is clicked, the specified URL opens.
- **Operation:** Select an operation to assign to the button.
- **Tooltip:** Enter a short message to display upon hovering over the button.
- **Icon Set:** Associate a button with an icon set. Each icon set has an Enabled, Disabled, and a Hover state as well as an Active state for when the button associated with the icon set is clicked.
- **Hide Label:** Hides the text label assigned to a button and renders the button as icon-only.

NOTE: When an icon set is assigned to a button, the “Hide Label” option is available on the **Button Settings** panel.

Button Settings


* Label	<input type="text" value="Modify"/>
Navigation	<input type="text" value="URL"/>
URL	<pre>"/commerce/new_equipment/products /model_configs.jsp?_from_punchin=true& product_line=" result.product_line &model=" result.model "&segment=" result.segment &bm_sales_root_bom_item_id=" result.bomkey "&configContextKey=" result.configContextKey</pre>
Open in	<input type="text" value="Current Window"/>
Operation	<input type="text" value="Get Configuration Instance"/>
Parameters	<pre>assetKey, commerceProcess, result, transactionDate, transactionId</pre>
Tooltip	<input type="text" value="Modify selected asset"/>
Icon Set	(none)  <input type="button" value="Browse"/>

Figure 44: Button Settings

TIPS AND CONSIDERATIONS

Consider the following tips when using UI Designer:

- Only one panel per layout is currently supported. When an administrator attempts to drag and drop an additional panel onto a layout, the layout is not updated with the additional panel.
- When defining a Help URL or Navigation URL, use `http://` or `https://` at the beginning of external links. Otherwise, the path is assumed internal within the CPQ Cloud page. If only an end-point such as `/commerce/display_company_profile.jsp` is entered, the CPQ domain is automatically added to make the entire URL `[site].oracle.com/commerce/display_company_profile.jsp`.
- As a recommended good practice, make the layout parameters used in a **Resource Query Filter** mandatory, especially if the parameter type is not text. Otherwise, the **Resource Query Filter** evaluation may fail, resulting in a runtime error when attempting to load the page.
- Icon sizes on buttons are constrained to 16 x 16 pixels. While it may appear that the icon size is constrained on panels as well, the images will display to the end user in the icon's unrestricted size. When defining an image for an icon set, make sure the image size is correct in File Manager.
- Not all of the attribute types available on the **Customer Assets List** layout or the **Performance Logs List** layout are supported. For example: An administrator can add Currency and Currency Code attributes to the **Customer Assets List** layout, but the attributes do not display values for the end user.

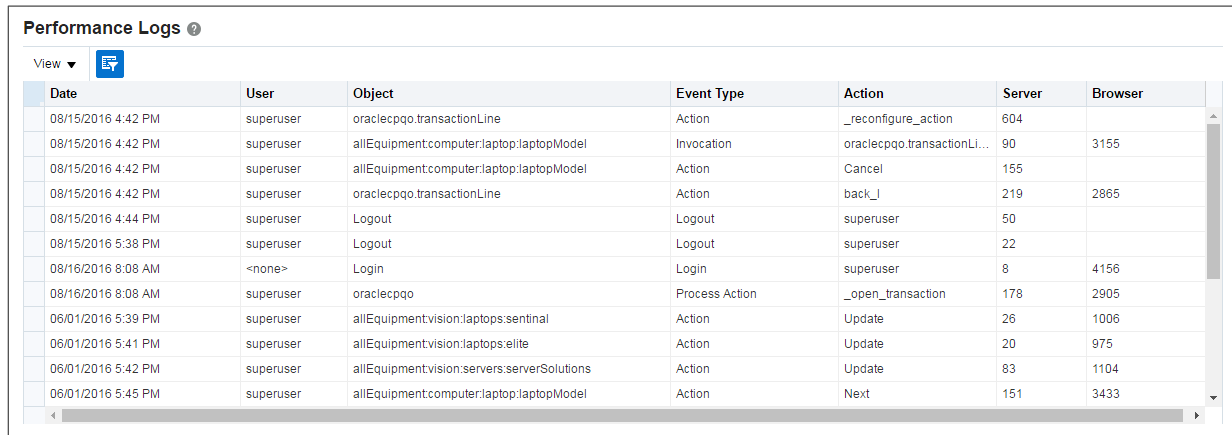
KEY RESOURCES

CPQ Cloud Online Help: Refer to the UI Designer topics.

PERFORMANCE LOGS PAGE

The new **Performance Logs** page, available in CPQ Cloud Release 2016 R1, allows administrators to monitor and analyze the performance experienced by CPQ Cloud sales users. The page leverages the Performance Logs REST API delivered by CPQ Cloud in Release 2015 R1, and the new UI Designer, to deliver improved access and usability to performance log data.

Access the **Performance Logs** page from the Admin Home Page by selecting **Performance Logs** in the **Developer Tools** section.



The screenshot shows the Performance Logs page with a table of user actions. The table has columns for Date, User, Object, Event Type, Action, Server, and Browser. The data is as follows:

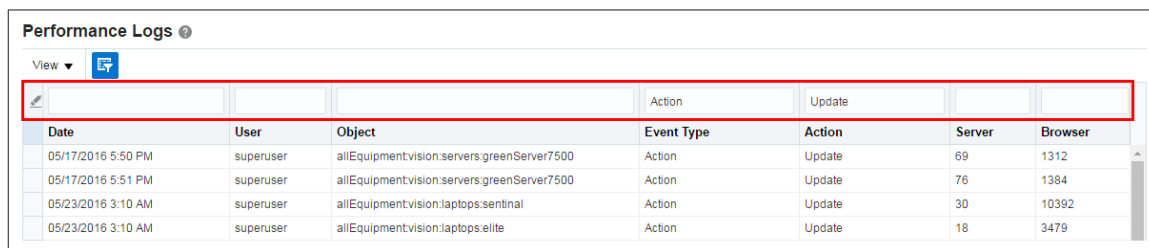
Date	User	Object	Event Type	Action	Server	Browser
08/15/2016 4:42 PM	superuser	oraclecpqo.transactionLine	Action	_reconfigure_action	604	
08/15/2016 4:42 PM	superuser	allEquipment:computer:laptop:laptopModel	Invocation	oraclecpqo.transactionLi...	90	3155
08/15/2016 4:42 PM	superuser	allEquipment:computer:laptop:laptopModel	Action	Cancel	155	
08/15/2016 4:42 PM	superuser	oraclecpqo.transactionLine	Action	back_I	219	2865
08/15/2016 4:44 PM	superuser	Logout	Logout	superuser	50	
08/15/2016 5:38 PM	superuser	Logout	Logout	superuser	22	
08/16/2016 8:08 AM	<none>	Login	Login	superuser	8	4156
08/16/2016 8:08 AM	superuser	oraclecpqo	Process Action	_open_transaction	178	2905
06/01/2016 5:39 PM	superuser	allEquipment:vision:laptops:sentinal	Action	Update	26	1006
06/01/2016 5:41 PM	superuser	allEquipment:vision:laptops:elite	Action	Update	20	975
06/01/2016 5:42 PM	superuser	allEquipment:vision:servers:serverSolutions	Action	Update	83	1104
06/01/2016 5:45 PM	superuser	allEquipment:computer:laptop:laptopModel	Action	Next	151	3433

Figure 45: Performance Logs

The page allows administrators to view user actions such as Logins, Logouts, Commerce and Configuration actions, with the elapsed server and browser times required to complete the actions. Click a column header to select it and enable ascending and descending sorting of the column's content. Hide or reveal the columns displayed in the table using the **View** menu. Users can resize and rearrange columns to customize their view.

QUERY BY EXAMPLE (QBE)

Filter logged events using the new Query by Example (QBE) feature, which is available on UI Designer generated pages. Click the filter icon to enable the query fields, and then enter filter criteria. Press enter to activate the filter. Filter results across multiple columns in this manner. Click the pencil eraser icon, on the left side of the QBE row, to clear the filter.



The screenshot shows the Performance Logs page with a Query by Example (QBE) filter applied. The filter is highlighted with a red box and contains the text "Action Update". The table below shows the filtered results:

Date	User	Object	Event Type	Action	Server	Browser
05/17/2016 5:50 PM	superuser	allEquipment:vision:servers:greenServer7500	Action	Update	69	1312
05/17/2016 5:51 PM	superuser	allEquipment:vision:servers:greenServer7500	Action	Update	76	1384
05/23/2016 3:10 AM	superuser	allEquipment:vision:laptops:sentinal	Action	Update	30	10392
05/23/2016 3:10 AM	superuser	allEquipment:vision:laptops:elite	Action	Update	18	3479

Figure 46: Query by Example (QBE)

KEY RESOURCES

CPQ Cloud Online Help: Refer to the Performance Logs and UI Designer topics.

BML ENHANCEMENTS

CPQ Cloud's markup language, BML, now allows administrators to create new Java Script Object Notation (JSON) data types (e.g. JSON, JSON array, and JSON null) and generate, modify, parse, extract, and query JSON data using BML JSON and JSON array manipulation functions. The addition of JSON Path expression in JSON manipulation functions allows advanced users to easily retrieve or modify a node or value in JSON data. Available in CPQ Cloud 2016 R1, these BML enhancements support Bill of Material (BOM) and Subscription Ordering.

Additional information about the 2016 R1 BML enhancements is provided below:

- [JSON Related Functions](#)
- [JSON Array Related Functions](#)
- [JSON Path Related Functions](#)
- [Functions to Support Remote Approvals](#)
- [Generate Unique IDs Function](#)
- [URL Access Function](#)
- [Same Server Authentication](#)
- [User Session Functions](#)
- [Global Dictionary Functions](#)
- [Throw Error Function](#)
- [Apply Template Function](#)
- [BML Print Log](#)

JSON RELATED FUNCTIONS

The following are JSON-related functions introduced in CPQ Cloud 2016 R1.

JSON Function	Description
<code>json()</code> :	Creates a JSON object.
<code>jsonkeys()</code> :	Retrieves all first-level keys from a JSON object and returns an array of strings. If the optional parameter (<code>ignoreNullValues</code>) is set to true, null value keys are ignored.
<code>jsontostr()</code> :	Converts a JSON object into a JSON formatted string.
<code>jsonget()</code> :	Gets the value from the JSON object for the key provided. If the key is not available, the value provided in the default Value parameter is returned.
<code>jsonput()</code> :	Puts an entry (key-value) into the JSON object.
<code>jsonremove()</code> :	Removes the first-level key-value pair from the JSON object.
<code>jsoncopy()</code> :	Returns a copy of the given JSON object.
<code>jsonnull()</code> :	Gets an instance of the JSON null object (represents null in the JSON string).
<code>isjsonnull()</code> :	Checks the null value in the JSON or JSON array object and returns true if the value is null.

JSON ARRAY RELATED FUNCTIONS

The following JSON Array-related functions are available in CPQ Cloud 2016 R1.

JSON Array Function	Description
<code>jsonarray()</code> :	Creates a JSON array object.
<code>jsonarraytostr()</code> :	Converts a JSON array object into a JSON array formatted string.
<code>jsonarrayget()</code> :	Gets the value from the JSON array object for the index provided.
<code>jsonarraysize()</code> :	Returns the size of the JSON array.
<code>jsonarrayappend()</code> :	Appends the given value at the end of the JSON array.
<code>jsonarrayremove()</code> :	Removes an object specified at given index from the JSON array.
<code>jsonarraycopy()</code> :	Returns a copy of the of the given JSON array object.

JSON PATH RELATED FUNCTIONS

The following are JSON-path related functions available in CPQ Cloud 2016 R1.

JSON Path Function	Description
<code>jsonpathgetsingle()</code>	Retrieves the value from the JSON object for the given JSON path expression.
<code>jsonpathgetmultiple()</code>	Retrieves the value(s) and path(s) from the JSON object for the given JSON path expression.
<code>jsonpathcheck()</code>	A JSON lookup function that checks if the JSON path is found in the JSON.
<code>jsonpathset()</code>	Updates all of the nodes corresponding to a given JSON path expression with a given value.
<code>jsonpathremove()</code>	Removes the object(s) and value(s) corresponding to a given JSON path expression.

JSON PATH EXPRESSION

A JSON Path expression is used to represent one or more nodes or values in a JSON structure and is also used for data filtering and further data insight.

There are two ways to write a simple JSON path expression.

SIMPLE DOT NOTATIONS:

- Generally used when keys are alphanumeric (e.g. contain only numbers and alphabets).
- Each dot notation signifies one level scan. For example: `$.id` represents the value corresponding to the first level “id” key.
- JSON paths have a similar use for an n-level deep scan. Example: `$.attributes.size` path starts by finding the first level “attributes” key and then goes to the second level “size” key.
- When double dots(..`)` are used, this represents a deep scan in the JSON structure. For example: `$.value` represents all of the values corresponding to all the “value” keys at any level in the structure.
- Combine single and double dot expressions to create JSON path expressions. For example: `$.size.value` first searches for the entire document and gets all the values corresponding to “size” keys at any level. Within these values, the values corresponding to the next level “value” key are then retrieved.
- Whenever an array is encountered while writing a JSON path expression, specify the index of that array. For example: `$.children[0].variableName` gets the variable name from the first child.
- When writing simple JSON path expression, combine dot, double-dots, object, and array notations to intuitively represent one or more values.
- One JSON path can also represent multiple values. For example: `$.label` will represent two different values.

BRACKET NOTATIONS:

- Bracket notations are generally used when keys are not alphanumeric, but the overall intuition is exactly the same as dot notations.
- For example: `$.['attributes'].['size'].['label'],$.['children'][0].['quantity']` Use the following operators to write complex JSON path expressions.

DOT OPERATORS

• Operators	Description
\$	The root element that starts all path expressions.
@	The current node within the processed filter.
*	Name or numeric wildcard character.
..	Used to search all child nodes. Generates an array of results.
.<name>	Single child relation representation.
['<name>' (, '<name>')]	Single or many child relation representation.
[<number> (, <number>)]	Single or many array index representation.
[start:end]	Array range representation.
[?(<expression>)]	Boolean filter expression.

Certain functions may be used at the end of a JSON path to perform specific functions using the dot notation. A few examples are located in the

JSON Path Examples section below. The min(), max(), avg(), and stddev() functions can be used to evaluate an array of numbers. The length() function can be used to provide the size of an array result.

Boolean filter expressions help reduce a JSON array to a subset of intended values. A typical filter is \$.children[?(@.quantity > 1)] where @ represents the current node being processed. Create more complex filters with logical operators && and ||. Ensure string literals are enclosed by single quotes: \$.children[?(@.partNumber = 'PT13345')]

Filter expressions support common comparison operators ==, !=, <, <=, >, >=, and the below advanced operators.

ADVANCED OPERATORS

Operators	Description
=~	Left matches regular expression [?(@.name =~ /foo.*?/i)].
in	Left side is contained in right. [?(@.color in ['Red', 'Green'])].
nin	Left side is not contained in right.
size	Left side size is equal to right.
empty	Left side is empty.

JSON PATH EXAMPLES

Given the following:

```
{
  "id": "7345ABCDE",
  "variableName": "BM54888-0",
  "partNumber": "BM54888",
  "quantity": 1,
  "definition": {
    "SequenceNum": 20,
    "ItemId": "EBS56321"
  },
  "fields": {
    "_price_list_price_each": 99.12,
    "line_action_code": "Update"
  },
  "attributes": {
    "size": {
      "value": "Large",
      "label": "Size"
    },
    "instruction": {
      "value": "Leave the package at the door.",
      "label": "Special Instruction"
    }
  },
  "children": [
    {
      "variableName": "BM54888-1",
      "partNumber": "PT13345",
      "quantity": 1
    },
    {
      "variableName": "BMDSK781-4",
      "partNumber": "DSK781-4",
      "quantity": 17
    }
  ],
  "numericVals": [1, 2, 4, 5.6, 89, 0.05],
  "Special key $$": true
}
```


JSON Path	Description / Output
\$.id	Get the id. "7345ABCDE"
\$.value	Get all the values available at any level. "Large" "Leave the package at the door."
\$.attributes.size.value	Get the value of the attributes -> size. "Large"
\$.size.value	Get the value of the size key available at any level. "Large"
\$.children[0]	Get the first child. { "variableName": "BM54888-1", "partNumber": "PT13345", "quantity": 1 }
\$.children[0].quantity	Get the quantity of the first child. 1
\$.['id']	Get the id. "7345ABCDE"
\$.['value']	Get all the values available at any level. "Large" "Leave the package at the door."
\$.['attributes']..['label']	Get all the labels under attributes. "Size" "Special Instruction"
\$.[' Special key \$\$ ']	Get the value corresponding to given key. true
\$.children[0,1]	Get first and second children. { "variableName": "BM54888-1", "partNumber": "PT13345", "quantity": 1 } { "variableName": "BMDSK781-4", "partNumber": "DSK781-4", "quantity": 17 }
\$.children[?(@.quantity > 1)]	Get the children whose quantity is greater than 1. { "variableName": "BMDSK781-4", "partNumber": "DSK781-4", "quantity": 17 }
\$.children[?(@.partNumber == 'PT13345')]	Get the children whose partNumber is 'PT13345'. { "variableName": "BM54888-1", "partNumber": "PT13345", "quantity": 1 }
\$.numericVals.avg()	Get the avg of numeric values. 16.941666666666666

JSON Path	Description / Output
\$.numericVals.length()	Get the length of the array corresponding to numericVals key.
	6
\$.children[?(@.partNumber nin ['PT13345'])].quantity	Get the quantity of the child whose partNumber is not 'PT13345'.
	17
\$..children[0].*	Get all the children values of first children.
	BM54888-1" "PT13345" 1
\$.attributes.*	Get all the direct children of attributes node.
	{"value": "Large", "label": "Size"} {"value": "Leave the package at the door.", "label": "Special Instruction"}
\$.attributes..*	Get all the deep level children of attributes node.
	{"value": "Large", "label": "Size"} {"value": "Leave the package at the door.", "label": "Special Instruction"} "Large" "Size" "Leave the package at the door." "Special Instruction"
\$..children[-1:].quantity	Get the last child's quantity.
	17
\$..children[*].quantity	Get the quantities for all the children.
	1 17

FUNCTIONS TO SUPPORT REMOTE APPROVALS

The following functions are available to support remote approvals. CPQ Cloud integration with Oracle Process Cloud Service (PCS) leverages these functions. The functions are also useable outside of PCS.

- dict<anytype>
- bytearray
- getattachmentdata
- urlmultipartbypost

NOTE: Use the remote approval functions to retrieve the content of a file attachment as a byte array data type and store the attachment details in a new data type called dict<anytype>.

DICT<ANYTYPE>

dict<anytype>		
Description	This function supports the addition of multiple types of objects in a dictionary. The dictionary get(), put(), and keys() functions also now support the addition or retrieval of values or keys from a dict<anytype>.	
Parameter	<anytype>	String <anytype> represents one or more combinations of the following data types: string, integer, float, date, string[], integer[], float[], date[], string[][], integer[][] , float[][] , date[][] , boolean, dict<string>, dict<anytype>, dict(dict<anytype>), JSON, JSON array, or byte array.
Syntax	dict (String <anytype>)	
Sample Input	<pre>d1 = dict("anytype"); put(d1, "key1", "value1"); jObj = json("{\"K1\": \"V1\"}"); put(d1, "key2", jObj);</pre>	
Sample Return	A dictionary to contain key-value entries of various data types is created and two key-value entries are inserted into the dictionary.	

BYTEARRAY()

bytearray()		
Description	This function stores a collection of binary data such as the contents of a file. It encodes the given string into a sequence of bytes using the specified character encoding. This new type was added to support PCS integration and is used in the BML getattachmentdata function to return the content of a file as a byte array.	
Parameter	content	String The string to be encoded.
	charSet	String The character encoding. This can be any encoding supported by Java SE Runtime Environment 6, such as ASCII, ISO-8859-1, UTF-32BE, etc. Optional, the default is UTF-8 if not provided.
Syntax	ByteArray bytearray(String content [, String charset])	
Sample Input	content	Sample String
	charset	UTF-16
Sample Return	bytearray [UTF-16]: Sample String	

GETATTACHMENTDATA()

getattachmentdata()			
Description	This function returns the file name (filename), file content (filecontent), and MIME type (mimetype) of a given file attachment attribute in a dictionary of <anytype>. The file content can be retrieved as a bytearray or as Base64 encoded string based on the input parameters.		
Parameter	attachmentId	String	The ID of the attachment returned. This ID is returned in the reference to the attachment attribute in BML.
	asBytes	Boolean	If true, the returned datatype will be a bytearray; If false, the returned datatype will be a Base64 encoded String.
			Optional, the default is false if not provided.
Syntax	Dict<anytype> getattachmentdata(String attachmentId [, Boolean asBytes])		
Sample Input	attachmentID	12345678	
	asBytes	false	
Sample Return	<pre>{ mimetype=text/plain, filename=example.txt, filecontent=VGhpcyBpcyBhbiBleGFtcGxlLg== }</pre>		

URLMULTIPARTBYPOST()

urlmultipartbypost()			
Description	This function supports remote approvals by sending multi-part messages with attachments.		
Parameters	url	String	The request url.
	payload	String	The payload.
	headers	Dictionary	The message headers.
			Optional.
attachments	Dictionary		
Syntax	Dictionary urlmultipartbypost(String url, String payload, [Dictionary headers, [Dictionary attachments]])		
Sample Input	url	requesturl	
	payload	<pre>{ "processDefId": "default~CPQApproval!1.0~QuoteApprovalProcess", "serviceName": "QuoteApprovalProcess.service", "operation": "start", "payload": "<quot: start xmlns:quot='http://xmlns.oracle.com/bpmn/bpmnCloudProcess/CPQApprovalDemo/QuoteApprovalProcess'><requestor>Suyog</requestor><quoteId123456</quoteId><quoteTotal>12567</quoteTotal><discountPercentage>23</discountPercentage><requestorEmail>approve_pCSsubmit</requestorEmail></quot:start>", "action": "Submit" }</pre>	
	headers	message header	
	attachments	dictionary of multiple attachments	
Sample Return	On success, a Dictionary representation of the http response, such as “Status-Code”, “Content-Type”, “Message-Body”, etc., is returned.		

GENERATE UNIQUE IDS FUNCTION

getuuid()		
Description	This function generates unique IDs for assets tracked in Subscription Ordering. Every asset is tracked in Subscription Ordering using an asset key. When a unique ID is generated, it becomes the asset key for an asset. The number of asset keys to generate is dependent on the count parameter in the function.	
Parameters	count	Integer The number of unique IDs to generate.
Syntax	String[] getuuid(Integer count)	
Sample Input	count	2
Sample Return	[6bafc278-25fd-495f-8360-67bcfb8776b0, 65abced9-5c47-47c6-bf18-ab96fb73935f]	

URL ACCESS FUNCTION

The `urldata()` function has been enhanced to support the http "DELETE" and "PUT" methods of RESTful web services.

Examples:

Invoke RESTful service using http DELETE method:

```
headerDict = dict("string");
encodecredential = encodebase64("<username>:<password>");
authStr = "Basic " + encodecredential;
put(headerDict, "Authorization", authStr);
response =
urldata("http://<hostname>/rest/v1/commerceDocumentsTransaction_bmClone_2Quote
/{bsid}/lineItem/{id}", "DELETE", headerDict);
```

Invoke RESTful service using http PUT method:

```
headerDict = dict("string");
encodecredential = encodebase64("<username>:<password>");
authStr = "Basic " + encodecredential;
put(headerDict, "Authorization", authStr);
put(headerDict, "Content-Type", "application/json");
cancelBody = "{\"id\":\"cancel\"}";
response = urldata("http://<hostname>/bpm/api/1.0/processes/101", "PUT",
headerDict, cancelBody);
```

SAME SERVER AUTHENTICATION

The `urldata()` and `urldatapost()` functions have been enhanced to allow optional authorization when invoking an internal RESTful web services call. If authorization credentials are not provided, the current user's login credentials are used.

Example:

Invoke RESTful service using `urldatapost()`:

```
url =
"http://<hostname>/rest/v1/commerceDocumentsTransaction_bmClone_2Quote/{id}
/actions/saveWithAdvancedValidation";
jsondesc = json();
jsonput(jsondesc, "quoteDescription", "new Quote");
payload = json();
jsonput(payload, "documents", jsondesc);
headerDict = dict("string");
//Authorization string is no longer required in the header
put(headerDict, "Content-Type", "application/json");
urldatapost(url, jsontostr(payload), "" , headerDict, true);
```

USER SESSION FUNCTIONS

User session functions support setting, removing, and getting a key-value pair from the session cache in BML to support Subscription Ordering. The values are available as long as the user session is active. Values stored in the session cache are removed automatically when the user logs out, the session expires, or the server is restarted.

The following are new user session functions:

- `usersessionset()`:
- `usersessionget()`:
- `usersessionremove()`:

USERSESSIONSET()

usersessionset()			
Description	This function sets a key-value pair to the user session. The set key-value pair is lost once the session expires.		
Parameter	key	String	The key corresponding to the stored value.
	value	String or JSON	The value to be stored. The value can be of type String or Json.
Syntax	usersessionset (String key, <ValueType> value)		
Sample Input	key	sessionKey1	
	value	value1	
Sample Return	A key-value pair is set in the user session.		

USERSESSIONGET()

usersessionget()			
Description	This function retrieves a value for a given key from a user session. If the key is not found, null is returned.		
Parameter	key	String	The key corresponding to the value to be retrieved.
	valueType	String or JSON	The expected data type of the returned value.
			Optional. The default value is String.
Syntax	<ValueType> usersessionget (String key [, String valueType])		
Sample Input	key	sessionKey1	
	valueType	String	
Sample Return	value1		

USERSESSIONREMOVE()

usersessionremove()			
Description	This function removes a key-value pair from the user session. The function returns true if the key-value pair is successfully removed, and returns false if the key does not exist in the user session.		
Parameters	Key	String	The key corresponding to the key-value pair to be removed.
Syntax	Boolean usersessionremove(String key)		
Sample Input	sessionKey1		
Sample Output	true		

GLOBAL DICTIONARY FUNCTIONS

Global dictionary functions support Subscription Ordering and are available for setting, getting, and removing a key-value pair from the global dictionary in BML. Available across multiple sessions, the values are removed periodically when they exceed the minimum time to live specified while setting the value. There is no guarantee the global dictionary values are available after the minimum time to live.

The following are the new global dictionary functions:

- `globaldictset()`:
- `globaldictget()`:
- `globaldictremove()`:

GLOBALDICTSET()

globaldictset()			
Description	This function adds or updates the key-value pair in the global dictionary.		
Parameters	key	String	If key is null or empty, a unique key is generated and added to the global dictionary. If the key provided is not present in the global dictionary, a key-value pair is added to the global dictionary.
	value	String	If a key is present, the value is updated for the corresponding key.
	minTimeToLive	Integer	The minimum time, in minutes, the key-value pair is guaranteed in the global dictionary. The value should be greater than 0 and less than 525600 minutes (365 days).
			Optional. The default value is 1440 minutes.
Syntax	String globaldictset(String key, String value [, Integer minTimeToLive])		
Sample Input	key	globalKey1	
	value	value1	
	minTimeToLive		
Sample Output	globalKey1		

GLOBALDICTGET()

globaldictget()			
Description	This function returns a value stored in the global dictionary corresponding to the given key. If the key is not found in the global dictionary, null is returned.		
Parameters	key	String	The key corresponding to the value to be retrieved.
	updateTimeToLive	Boolean	If set to true, the minimum time to live is recalculated from the retrieved time. If set to false, there is no change to minimum time to live.
			Optional. The default value is false.
Syntax	String globaldictget(String key [, Boolean updateTimeToLive])		
Sample Input	key	globalKey1	
	updateTimeToLive		
Sample Output	value1		

GLOBALDICTREMOVE()

globaldictremove()			
Description	This function removes a given key-value pair from the global dictionary. The function returns true if the key-value pair is successfully removed, and returns false if the key does not exist in the global dictionary.		
Parameters	key	String	The key corresponding to the key-value pair to be removed.
Syntax	Boolean globaldictremove(String key)		
Sample Input	globalKey1		
Sample Output	true		

THROW ERROR

throwerror()			
Description	<p>This function stops the execution of a BML script and displays either a business logic error or a system error to the end user. In the case of a system error, a generic error message displays and the actual error message is logged to the error log file.</p> <p>Note: Oracle does not recommend using this function in Configuration or Commerce rule conditions. When the rules are triggered from quote transaction screens, the error does not always display to the user.</p>		
Parameters	<code>errorMessage</code>	String	The error to display to the end user.
	<code>isSystemError</code>	Boolean	The flag to select an error type. If false, <code>errorMessage</code> is displayed to the user. If true, a system error is thrown and <code>errorMessage</code> is printed in the error log file.
			Optional. The default value is false.
Syntax	<code>throwerror(String errorMessage [, Boolean isSystemError])</code>		
Sample Input	<p>Scenario 1 Input:</p> <pre>throwerror("This is a custom error!!");</pre> <p>Scenario 2 Input:</p> <pre>throwerror("This is a system error!!",true);</pre>		
Sample Output	<p>Scenario 1 Output:</p> <p>Message displayed to the user "This is a custom error!!"</p> <p>Scenario 2 Output:</p> <p>Message displayed to the user: "An unknown error has occurred. Please contact system administrator."</p> <p>The error log contains the following message: "This is a system error!!"</p>		

APPLY TEMPLATE

applytemplate()			
Description	This function applies a set of token key-value pairs to the template file using JSON data. This function supports JSON data along with the already-existing dictionary payload. If a value is present in both the dictionary payload and JSON data for the same key, the JSON data takes precedence over the dictionary payload.		
Parameters	templateFileLocation	String	The location of the template file.
	payload	Dictionary	The payload containing values for the defined tokens in the template.
			Optional. The default value is null.
	defaultErrorMessage	String	The default error message to be displayed in case any error is encountered.
			Optional. The default value is null.
jsonIdentifier	Json	The JSON containing values for the defined tokens in the template.	
		Optional. The default value is null.	
Syntax	String applytemplate(String templateFileLocation [, Dictionary payload [, String defaultErrorMessage [, Json jsonIdentifier]])		
Sample Input	<p>Template file test.txt: 4</p> <div style="border: 1px solid black; padding: 5px; margin: 5px 0;"> <p>This is user defined variable VAR1, value = {{VAR1}}. This is user defined variable VAR2, value = {{VAR2}}. This is user defined variable VAR3, value = {{VAR3}}.</p> </div> <p>BML:</p> <pre>templateFileLocation = "\$BASE_PATH\$/ApplytemplateTest/test.txt"; payload = dict("string"); put(payload, "VAR1", "payloadVal1"); put(payload, "VAR2", "payloadVal2"); jsonObj = json("{\"VAR2\":\"jsonVal2\",\"VAR3\":\"jsonVal3\"}"); output = applytemplate(templateFileLocation, payload, "", jsonObj);</pre>		
Sample Output	<p>This is user defined variable VAR1, value = payloadVal1. This is user defined variable VAR2, value = jsonVal2. This is user defined variable VAR3, value = jsonVal3.</p>		

BML PRINT LOG

Print the output of BML print statements to the log file for easier debugging of errors related to functions referenced in Subscription Ordering scripts. An "Enable BML print logging" option was added to the General Site Options area of the CPQ Cloud Administration Platform. Use this option to enable or disable the logging of print statements in BML. By default, BML Print Logging is disabled.

When BML Print Logging is enabled and an error is encountered, an error message is logged in the error logs. All of the print statements after the line throwing error are not logged into the error logs. This behavior is used to debug scripts by providing multiple print statements and observing the print statements that are not logged in the error logs. This points administrators to the location where the error is thrown.

For Example: The figure below shows two print statements and a piece of code in-between, which is used to throw an error. After executing the BML script, the error logs will contain "start" but not "end". As a result, the administrator can conclude that the error is thrown somewhere after "start" and before "end".

```
print "start";  
  
a= 1/0;  
  
print "end";
```

```
06 May 2016 01:52:40,398 [BMLPRINT] - start  
06 May 2016 01:52:40,400 [ERROR] com.bm.xchange.serviceproviders.AbstractServiceProvider.invoke(AbstractServiceProvider.java:268) - Error while invoking  
operation:  
Divide by 0  
.....
```

bm.log file

Figure 47: Sample BML Print Log

NOTE: Administrators should disable the "Enable BML print logging" option in production. Oracle recommends that no sensitive content be printed to the logs. Oracle does not assume responsibility for the data placed in the logs.

KEY RESOURCES

- For an overview of JSON data, refer to www.json.org.
- CPQ Cloud Online Help: Refer to the Using BML topics.

DOCUMENT DESIGNER ENHANCEMENTS

Document Designer is a drag and drop tool for creating and administering document templates. In CPQ Cloud 2016 R1, Document Designer introduces the following new enhancements:

- Performance Enhancements
- Formatting and Style Enhancements
- Contract Negotiation Enhancements

PERFORMANCE ENHANCEMENTS

New performance enhancements reduce the amount of time taken to load or save template files. In previous versions of CPQ Cloud, loading large templates took a long time and sometimes resulted in the browser timing out. Document Designer templates now load faster, allowing administrators to access and edit templates more quickly.

FORMATTING AND STYLE ENHANCEMENTS

The following formatting and style enhancements are available in CPQ Cloud 2016 R1:

- Continuous Sections
- Conditional Background Images
- Column Break Element
- Expand/Collapse All Buttons
- Header and Footer Margins
- Table Alignment

CONTINUOUS SECTIONS

Use a **Continuous Section** to split a single Document Designer section into two or more continuous sections without a page break between each section. A continuous section functions as a child of the parent section and inherits the margins, page orientation, and background image of the parent section. When dragged, a continuous section moves with its parent section. Parent and continuous sections are identified by icons that display in the section's header.

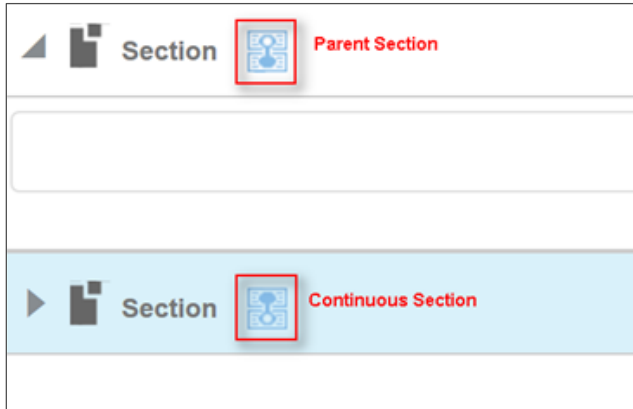


Figure 48: Parent Section and Continuous Section

In CPQ Cloud 2016 R1, a **Continuous Section** property is available in the **Section Properties** panel. The **Section Properties** panel is accessed by clicking a specific section of a Document Designer template. Selecting the **Continuous Section** checkbox makes the selected section a child of the preceding section.

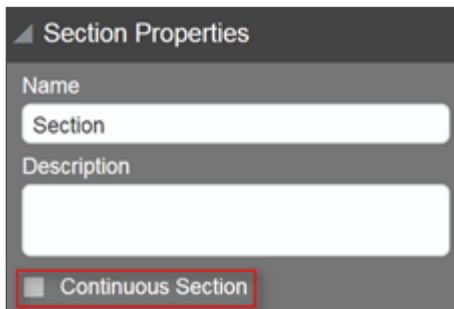


Figure 49: Continuous Section Check Box

TIPS AND CONSIDERATIONS FOR CONTINUOUS SECTIONS

Consider the following tips when using the new Continuous Section feature:

- The **Continuous** checkbox is disabled for the section at the beginning of a layout and the sections after the **Header**, **Footer**, and **Table of Contents** elements.
- Administrators can apply conditions and loops to both sections and continuous sections. When the **Apply to Continuous Sections** checkbox is selected from the **Loop** dialog or the **Conditional** dialog for the parent section, the loop or condition applies to all of the continuous sections.

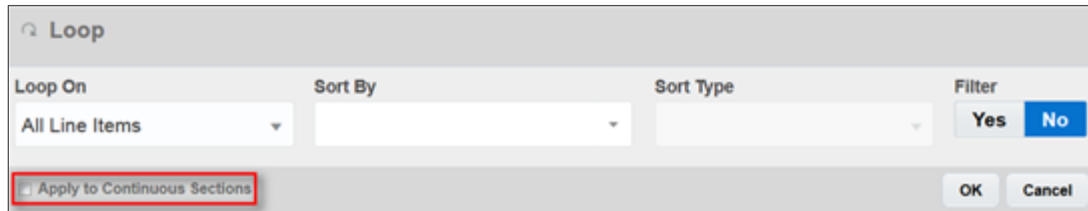


Figure 50: Apply to Continuous Sections Check Box – Loops

- Page-related properties are disabled when working with continuous sections. Only non-page-related properties are editable in continuous sections.
- When a section and a continuous section are both selected, the **Section Properties** panel does not display any properties.
- Administrators cannot delete a section with continuous sections unless the **Continuous Section** checkbox is unchecked for the sections.

CONDITIONAL BACKGROUND IMAGES

CPQ Cloud 2016 R1 introduces two new enhancements related to background images: Background images in .DOCX output files and conditional background images. In both cases, administrators can set the background image or the conditional background image at either the document level (i.e. throughout the document) or the section level (i.e. within a specific section of the document).

NOTE: Use the **Document Properties** panel to set a background image or a conditional background image at the document level. Use the **Section Properties** panel to set a background image or a conditional background image at the section level. The same **Background Image** fields and buttons are available in both the **Document Properties** panel and the **Section Properties** panel.

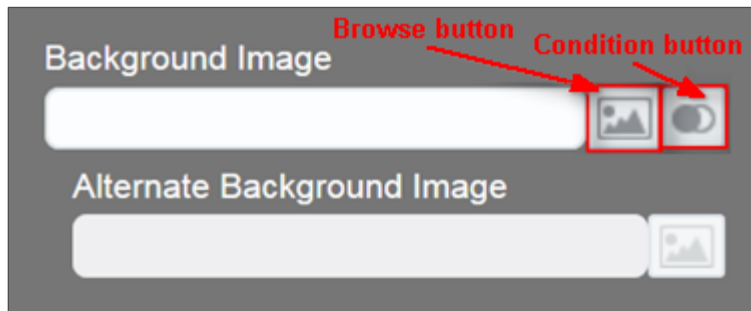


Figure 51: Browse to Background Image for .DOCX Output File

Use the **Browse File Manager** window to select the image to display in the .DOCX output file.

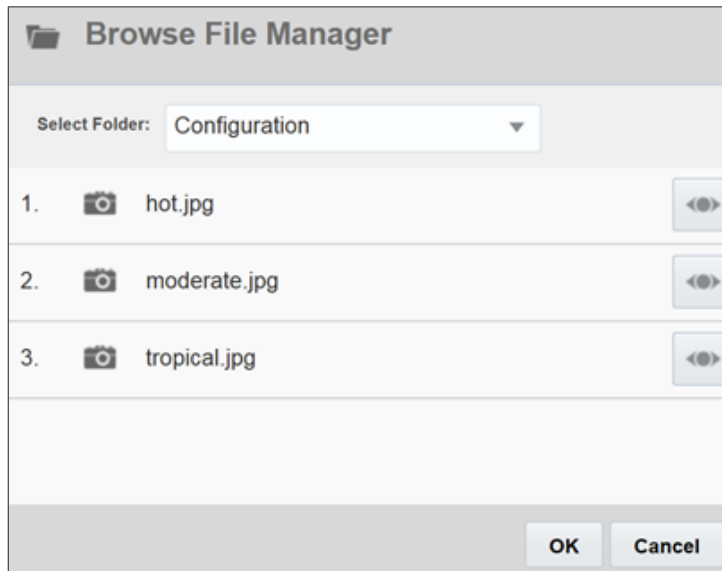


Figure 52: Browse File Manager

Administrators can set a conditional background image at either the document or section level. When a condition is set, administrators can place a background image on a template and show, hide, or change an image based on the condition.

Conditions are evaluated as either True or False. If true, the image in the **Background Image** field displays. If False, administrators can specify another image in the **Alternate Background Image** field. If no image is specified in **the Alternate Background Image** field, no background image displays. When no condition is set, the image selected in the first field becomes the default background image.

If a section's background image results in a blank URL displaying, the URL of the document's background image will display instead.

For example: Assume there is a background image (e.g. document.jpg) set at the document level and a conditional background image set at the section level (e.g. section.jpg) in the same document. No image is specified in the **Alternate Background Image** field. If the section's condition is set to true, then section.jpg is shown. If the section's condition is false, then document.jpg is shown. If **the Background Image** field is left blank but an alternate image is specified (e.g. alternate.jpg), the alternate image is shown.

To specify a conditional background image:

1. Open the **Document Properties** panel to specify a condition at the document level.
-or-
Open the **Section Properties** panel to specify a condition at the section level.
2. Click the **Condition** button next to the **Background Image** field to open the **Conditional** dialog and enter a condition.
3. Click **OK**.

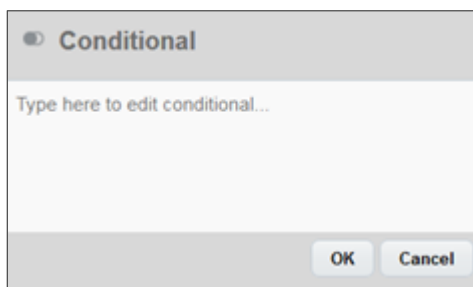


Figure 53: Conditional Dialog

COLUMN BREAK ELEMENT

The column break element is a new type of element introduced in CPQ Cloud 2016 R1. Drag and drop one or more column break elements onto a section to start the content in the next column. In a section with only a single column, the column break behaves like a page break. Column break elements only display in Document Designer and are not visible to users in the output document.

The **Column Break** element is available in the **Elements** panel of Document Designer.

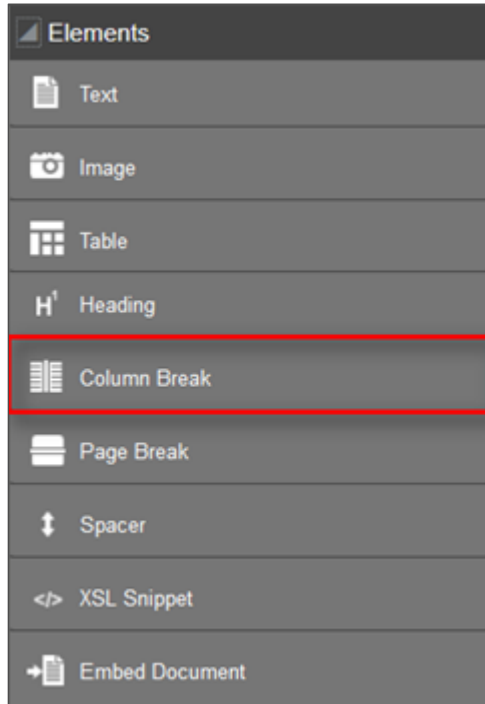


Figure 54: Column Break Element in Elements Panel

NOTE: A column break element can be placed in a section but not in a table cell, header, or footer.

EXPAND ALL/COLLAPSE ALL BUTTONS

Use the **Expand All** and **Collapse All** buttons in the header of the Document Designer Editor to expand or collapse all Document Designer layout objects (e.g. Section, Table of Contents, Header & Footer) in a template.

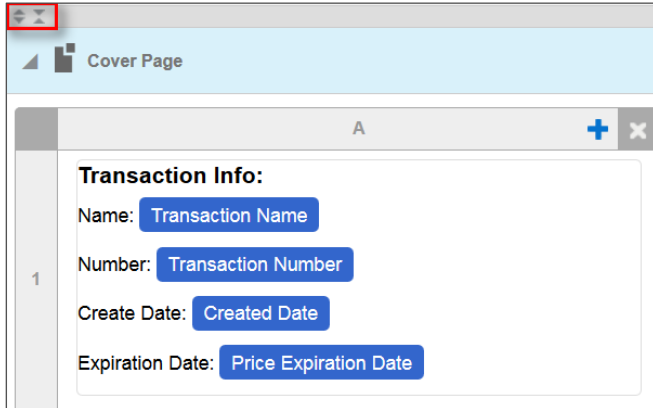


Figure 55: Expand All & Collapse All Buttons

NOTE: Using the Expand All action for large templates with several sections may have an impact on performance.

When expanding sections that contain more than approximately 500 components, the following warning message displays, “The maximum number of supported elements was exceeded in some sections, so not all elements are displayed. Please redistribute elements among continuous sections. Then, save and reload the template.”

To avoid performance issues when working with continuous sections, consider the following tips:

- Split large sections into small continuous sections.
- Save the template frequently as a general best practice.
- Do not have several sections expanded at the same time.
- After saving a section and before working in a new section, refresh the page to clear the browser’s cache.
- If parts of a document are static, consider creating separate files and embedding the files as components.
- As a good practice, Oracle recommends not creating extremely large sections.

HEADER AND FOOTER MARGINS

Separate header and footer margin fields are now available and can be applied to sub-sections individually. The top and bottom margins of a section are applied after the **Header** and **Footer** margins are applied.

To access the **Margin** fields, click on a header or footer to open the **Properties** panel for the header and footer.

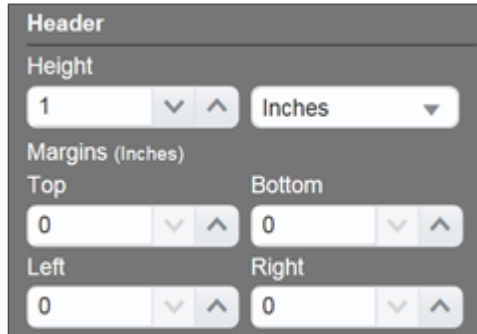


Figure 56: Header Margins

NOTE: When printing output files of type .DOCX or PDF and the left and right header and footer margins are different from the left and right page margins, the section's left and right margins are applied instead of the header and footer margins and page margins.

TABLE ALIGNMENT

Change the alignment of a table by left-aligning, centering, or right-aligning the table based on the width of the page. By default, tables are left-aligned. When multiple tables are selected, the selected table alignment option applies to all of the tables. The new **Table Alignment** property is available in the **Elements** panel under **Table** properties.

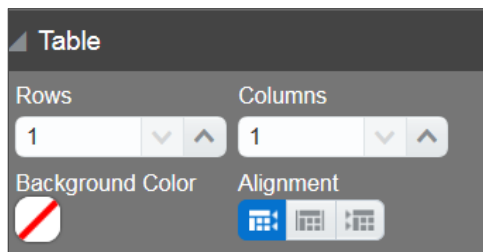


Figure 57: Table Alignment

CONTRACT NEGOTIATION ENHANCEMENTS

The following CPQ Cloud 2016 R1 Document Designer enhancements support the Contract Negotiation feature introduced in the 2016 R1 release. While designed to support Contract Negotiation, users can include the following Document Designer enhancements on any Document Designer template when Contract Negotiation is enabled on their CPQ site. For additional information, refer to the [Contract Negotiation](#) section of this document.

- **Track Changes:** Allow users to leverage Microsoft Word's track changes feature in .DOCX output files. The Track Changes feature is enabled in .DOCX output files by selecting the **Track Changes** checkbox in the **Document Properties** panel of Document Designer.

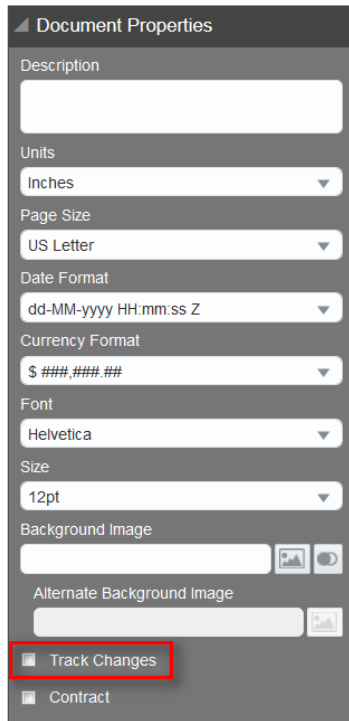


Figure 58: Track Changes Check Box

- **eSignature Vendor:** Integrate with DocuSign and place an eSignature tag attribute within a text or heading element in a section, table, header, or footer of a Document Designer template.

NOTE: For additional information about the Contract Negotiation feature available in CPQ Cloud 2016 R1, refer to the Contract Negotiation section of this document.

STEPS TO ENABLE

The CPQ Cloud 2016 R1 Document Designer enhancements are automatically available on CPQ Cloud 2016 R1 sites.

KNOWN ISSUES

Consider the following known issues when working with Document Designer:

- When a large number of elements (e.g. text elements, images, table cells) are included in a section, issues with search results may occur. As a best practice, keep the number of elements within individual sections to 300 or less.
- When the maximum number of supported elements is exceeded in a section, a warning message displays and only a partial list of elements is available in the Editor. If the **Copy** button is clicked, only the elements displayed in the Editor are copied. When the copied section is then pasted into the document, the previously mentioned warning message displays.
- When an attribute is added to a text element and the attribute is then deleted from the **Process Definition** area of CPQ Cloud (**Admin > Commerce and Documents > Process Definition**), an error message does not initially display on the text element. When the document is re-opened, the error message will then display on the text element.
- When a Commerce attribute is deleted while a Document Designer template is open, a validation error does not display and changes made to the template are still saved. If the template is re-loaded, the validation error then displays.

KEY RESOURCES

CPQ Cloud Online Help: Refer to the Document Designer topics.

LONG RUNNING THREAD DIAGNOSTICS

Long running thread diagnostics provide administrators with more tools to isolate and resolve performance issues. Prior to this enhancement, administrators had to shut down and restart an entire site to terminate long running actions. Diagnostics for long running requests that exceed a customizable time threshold are now provided for all Commerce and Configuration actions. There are three Timeout action settings: Kill and Log, Log Only, and No Action.

TIMEOUT ACTION SETTINGS

The following table summarizes actions performed for each of the Timeout Action settings.

Timeout Action	Kill Action Request	Error Message Displayed	Event Sent to Slow Thread Log	Event Sent to Performance Log
Kill and Log	✓	✓	✓	✓
Log Only			✓	✓
No Action				

For example, when an action exceeds the Time Threshold and the Timeout Action is Kill and Log, the system will perform the following actions:

- Send a Kill Action request
- Record the event in the Slow Thread Log
- Record the event in the Performance Log
- Display an error message on the page where the action was initiated

⊗ Could not perform Action: Longer Delay Action
⊗ Your last action timed out and was not completed successfully, and your quote has not been updated. If the problem persists you may need to contact your site administrator.
⊗ Action Advanced Modify – After Formulas failed. Process : Migration Document : Quote Action : Longer Delay Action

Figure 59: Long Running Thread Diagnostics Error Message

NOTE: When the system sends a Kill Request, the terminating action waits for an interruptible point in the long running action.

VIEW LOGS

The system records events in the Performance Log and the Slow Thread Log to aid administrators in troubleshooting. Event logging occurs when actions exceed the Timeout Threshold and the Timeout Action is set to Kill and Log or Log Only.

PERFORMANCE LOGS

The Performance Log provides an overall snapshot of system performance. Administrators can access **Performance Logs** from the Admin Home page in the **Developer Tools** section.

Performance Logs							
Date	User	Object	Event Type	Action	Server	Browser	
07/29/2016 1:14...	superuser	layouts	Resource Opera...	QUERY	2		
07/29/2016 1:15...	superuser	layouts	Resource Opera...	QUERY	7		
07/29/2016 1:15...	superuser	lookupTypes	Resource Opera...	QUERY	27		
07/29/2016 1:15...	superuser	resources	Resource Opera...	QUERY	20		
07/29/2016 1:15...	superuser	layouts	Resource Opera...	QUERY	158		
07/29/2016 1:15...	superuser	resources	Resource Opera...	QUERY	62		
07/29/2016 1:18...	superuser	iconSets	Resource Opera...	QUERY	14		
07/29/2016 1:18...	superuser	layouts	Resource Opera...	QUERY	2		
07/29/2016 1:18...	superuser	layouts	Resource Opera...	QUERY	10		
07/29/2016 1:18...	superuser	lookupTypes	Resource Opera...	QUERY	25		
07/29/2016 1:18...	superuser	resources	Resource Opera...	QUERY	24		
07/29/2016 1:18...	superuser	layouts	Resource Opera...	QUERY	96		
07/29/2016 1:18...	superuser	resources	Resource Opera...	QUERY	87		
07/29/2016 1:18...	superuser	iconSets	Resource Opera...	QUERY	8		
07/29/2016 1:23...	superuser	layouts	Resource Opera...	QUERY	2		

Figure 60: Performance Logs

SLOW THREAD LOG

The Slow Thread Log file provides detailed information regarding errors. To view the Slow Thread Log:

1. Click **Admin** to go to the Admin Home page.
2. Click Error Logs in the Developer Tools section.
3. Click `slow_thread_executions.log`.



The screenshot shows a web browser window with the URL `https://cpq-extdev-07.oracle.com/admin/log/log_file.jsp`. The page title is **Log File slow_thread_executions.log - Last 5242k**. The log content is as follows:

```
30 Jun 2016 11:43:34,004 [SLOWTHREAD] com.bm.xchange.threads.ThreadRegistry.handleAction(ThreadRegistry.java:323) - Thread 12 exceed its timeout of 60000ms,
Login: developer
Action: slowAction
Ref Obj: formula.main
BS ID: 18318275
Stack Trace at the time of interrupt:
java.lang.Exception
  at java.lang.Integer.valueOf(Integer.java:582)
  at com.bm.xchange.util.BMContext.getBMLArrayRowSizeLimit(BMContext.java:1700)
  at com.bm.xchange.services.scripts.bmjsript.compiler.StandardScript.Functions.checkArrayDimension(StandardScriptFunctions.java:1355)
  at com.bm.xchange.services.scripts.bmjsript.BMLArray.setData(BMLArray.java:285)
  at com.bm.xchange.bmjsript.commerce.document.action.advancedpremodify.CommerceProcess_183182247.exec(CommerceProcess_183182247.java:48)
  at com.bm.xchange.services.commerce.rule.CmBsRuleServiceImpl.exec(CmBsRuleServiceImpl.java:358)
  at com.bm.xchange.services.commerce.rule.CmBsRuleServiceImpl.exec(CmBsRuleServiceImpl.java:174)
  at com.bm.xchange.services.commerce.rule.CmBsRuleServiceImpl.exec(CmBsRuleServiceImpl.java:163)
  at com.bm.xchange.services.commerce.rule.CmBsRuleServiceImpl.exec(CmBsRuleServiceImpl.java:151)
  at com.bm.xchange.services.commerce.rule.CmUtilapplyBulkAdvanceRule(CmUtil.java:2781)
  at com.bm.xchange.services.commerce.rule.CmUtilapplyBulkAdvanceRule(CmUtil.java:2745)
```

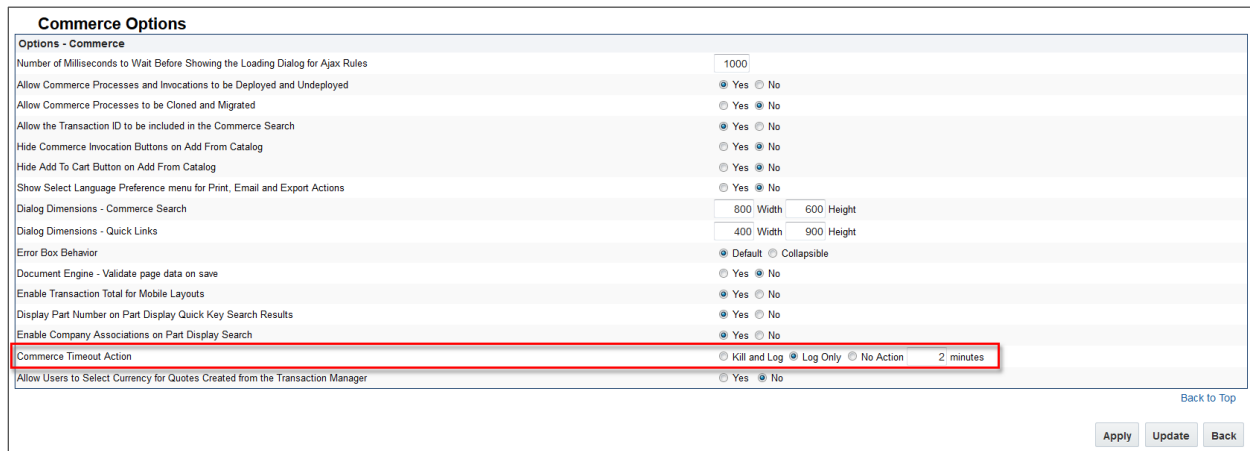
Figure 61: Slow Thread Log

TIMEOUT ACTION SETTINGS

The settings for long running thread diagnostics are available from the **Commerce Settings** and **Configuration Settings** pages. The long running thread diagnostics setup on these pages controls all actions for their respective areas. The default **Timeout Action** setting is **Log Only**, and the **Timeout Threshold** is set to **2 minutes**.

COMMERCE TIMEOUT ACTION SETTINGS

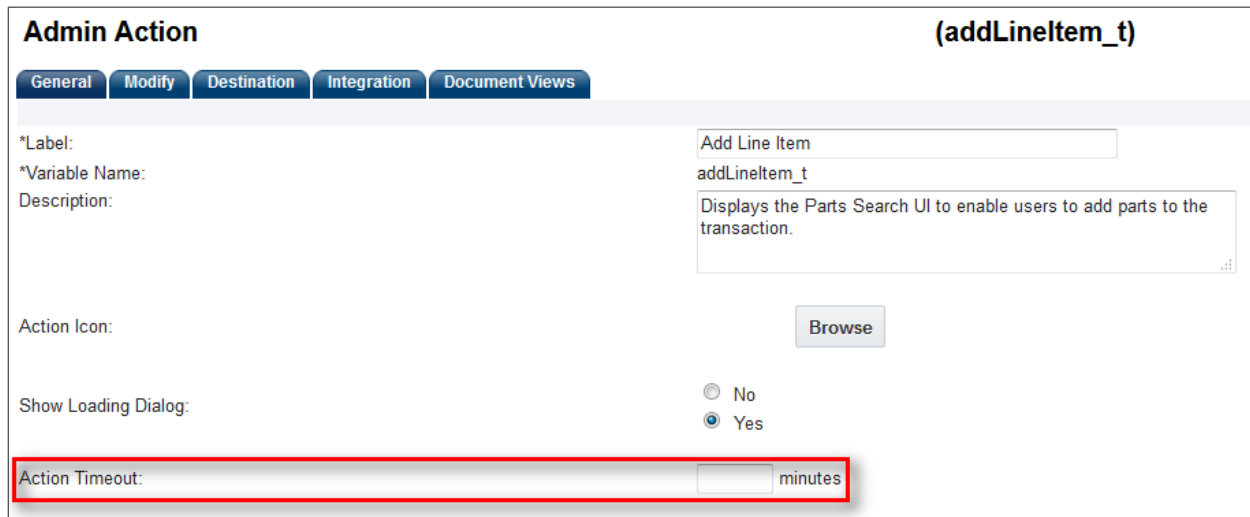
Administrators can set **Timeout Actions** and **Thresholds** for all Commerce actions from the **Commerce Options** page.



The screenshot shows the 'Commerce Options' configuration page. The 'Commerce Timeout Action' setting is highlighted with a red box. It is set to 'Log Only' with a '2 minutes' threshold. Other settings include 'Number of Milliseconds to Wait Before Showing the Loading Dialog for Ajax Rules' (1000), 'Allow Commerce Processes and Invocations to be Deployed and Undeployed' (Yes), 'Allow Commerce Processes to be Cloned and Migrated' (Yes), 'Allow the Transaction ID to be included in the Commerce Search' (Yes), 'Hide Commerce Invocation Buttons on Add From Catalog' (Yes), 'Hide Add To Cart Button on Add From Catalog' (Yes), 'Show Select Language Preference menu for Print, Email and Export Actions' (Yes), 'Dialog Dimensions - Commerce Search' (800 Width, 600 Height), 'Dialog Dimensions - Quick Links' (400 Width, 900 Height), 'Error Box Behavior' (Default), 'Document Engine - Validate page data on save' (Yes), 'Enable Transaction Total for Mobile Layouts' (Yes), 'Display Part Number on Part Display Quick Key Search Results' (Yes), 'Enable Company Associations on Part Display Search' (Yes), and 'Allow Users to Select Currency for Quotes Created from the Transaction Manager' (Yes). Buttons for 'Apply', 'Update', and 'Back' are at the bottom right.

Figure 62: Commerce Timeout Action

In addition, administrators can set **Timeout Thresholds** on individual Commerce actions. The specific Commerce action **Timeout Threshold** setting takes precedence over the general Commerce setting. This allows the administrator to isolate performance issues on a single Commerce action.



The screenshot shows the 'Admin Action' configuration page for the action 'addLineItem_t'. The 'Action Timeout' setting is highlighted with a red box and is set to 'minutes'. Other settings include '*Label:' (Add Line Item), '*Variable Name:' (addLineItem_t), 'Description:' (Displays the Parts Search UI to enable users to add parts to the transaction.), 'Action Icon:' (Browse button), and 'Show Loading Dialog:' (Yes). Buttons for 'General', 'Modify', 'Destination', 'Integration', and 'Document Views' are at the top.

Figure 63: Individual Commerce Action Timeout

CONFIGURATION TIMEOUT ACTION SETTING

Administrators can set **Timeout Actions** and **Thresholds** for all Configuration actions from the **Configuration Options** page.

Configuration Options	
Options - Configuration UI	
Number of Configurable Attributes Needed to Trigger the Processing Dialog	<input type="text" value="5"/>
Number of Milliseconds to Wait Before Showing the Loading Dialog for Ajax Rules	<input type="text" value="1000"/>
Number of Tabs to Display on each Row in the Model Configuration Page	<input type="text" value="8"/>
Options - Configurable Price	
Display Price on All Pages	<input checked="" type="radio"/> Yes <input type="radio"/> No
Hide Zero Total Price	<input checked="" type="radio"/> Yes <input type="radio"/> No
Display Total Sum of Recommended Items	<input checked="" type="radio"/> Yes <input type="radio"/> No <input type="radio"/> Yes with Subtotal of Mandatory Items
Options - Recommended Item	
Display Items on End Page	<input type="radio"/> Yes <input checked="" type="radio"/> No <input type="radio"/> Hide <input type="radio"/> Hide Main Model
Hide Missing Spares	<input type="radio"/> Yes <input checked="" type="radio"/> No
Select Recommended Items by Default	<input checked="" type="radio"/> Yes <input type="radio"/> No
Sum Recommended Items Quantities	<input checked="" type="radio"/> Yes <input type="radio"/> No
Configuration Save Options	
Disable auto-updates on BOM-Mapping rules	<input type="radio"/> Yes <input checked="" type="radio"/> No
Options - BOM Mapping	
Configuration Save Options	
Enable auto save of Pending Configurations	<input type="radio"/> Yes <input checked="" type="radio"/> No
Configuration Debug Options	
Configuration 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"/>
Back to Top	
<input type="button" value="Apply"/> <input type="button" value="Update"/> <input type="button" value="Back"/>	

Figure 64: Configuration Timeout Action

STEPS TO ENABLE

Long Running Thread Diagnostics are automatically available on all 2016 R1 sites.

KEY RESOURCES

CPQ Cloud Online Help: Refer to the Configuration Settings, Commerce Process Settings, and Long Running Thread Diagnostics topics.

INTEGRATION

CPQ Cloud administrators can leverage the power of CPQ Cloud by integrating with other software applications. CPQ Cloud administrators can use these point-to-point integration solutions as defined out-of-the-box or enhance the provided integration patterns to meet their distinct information technology landscapes and requirements.

New integration features available in CPQ Cloud 2016 R1 include:

- [Salesforce1 Integration](#)
- [CPQ Cloud – eBusiness Suite \(EBS\) BOM Reference Integration](#)
- [Process Cloud Service \(PCS\) Integration](#)
- [Rest APIs](#)
- [Platform as a Service \(PaaS\) Integration Sample Applications](#)

SALESFORCE1 INTEGRATION

The Salesforce1 mobile offering from Salesforce was delivered with new mobile standards that are supported by CPQ Cloud 2016 R1. Using the Salesforce Commerce Integration Managed Package, administrators can integrate the CPQ Cloud mobile UI with the Salesforce1 mobile UI. This integration supports all of the existing capabilities of the desktop integration between CPQ Cloud and Salesforce on a mobile device. The Salesforce1 integration is packaged in the Salesforce Reference Application, which includes updated cascading style sheets (CSS) to make the CPQ mobile experience look more like Salesforce1.

NOTE: The Salesforce1 integration does not affect the CPQ Cloud-Salesforce desktop integration. The Salesforce Commerce Integration Managed Package supports both the mobile integration and the desktop integration.

STEPS TO ENABLE

The Salesforce Commerce Integration Managed Package is installed in Salesforce to integrate with CPQ Cloud. Version 7.0 of the Salesforce Commerce Integration Managed Package supports Salesforce1. For instructions on how to integrate the CPQ Cloud mobile UI with the Salesforce1 mobile UI, refer to version 7.0 of *the Oracle CPQ Cloud – Salesforce Commerce Integration Managed Package Implementation Guide*.

TIPS AND CONSIDERATIONS

Consider the following tips when using the Salesforce1 integration:

- When using Salesforce1 to generate a quote, iPad users are able to print the quote, opening a PDF or Word file within Oracle CPQ Cloud on the Salesforce1 app. However, the same does not work on Android devices due to a Salesforce1 platform limitation. When the Print option is selected from a quote on an Android device in Salesforce1, the system attempts to download the file and open it in a browser. Since the browser is a separate app, the Salesforce1 authentication will not work, and thus the document cannot be viewed. Customers may work around this issue by sending the document via an email action for users on Android devices.
- When the New button is clicked from the Salesforce1 mobile application to create a new quote and the Cancel button is then clicked to cancel the quote, the **Salesforce1 Home** page or a blank page sometimes displays instead of the **Oracle Quotes and Orders** page.
- When using the Salesforce1 application on an Android device, a blank page displays upon navigating to a CPQ Cloud quote. As a workaround for this Salesforce1 issue, complete the following steps upon initial installation of Salesforce1
 - Navigate to Settings > Applications > Application Manager > SF1 App.
 - Click Manage Storage.
 - Click “Yes” to log out current users.

Upon logging back in, the issue is resolved and users can successfully navigate to a CPQ Cloud quote from an android device.

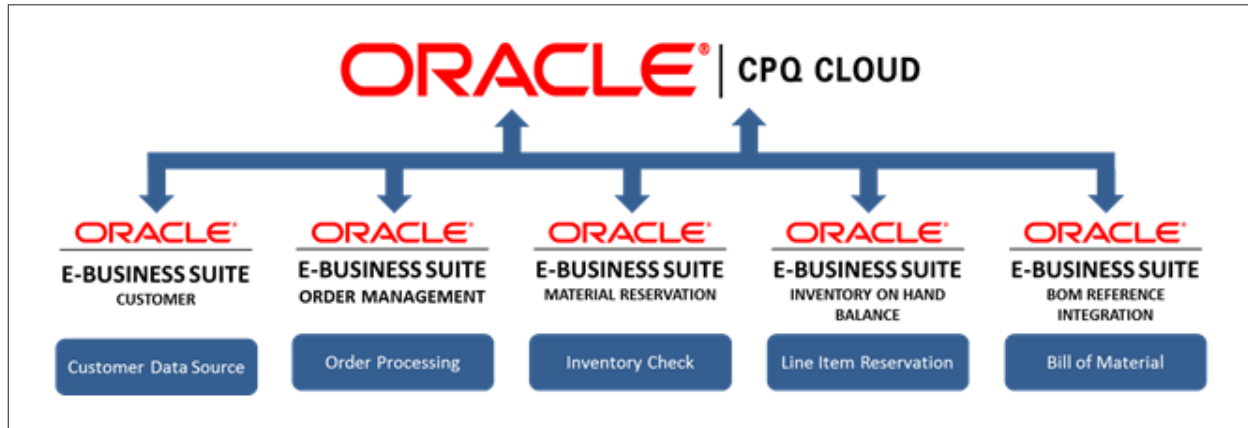
NOTE: Refer to Oracle - CPQ Cloud Salesforce Commerce Integration Managed Package version 7.0 Implementation Guide for Supported Devices, Operating Systems, and Browsers.

KEY RESOURCES

- Salesforce Reference Application: The Salesforce Reference Application applies to both Salesforce and Salesforce1.
- Salesforce1: Upgrade to Salesforce1 to use Salesforce on a mobile device.
- Salesforce Commerce Integration Managed Package version 7.0: Install version 7.0 of the Salesforce Commerce Integration Managed Package to integrate the CPQ Cloud mobile UI with the Salesforce1 mobile UI.
- Oracle - CPQ Cloud Salesforce Commerce Integration Managed Package version 7.0 Implementation Guide: Refer to the implementation guide for additional implementation guidance.

CPQ CLOUD – EBUSINESS SUITE (EBS) BOM REFERENCE INTEGRATION

The Oracle eBusiness Suite (EBS) integration introduced in CPQ Cloud 2015 R1 enabled direct integration to EBS Customer, EBS Order Management, EBS Inventory On Hand Balance, and EBS Material Reservation for common CPQ Cloud interactions. In 2016 R1, CPQ Cloud extends the EBS integration with support of multi-level Bills of Material (BOMs). CPQ Cloud integrations with EBS Customer and EBS Order Management are required to implement an EBS BOM Reference Integration.



STEPS TO ENABLE

CPQ Cloud

- Implement a CPQ Cloud - EBS Reference integration on any 2016 R1 or later CPQ Cloud site with the base Reference Application deployed and the BOM Mapping rules implemented. Refer to the [BOM Mapping](#) section for more detailed information.
- Migration Packages will be released for the EBS integrations so that administrators can download the components needed for integration directly to CPQ Cloud.

E-Business Suite

Individual EBS applications must use EBS Release 12 or later to integrate with CPQ Cloud. Additionally, EBS SOAP web services must be running to enable integration. Enabling these web services requires an Internet Service Gateway (ISG) to be running on the EBS environment. See the CPQ Cloud-EBS Integration Implementation Guide (see the Key Resources section) for a complete list of necessary EBS web services.

TIPS AND CONSIDERATIONS

Consider the following tip when using the CPQ Cloud - EBS BOM integration:

Existing CPQ Cloud customers who do not have the base Reference Application deployed can still integrate CPQ Cloud with EBS, but will require additional implementation setup to apply the integration components to their sites. Contact your Customer Success Manager for more information.

KEY RESOURCES

[My Oracle Support](#) contains the following documents to assist with CPQ Cloud-EBS integration implementation:

- CPQ Cloud & E-Business Suite Integration Implementation Overview
- CPQ-EBS Integration Implementation Guide
- BOM Mapping Implementation Guide

PROCESS CLOUD SERVICE (PCS) INTEGRATION

CPQ Cloud provides native workflow management features for Administrators to define complex approval flows. However, customers may want to use remote approval systems to consolidate all of their approvals in a single workflow engine such as Oracle Process Cloud Service (PCS), or Salesforce Approval Processes. These remote applications provide a common configuration and common attributes to manage approvals for an entire suite of applications, eliminating migration of approval data from one system into another. When a customer chooses remote approvals, approval requests are sent to external approval systems for processing. After approval completion, the associated approve/reject action is performed on the remote approval system. The remote approval system calls the appropriate approve/reject CPQ REST Endpoint to update the CPQ quote history and status.

Before you can leverage the pre-built PCS flows, you must establish a connection between PCS and CPQ Cloud. For information on how to set up these connections, refer to the CPQ Cloud – PCS Integration Implementation Guide.

The Remote Approval system processes Approvals when:

- The Oracle Process Cloud Service or other approvals service is set up.
- The remote approval sequence is enabled.
- Remote approval process functions are defined.
- A user submits a Quote for approval.

APPROVALS OVERVIEW

The following tables summarize the differences in approval flow actions and resulting behavior when using native CPQ Approval Flows versus Remote Approval Flows.

CPQ APPROVAL FLOW SUMMARY

CPQ Approval Flow	
Submit	<ul style="list-style-type: none">• The Request Action is called internally. The reason graph is evaluated and the reasons are presented to the submitter.• The submitter can pass on any comments to the approvers.
Approval	<ul style="list-style-type: none">• Approvers are notified that their approval is required. They provide their approval or rejection. The number of approvers depends on the defined reason graph.
Final Approval	<ul style="list-style-type: none">• If the quote is approved, after the last approver has approved, the submit action is fired internally to transition the quote to the approved stage.• If the quote is rejected, the approval status is changed to rejected, and the quote transitions back to the saved status.
Revise	<ul style="list-style-type: none">• Resets the entire process.

REMOTE APPROVAL FLOW SUMMARY

Remote Approval Flow	
Submit	<ul style="list-style-type: none">• The Request Action is called internally.• BML is executed to communicate with the remote approval system, the response is parsed, and the Remote Process ID is stored as an attribute.
Approval	<ul style="list-style-type: none">• Once the approval process has been completed, the Remote Approval System makes a REST call to the corresponding CPQ approve or reject action.
Final Approval	<ul style="list-style-type: none">• BML is executed from the approve or reject action to poll the remote approval system for the approval history data. This data is inserted into the CPQ approval history and the process is completed.
Revise	<ul style="list-style-type: none">• BML is executed to communicate with the remote approval system to cancel/abort the process and Remote Process ID is reset.• Resets the entire process.

APPROVAL SEQUENCE SELECTION

The **Approval Sequence** selection has been added to the **Admin Action** page for submit actions. Administrators select **Use Remote** to enable the remote approval sequence.

The screenshot shows the 'Admin Action' configuration page for '(remoteSubmit)'. The page has tabs for 'General', 'Modify', 'Integration', and 'Document Views'. The 'Approval Sequence' section is highlighted with a red box, showing two radio buttons: 'Use Approvals' (unselected) and 'Use Remote' (selected). Other sections include 'Label' (Remote Submit), 'Variable Name' (remoteSubmit), 'Description', 'Action Icon' (Browse), 'Show Loading Dialog' (No), 'Action Timeout' (minutes), 'Desktop Layout Path' (Unassigned), 'Mobile Layout Path' (Unassigned), 'Advanced Modify - Before Formulas', 'Advanced Modify - After Formulas', and 'Advanced Validation'. There are 'Define Function' buttons for the advanced modify and validation sections. At the bottom, there are buttons for 'Translations', 'Apply', 'Update', 'Update and New', and 'Back'. A note at the bottom right states: '* Changes to the document will not be saved when the action is performed, and transition rules will not trigger.'

Figure 65: Enable Remote Approval Sequence

NOTE: If a user disables the remote approval sequence - by selecting **Use Approvals** - any Integration Processes tied to the remote approvals will no longer execute; these processes are effectively disabled as well.

REMOTE APPROVAL PROCESS FUNCTIONS

If a remote approval is enabled, a **Remote Call Processing - Define Function** button appears to define the calls to be made to the Remote approvals site. The administrator must then set up BML functions for remote request approval, approve, and reject. Set up is not required for revise, which uses the same process flow as CPQ revise.

- Request Approval - The administrator writes BML to send a call to the Remote Approval System to initiate the approvals process, parse the response, and return the Process ID. Created functions will execute at run time. Failure to create a function will throw an error at runtime.
- Approve/ Reject - The administrator writes BML to send a call to the Remote Approval System to pull the approval history data and return it in a tilde-separated string. Failure to create a function will complete the approval process without updating the approval history.

NOTE: Refer to the CPQ Cloud - PCS Implementation Guide for Best Practices and instructions on building BML functions. The guide is available on [My Oracle Support](#).

Admin Action (request_approval_remoteSubmit) Document : Oracle Quote to Order > Transaction

General Modify Integration

*Label: Request Approval[Remote Submit]
*Variable Name: request_approval_remoteSubmit
Description: This action is performed when submit action is clicked and there is at least one reason to approve.

Action Icon: Browse

Show Loading Dialog: No Yes

Action Timeout: minutes

Remote Call Processing Define Function

Advanced Modify - Before Formulas No Advanced Modify - Before Formulas Define Advanced Modify - Before Formulas Define Function

Advanced Modify - After Formulas No Advanced Modify - After Formulas Define Advanced Modify - After Formulas Define Function

Advanced Validation Simple Validations Define Validation Rules (deprecated - use Commerce Rules) Define Function
 Save Without Validating
 Modify Without Saving or Validating

* Auto Submit

* Changes to the document will not be saved when the action is performed, and transition rules will not trigger.

Translations Apply Update Update and New Back

Figure 66: Remote Call Processing - Define Function

STEPS TO ENABLE

Refer to the CPQ Cloud - PCS Implementation Guide for detailed implementation instructions.

TIPS AND CONSIDERATIONS

You must use Oracle CPQ Cloud 2016 R1 or later to integrate with PCS.

NOTE: When using remote approval functionality, do not use REST/SOAP web services to initiate approvals. The remote approval sequence does not support initiating approvals with REST/SOAP web services. The only web service supported, to notify CPQ Cloud of the approval status, is on the REST objects for approve and reject sub action REST Endpoints.

KEY RESOURCES

[CPQ Cloud – PCS Integration Implementation Guide](#): Provides an overview of remote approvals and instructions on how to implement the CPQ Cloud – PCS integration.

In 2016 R1, CPQ continues our effort to expose objects through REST APIs and RESTful standards to empower customers to powerfully extend the capabilities of their CPQ Cloud implementations. These services use HTTP standards to promote simpler API calls and robust integrations. New APIs, available in CPQ Cloud 2016 R1, provide functionality for Contract Negotiations and Subscription Ordering.

[Contract Negotiation REST APIs](#)

- DOCX Compare - Compares two Contract DOCX documents and returns a list of differences.
- DOCX Merge - Merges approved changes into a Contract DOCX.

[Subscription Ordering REST APIs](#)

- CRUD Services for Assets consists of actions for asset maintenance.
- Synchronize Custom Operation synchronizes the payload edit with the hierarchy contents.
- Export Custom Operations consists of actions to export and retrieve asset objects.
- Import Custom Operations consists of actions to upload CSV files and import the data into asset object files.

[Enhancements to Query Parameters](#) - includes additions to the expand parameter and \$like operator support for the q parameter.

INTERFACE CATALOG

Documentation on all CPQ Cloud SOAP and REST services can be obtained by accessing the **Interface Catalogs** page from **Admin > Integration Platform > Interface Catalogs**.

Interface Catalogs

Search

Name

Description

Interface Type:

Interface Type	Name	Description
REST	commerceDocumentsOraclecpqTransaction	Main (Header Level) Commerce Document - serves as Quote/Order depending on step in process flow
REST	customOracle_BomAttr_Tr	
REST	customOracle_BomAttrDef	
REST	customOracle_BomAttrMap	
REST	customOracle_BomItemDef	
REST	customOracle_BomItemMap	
REST	customBomItem	
REST	customStatus	
REST	documentGenerator	
REST	emailGenerator	
REST	docxMerge	
REST	docxCompare	
REST	assets	Used to store an instance of the product owned by a customer.
REST	performanceLogs	Used to store Performance Log for Application Events

Figure 67: Interface Catalogs Page

REST API Endpoint

The endpoint for each REST API appends onto `http://{siteurl}`, where `{siteurl}` is the base URL of the CPQ Cloud site.

REST APIS FOR CONTRACT NEGOTIATIONS

This section describes DOCX Compare and DOCX Merge REST APIs.

DOCX COMPARE REST API

DOCX Compare	
Description	This operation compares two Contract DOCX documents created by Document Designer and returns a list of modified clauses.
URI Endpoint	/rest/v2/docxCompare
Parameters	processVarname Variable Name of Commerce Process
	transactionId ID of Transaction
	oldDocAttachId ID of File Attachment for Old Document
	newDocAttachId ID of File Attachment for New Document
HTTP Method	POST
Response Diffs (Array of JSON objects)	clauseId ID of changed clause
	type Enum value of change type, eg: "MODIFIED"
	clause label Label of changed clause
	parentlabel Label of parent clause
	reviewed Changes have been approved in Word
	action English label of 'type', eg: "Updated"
Sample Response	<pre>{ "diffs": [{ "clauseId": "123456", "type": "MODIFIED", "clauselabel": "Clause A Name", "parentlabel": "Parent Clause Name", "reviewed": "Y", "action": "Updated" }, { "clauseId": "123789", "type": "MODIFIED", "clauselabel": "Clause B Name", "parentlabel": "Parent Clause Name", "reviewed": "Y", "action": "Updated" }] }</pre>

DOCX MERGE REST API

DOCX Merge		
Description	This operation merges changes into a Contract DOCX document using a list of approved changes and another Contract document as a source. It returns the updated document as a base64 string.	
URI Endpoint	/rest/v2/docxMerge	
Parameters	processVarname	Variable Name of Commerce Process
	transactionId	ID of Transaction
	sourceFileAttachId	ID of File Attachment for Source Document
	targetFileAttachId	ID of File Attachment for Merge Target Document
	approvedChanges	Array of changes to be applied
HTTP Method	POST	
Response	errors	Array of error messages
	base64Doc	String base64 document

REST APIS FOR SUBSCRIPTION ORDERING

CPQ Cloud exposes asset data through REST APIs. Asset REST APIs support create, read, update, and delete (CRUD) operations. They also provide synchronize, import, and export operations.

- CRUD Operations: [Get Asset List \(GET\)](#) | [Get Asset \(GET\)](#) | [Create Asset \(POST\)](#) | [Update Asset \(POST\)](#) | [Delete Asset \(DELETE\)](#)
- [Synchronize Custom Operation](#)
- Custom Export Operations export and retrieve an asset object.
[Export Action](#) | [Get Exported File](#)
- Custom Import Operations upload CSV files and import the data into asset object files.
[Upload File](#) | [Import Action](#) | [Get Import Log File](#)

GET ASSET LIST REST API

Get Asset List (GET)	
Description	This operation returns all the assets data.
URI Endpoint	<code>/rest/v2/assets</code>
Parameters	<p>None</p> <p>Query specifications that follow CPQ Cloud query and pagination parameters syntax, and query specifications that follow a subset of MongoDB syntax can organize or narrow return data. For more information, see the Query Specification Syntax article in CPQ Cloud Online Help.</p>
HTTP Method	GET
Success Response	JSON data of all assets
Sample Payload Request	None (nothing should be in the request payload)
Notes	<ul style="list-style-type: none"> Returns all attributes from the asset object in JSON format. Returns Currency and Foreign Key (FK) attributes as complex attributes. Refer to the Create Asset REST API for a sample complex attribute.

GET ASSET REST API

Get Asset (GET)			
Description	This returns asset data for a specific asset.		
URI Endpoint	<code>/rest/v2/assets/{id}</code>		
Parameters	<table border="1"> <tr> <td>{id}</td> <td>The unique ID of the Asset</td> </tr> </table> <p>Query specifications that follow CPQ Cloud query and pagination parameters syntax, and query specifications that follow a subset of MongoDB syntax can organize or narrow return data. For more information, see the Query Specification Syntax article in CPQ Cloud Online Help.</p>	{id}	The unique ID of the Asset
{id}	The unique ID of the Asset		
HTTP Method	GET		
Success Response	JSON data of the asset		
Sample Payload Request	None		
Notes	<ul style="list-style-type: none"> Returns all attributes from the asset object in JSON format. Returns Currency and Foreign Key (FK) attributes as complex attributes. Refer to the Create Asset REST API for a sample complex attribute. 		

CREATE ASSET REST API

Create Asset (POST)	
Description	This operation creates a new asset.
URI Endpoint	/rest/v2/assets
Parameters	None
HTTP Method	POST
Success Response	JSON data for the new asset is created.
Sample Payload Request	<pre>{ "partNumber": "part1", "quantity": "1.0", "displayKey": "display-100-2-1234", "customer": "SpecialAccount100", "assetKey": "abo_ae100", "discountPercent": "5.0", "discountAmount": { "currency": "USD", "value": 15.0 }, "currency": { "currencyCode": "USD" }, "fixedRecurringAmount": { "currency": "USD", "value": 50.00 }, "oneTimeNetAmount": { "currency": "USD", "value": 300.00 } }</pre>
Notes	<ul style="list-style-type: none"> • Currency is a Foreign Key (FK) attribute. Complex is the expected input type. • Example complex type value for Currency attributes: discountAmount, fixedRecurringAmount, oneTimeNetAmount

UPDATE ASSET REST API

Update Asset (POST)	
Description	This operation updates an existing asset.
URI Endpoint	/rest/v2/assets/{id}
Parameters	{id} The unique ID of the Asset to update
HTTP Method	POST
Success Response	Creation of the JSON data for the updated asset.
Sample Payload Request	<pre>{ "partNumber": "part1", "quantity": "1.0", "displayKey": "display-100-2-1234", "customer": "SpecialAccount100", "assetKey": "abo_ae100", "discountPercent": "5.0", "discountAmount": { "currency": "USD", "value": 15.0 }, "currency": { "currencyCode": "USD" }, "fixedRecurringAmount": { "currency": "USD", "value": 50.00 }, "oneTimeNetAmount": { "currency": "USD", "value": 300.00 } }</pre>

DELETE ASSET REST API

Delete Asset (DELETE)	
Description	This operation deletes an existing asset
URI Endpoint	/rest/v2/assets/{id}
Parameters	{id} The unique ID of the Asset to delete
HTTP Method	DELETE
Success Response	HTTP 200 OK

Synchronize Custom Operation	
Description	This operation invokes a Synchronize action using the payload to edit the hierarchy contents.
URI Endpoint	<code>/rest/v2/assets/actions/synchronize</code>
Input Payload Parameters	<p>Documents</p> <ul style="list-style-type: none"> Type: JSON Holds a row or collection. Each row can include an optional “_sync_action” operation parameter.
	<p>_client_driven_action</p> <ul style="list-style-type: none"> Type: Boolean Enables apply changes instead of replacing the target asset object. Currently this parameter must be set to ‘true’.
Output Payload Parameter	<p>Documents</p> <ul style="list-style-type: none"> Type: JSON Holds a row or collection. Each row can include an optional “_sync_status” operation parameter.
HTTP Method	POST
Success Response	The JSON data for the asset object that was updated
Sample Request Payload	Synchronize Request Payload Sample
Sample Response Payload	Synchronize Response Payload Sample
Notes	<ul style="list-style-type: none"> When updating a collection, utilize the user key to refer to the target asset row. The _sync_action parameter supports add, modify, and delete values. The _sync_status parameter values include created and updated. The _proxy_id parameter, added to each row in the input payload, enables parameter cross references. The value for each row must be unique. The response payload does not include deleted objects assets.

SAMPLE SYNCHRONIZE REQUEST PAYLOAD

```
{
  "_client_driven_action": true,
  "documents": {
    "items": [
      {
        "assetKey": "root_L1",
        "partNumber": "part1_update",
        "_proxy_id": "11",
        "_sync_action": "update",
        "childAssets": {
          "items": [
            {
              "assetKey": "child_L2",
              "partNumber": "part2_update",
              "_proxy_id": "1",
              "_sync_action": "update",
              "childAssets": {
                "items": [
                  {
                    "assetKey": "child_child_L3_1",
                    "partNumber": "part3_update",
                    "_proxy_id": "2",
                    "_sync_action": "update"
                  },
                  {
                    "assetKey": "child_child_L3_2",
                    "_proxy_id": "3",
                    "_sync_action": "delete"
                  },
                  {
                    "assetKey": "child_child_L3_3",
                    "partNumber": "part5",
                    "customer": "SpecialAccount100",
                    "displayKey": "display-100-2-1234",
                    "_proxy_id": "4",
                    "_sync_action": "create"
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

SAMPLE SYNCHRONIZE RESPONSE PAYLOAD

```
{
  "_client_driven_action": true,
  "documents": {
    "items": [
      {
        "assetKey": "root_L1",
        "partNumber": "part1_update",
        "_proxy_id": "11",
        "_sync_action": "update",
        "childAssets": {
          "items": [
            {
              "assetKey": "child_L2",
              "partNumber": "part2_update",
              "_proxy_id": "1",
              "_sync_action": "update",
              "childAssets": {
                "items": [
                  {
                    "assetKey": "child_child_L3_1",
                    "partNumber": "part3_update",
                    "_proxy_id": "2",
                    "_sync_action": "update"
                  },
                  {
                    "assetKey": "child_child_L3_2",
                    "_proxy_id": "3",
                    "_sync_action": "delete"
                  },
                  {
                    "assetKey": "child_child_L3_3",
                    "partNumber": "part5",
                    "customer": "SpecialAccount100",
                    "displayKey": "display-100-2-1234",
                    "_proxy_id": "4",
                    "_sync_action": "create"
                  }
                ]
              }
            }
          ]
        }
      }
    ]
  }
}
```

CUSTOM EXPORT OPERATIONS

The following actions provide the capability to export an asset collection or single instance to a CSV file and retrieve the results.

- [Export Action](#) exports an asset collection or single instance to a CSV file.
- [Get Exported File](#) retrieves the CSV file downloaded on the browser.

EXPORT ACTION REST API

Export Action	
Description	This operation invokes an Export action to export an asset collection or single instance to a CSV file. It exports the object attributes as column headers and the key attributes with the dotted notation.
URI Endpoint	<code>/rest/v2/assets/actions/export</code>
Input Payload Parameter	Criteria Type: JSON Requirements for the asset collection or single instance row export.
Output Payload Parameter	exportedFileName Type: JSON This parameter contains the absolute path to the JSON control file used in the consecutive call to retrieve the CSV file.
HTTP Method	POST
Success Response	The JSON data of the absolute path of the control file.
Sample URI Endpoints	<code>/rest/v2/assets/actions/export</code> <code>/rest/v2/assets/{\$asset_id}/actions/export</code>
Sample Request Payload	<pre>{ "criteria":{ "fields":["assetKey","displayKey"], "orderby":["assetKey:desc"], "q":{"assetKey":{"\$eq: 'asdfgh'}}" } }</pre>
Sample Response Payload	JSON data of the asset object.

Export Action	
Notes	<ul style="list-style-type: none"> • Use the user key of the instance when exporting a single asset instance. • This API only supports CSV format. • Exports use dot notation for Foreign Key (FK) attributes. For example, an asset object has an FK attribute called 'rootAsset'. When querying data from this asset, the REST service returns a JSON object for the 'rootAsset' attribute. The JSON object contains the id and 'assetKey' of the 'rootAsset' object. The export operation exports these attributes as 'rootAsset.id,rootAsset.assetKey'. • Exports also use dot notation for Currency attributes. For example, an asset object has a Currency attribute called 'discountAmount'. When querying data from this asset, the Rest service returns a JSON object for the 'discountAmount' attribute. The JSON object contains the value and Currency of the 'discountAmount' attribute. The export operation exports these attributes as 'discountAmount.value,discountAmount.currency'. • Exports also use dot notation for LOVs.

GET EXPORTED FILE REST API

Get Exported File	
Description	This operation invokes a GET call to retrieve the CSV file downloaded on the browser.
URI Endpoint	/rest/v2/files/fileName
Parameter	fileName This parameter contains the control file name from the export call response.
HTTP Method	GET
Success Response	The CSV file with the data.
Sample URI Endpoint	/rest/v2/files/assets_1464387178004
Sample Request Payload	None
Sample Response Payload	Refer to the exported Asset CSV data file.
Notes	These calls perform user access validation, so the same user must perform the export and get file calls.

CUSTOM IMPORT OPERATIONS

- [Upload File Operation](#) uploads a CSV file and returns a control file used for import.
- [Import Action](#) imports the uploaded CSV file to modify data for a particular resource. It supports add, modify, and delete operations.
- [Get Import Log File](#) retrieves the import log file.

UPLOAD FILE OPERATION REST API

Upload File Operation	
Description	This operation invokes a POST call to upload a file from the user's file system. It returns a control file used for the import action.
URI Endpoint	<code>/rest/v2/files</code>
File Parameters	attachment The CSV file
	Header content-disposition
	Media-type application-JSON/octet-stream
HTTP Method(s)	POST
Success Response	The response provides the absolute path to the control file, which contains user information and the absolute path to the CSV to be imported.
Sample URI Endpoint	<code>/rest/v2/files/</code>
Sample Request Payload	CSV file
Sample Response Payload	<code>/rest/v2/files/upload_1465847155416</code>
Notes	<ul style="list-style-type: none"> • The import CSV file should contain a column called <code>'_sync_action'</code> to specify actions to perform. If the CSV file does not contain this column, the default upsert action is used. • Use dot notation to represent Foreign Key (FK) attributes. For example, an asset object has an FK attribute called <code>'rootAsset'</code>. The import operation should import these attributes as <code>'rootAsset.id,rootAsset.assetKey'</code>. • Use dot notation to represent Currency attributes. For example, an asset has a Currency attribute called <code>'discountAmount'</code>. The import operation needs to import these attributes as <code>'discountAmount_value,discountAmount.currency'</code>. • Use dot notation to represent LOVs.

IMPORT ACTION REST API

Import Action	
Description	This operation invokes an Import action to process the uploaded CSV to import data into a specified. It supports create, update, and delete operations.
URI Endpoint	/rest/v2/assets/actions/import
Input Payload Parameter	fileName File name extracted from the response rest link in the previous call.
Output Payload Parameter	importLogFileName This parameter contains the absolute path to the control file, which contains user Information and the absolute path to the log file.
HTTP Method	POST
Success Response	The absolute path to the control file, which contains user Information and the absolute path to the log file.
Sample URI Endpoint	/rest/v2/assets/actions/import
Sample Request Payload	{ "fileName":"upload_1465847155416" }
Sample Response Payload	{ "importLogFileName": "https://cpq-072.us.oracle.com/rest/v2/files/assets_output_1465848183346" }

GET IMPORT LOG FILE REST API

Get Import Log File	
Description	Invokes a GET call to retrieve the actual import log file.
URI Endpoint	/rest/v2/files/fileName
Parameter	fileName The name of the import log file from the import call response
HTTP Method	GET
Success Response	The CSV file with the object data as the status (CSV column) of the import for each row and if the import failed for the row the error message (CSV column).
Sample URI Endpoint	/rest/v2/files/assets_output_1465848183346
Sample Request Payload	None
Sample Response Payload	CSV file
Notes	User access is validated, so both the import and get log file calls should be performed by the same user.

ENHANCEMENTS TO QUERY PARAMETERS

The following query enhancements are available with CPQ 2016 R1.

- [Expand Query Parameter](#)
- [\\$like Operator Support for the q Parameter](#)

EXPAND QUERY PARAMETER

Existing functionality allows expanding the root, which expands the first level children for the root. Additions in this release provide the following enhancements:

- Expand on any level. Child object references use dot notation.
- Recursive expand on linked objects with the syntax *.all.
- Activate selective fields during expand and recursive expand.

Query Parameters URL Format

`{resourceURI}?{param}={paramSpec}&{param}={paramSpec}&{param}={paramSpec}`

Expand Query Parameter Examples

Expand Query Parameter	Description / Example
all	This parameter expands an asset object and its children.
	<code>/rest/v2/dynamicResource/assets?expand=all</code>
childAssests.all	This parameter expands a child asset object and its children.
	<code>/rest/v2/dynamicResource/assets?expand=childAssests.all</code>
childAssets.wrongname.all	This parameter expands a non-existing object. Child wrongname at path childAssets.wrongname does not exist or is not accessible in childAssets object for current use.
	<code>/rest/v2/dynamic/assets?expand=childAssets.wrongname.all</code>
childAssets*.all	This parameter recursively expands an asset object and all children assets.
	<code>/rest/v2/dynamicResource/assets?expand=childAssets*.all</code>
childAssets*.all&fields=childAssets.partNumber	This parameter recursively expands and activates specified fields. It will only activate and retrieve fields specified in the query.
	<code>/rest/v2/dynamicResource/assets?expand=childAssets*.all&fields=childAssets.partNumber</code>

\$LIKE OPERATOR SUPPORT FOR Q PARAMETER

This addition provides sequel style "LIKE" operations via the \$regex operator, because this functionality is not supported by the MONGO standard. To ease usage, it also provides an extension "\$like" operator, which translates to the SQL LIKE operator more intuitively. Both cases support a Case Insensitive search option.

\$like Operator Examples:

Description	\$like Operator Example	\$regex Operator Example
FieldN contains "myValue"	?q={"FieldN": {"\$like: "%myValue%"}}	?q={"FieldN": {"\$regex: "myValue"}}
FieldN starts with "myValue"	?q={"FieldN": {"\$like: "myvalue%"}}	?q={"FieldN": {"\$regex: "^myvalue"}}
FieldN ends with "myValue"	?q={"FieldN": {"\$like: "%myvalue"}}	?q={"FieldN": {"\$regex: "myvalue\$"}}
FieldN matches "myValue"	?q={"FieldN": {"\$like: "myvalue"}}	?q={"FieldN": {"\$regex: "^myvalue\$"}}
FieldN matches "myValue" Case Insensitive	?q={"FieldN": {"\$like: "myvalue", \$options: "I"}}	?q={"FieldN": {"\$regex: "/^myvalue\$/i"}}

STEPS TO ENABLE

Query Parameter Enhancements, Contract Negotiation REST APIs, and Subscription Ordering REST APIs are automatically available on CPQ Cloud 2016 R1 sites.

TIPS AND CONSIDERATIONS

Consider the following tips when using the new REST APIs available in CPQ Cloud 2016 R1:

- REST APIs will not invoke Process actions.

KEY RESOURCES

- CPQ Cloud Online Help: Refer to the REST API topics.
- [MongoDB Query Documents](#)

PLATFORM AS A SERVICE (PaaS) INTEGRATION SAMPLE APPLICATIONS

Platform as a Service (PaaS), specifically PaaS-Software as a Service (SaaS) Extension allows customers to deploy their in-house web services and applications in a managed hosting environment to integrate with and extend the features offered by SaaS Solutions, such as CPQ Cloud. Sample applications delivered by CPQ Cloud exhibit design patterns that can solve some common use cases for integration between CPQ Cloud and PaaS-SaaS Extensions. The following sections provide an overview of the sample applications.

XLS TO CSV CONVERTER SAMPLE APPLICATION

This sample application exhibits a pattern where a web service hosted on Java Cloud Service (JCS)-SaaS Extension is used to transform data before using it in CPQ Cloud; in the sample, the web service simply transforms a Microsoft Excel® file to a CSV file. Out of the box, CPQ cannot read XLSX files in the rule engine, but it can read CSV files. Using this sample as a model, a web service can be developed to enable Sales Representatives to use the Excel to CSV Converter web service to update a quote using additional line item information provided in an Excel XLSX file. In another use case, a web service could be built to allow a sales user to upload an Excel file with new corporate rates for a set of products, and then apply the new pricing to a quote. The service exposes a REST Endpoint that takes an XLSX file and responds with the equivalent CSV data. The CPQ application is configured by an Admin user to call this endpoint within the appropriate workflow, so that the uploaded Excel file is transformed and used as expected by the use case. The web service endpoint is protected using OAuth, and the CPQ web application uses a confidential client to gain access to the service. The Secure Data Table Columns feature, delivered in 2016 R1, provides a method for securely storing credentials of the confidential client in CPQ Cloud.

QUOTE STATISTICS SAMPLE APPLICATION

This sample application displays analytic information for a quote on a UI that is embedded on the CPQ Cloud opportunity screen. Customer information is sent to the application, which then consumes data via CPQ Cloud REST APIs to generate information. The sample thus exhibits a strategy for embedding JCS-SaaS Extension application UI elements in CPQ Cloud, and a strategy for CPQ REST API access via the JCS-SaaS Extension. In this sample, the average net total across quotes for this customer is calculated and displayed on the CPQ opportunity screen. Although this application calculates a simple statistic, it can be used as a model to develop extended applications to provide information for Sales users. This information can help sales users make key decisions about the quotes they are generating and make adjustments to maximize conversion potential.

KEY RESOURCES

The PaaS Integration sample applications are available from [Oracle Cloud Developer Portal](#), part of the platform service offerings in Oracle Cloud. Each application will provide the sample application and step-by-step instructions to:

- Understand and prepare your PaaS environment
- Research and plan the topology, technologies, and security model used to develop your extension implementation
- Discover data objects exposed by CPQ that can be consumed by JCS - SaaS Extension
- Develop, deploy, and launch your application
- Monitor and troubleshoot your environment

PRE-UPGRADE CONSIDERATIONS

KNOWN FUNCTIONALITY

CONFIGURATOR WEB SERVICE SUPPORT TO BOM MAPPING

This release introduces the optional `bomprice` element to the Configuration web service (v1 and v2). The response includes the BOM price, if BOM mapping is enabled and contains a non-zero BOM price.

NOTE: BOM Mapping is available on all 2016 R1 sites, refer to the [BOM Mapping](#) section for additional information.

Sample Response:

```
<bm:price>
  <bm:bomPrice>$16.0000</bm:bomPrice>
  <bm:totalPrice>$45.0000</bm:totalPrice>
</bm:price>
```

The web service WSDL, upon regenerated, includes the newly introduced "bomPrice" optional element:

```
<xsd:complexType name="ConfigurationPriceType">
  <xsd:sequence>
    ...
    <xsd:element maxOccurs="1" minOccurs="0" name="bomPrice" nillable="true"
type="xsd:string"/>
    <xsd:element maxOccurs="1" minOccurs="1" name="totalPrice"
nillable="false" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

SECURE URL ADDRESSES

Confirm that all references to your CPQ Cloud URL, such as in customizations or third-party tools, use `https://` in the URL.

TRANSLATION

For some system-defined messages and components, some strings have been removed and others have been added in CPQ Cloud 2016 R1. If you have created your own implementation-specific translations of system-defined strings, some of these strings may no longer appear, and other strings may appear in English. The strings that appear in English are new, and need to be translated.

Most of these messages and components are on the Admin side of CPQ Cloud, but you should review both your end user and administration pages before deploying your updated installation to confirm that all strings appear in the desired language.

MIGRATION

When migrating from one site to another using the Migration Center, both sites must use the same major release. Content may only be migrated across minor releases within the same major release. Migration across major releases cannot occur.

- “Major release” = A major product release, e.g. 2016 R1
- “Minor release” = A release update, e.g. 2015 R2 Update 5

RESOLVED KNOWN ISSUES

For information on bugs fixed in 2016 R1, refer to the 2016 R1 Resolved Known Issues document available on [My Oracle Support](#) and the CPQ Cloud Online Help.

TRANSLATION STATUS

CPQ Cloud supports the consumption of both single and multi-byte character sets. Submit a service request on [My Oracle Support](#) to enable your site for a new language.

For the following languages, a translation of the CPQ Cloud user interface is available for both the platform and the reference application:

- Chinese (Simplified) [China]
- Chinese (Traditional) [Taiwan]
- Czech [Czech Republic]
- Danish [Denmark]
- Dutch [Netherlands]
- English
- Finnish [Finland]
- French
- French [Canada]
- German
- Hungarian [Hungary]
- Italian
- Japanese [Japan]
- Korean [South Korea]
- Norwegian (Bokmål) [Norway]
- Polish [Poland]
- Portuguese [Brazil]
- Romanian [Romania]
- Russian [Russia]
- Spanish (Worldwide)
- Swedish [Sweden]
- Turkish [Turkey]

POST-UPGRADE CONSIDERATIONS

Upgrade and test all test instances on Oracle CPQ Cloud 2016 R1 before upgrading to production.

BROWSER SUPPORT

CPQ Cloud supports all browser versions that meet the criteria of the Oracle Software Web Browser Support Policy.

SUPPORTED BROWSERS

Windows

- Major releases of Google Chrome upon general browser availability and until Google no longer supports the version
- Major releases of Mozilla Firefox upon general browser availability and until Mozilla no longer supports the version
- Major releases of Internet Explorer/Microsoft Edge within nine months of general browser availability and until Microsoft no longer supports the version

Mac OS X

- Major releases of Google Chrome upon general browser availability and until Google no longer supports the browser version
- Major releases of Mozilla Firefox upon general browser availability and until Mozilla no longer supports the version
- Major releases of Safari within nine months of general browser availability and until Apple no longer supports the version

Android

- Major releases of Google Chrome upon general browser availability and until Google no longer supports the browser version

iOS

- Major releases of Safari within nine months of general browser availability and until Apple no longer supports the browser version.

If you experience issues using a supported browser version, open a ticket on [My Oracle Support](#) to resolve the issue. If an issue arises when using a supported browser, use a certified browser version until a fix is delivered. Certified browsers are selected based on current market share and are thoroughly tested to work with the current version's standard functionality.

CERTIFIED BROWSERS

Windows

- Google Chrome 51.x
- Mozilla Firefox 47.x
- Internet Explorer 11.x

Mac OS X

- Google Chrome 51.x
- Mozilla Firefox 47.x

Android

- Operating System: Android Lollipop 5.x
- Browser: Google Chrome 51.x
- Screen resolution: 2560 x 1600

iOS

- Operating System: iOS 9.x
- Browser: Safari 9.x
- Screen resolution: 2048 x 1536

NOTE: Compatibility issues with the selected browsers may exist when sites contain additional JavaScript, alternate CSS, or other custom functionality. Customizations may require add-on work. Contact [My Oracle Support](#) to determine the availability of workarounds and minor fixes.

SALESFORCE MANAGED PACKAGE SUPPORT

CPQ Cloud no longer releases updates to the Salesforce Managed Packages prior to v7.0. With the release of 2016 R1, only Managed Packages v7.x are officially supported. Although Salesforce integrations that use a Managed Package prior to v7.0 are still expected to function, new issues that arise in these versions are not addressed by CPQ Cloud.

TRAINING

Please refer to the release documentation for all versions between your current version and the version to which you are upgrading to see all new functionality, resolved known issues, and functional known issues.

Refer to the CPQ Cloud Online Help to become familiar with the new features introduced in Oracle CPQ Cloud 2016 R1. For additional help, see [My Oracle Support](#).

Verify any information not explicitly mentioned in this document as supported by the software against the product help for Oracle CPQ Cloud 2016 R1 or the Oracle CPQ Cloud Consulting team.

ADDITIONAL INFORMATION

For more information on Oracle CPQ Cloud, visit the [Oracle CPQ Cloud documentation site](#).

Copyright © 2016 Oracle and/or its affiliates. All rights reserved.

This document is provided for information purposes only, and the contents hereof are subject to change without notice. This document is not warranted to be error-free, nor subject to any other warranties or conditions, whether expressed orally or implied in law, including implied warranties and conditions of merchantability or fitness for a particular purpose. We specifically disclaim any liability with respect to this document, and no contractual obligations are formed either directly or indirectly by this document. This document may not be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without our prior written permission.

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.

Integrated Cloud Applications & Platform Services