ORACLE

Oracle CPQ Asset-Based Ordering Implementation Guide

Updates 19B and Later

April 2024

TABLE OF CONTENTS

REVISION HISTORY

This document will continue to evolve as existing sections change and we add new information. All updates appear in the following table:

DATE	WHAT'S CHANGED	NOTES
APR 2024	Oracle CPQ 24B Updates	
DEC 2024	Oracle CPQ 24A Updates	
SEP 2023	Oracle CPQ 23D Updates	
JUN 2023	Oracle CPQ 23C Updates	
MAR 2023	Oracle CPQ 23B Updates	
FEB 2022	Oracle CPQ 22B Updates	
OCT 2021	Oracle CPQ 21D Updates	
MAR 2021	Oracle CPQ 21B Updates	
DEC 2020	Oracle CPQ 21A Updates	
SEP 2020	Oracle CPQ 20A Updates	
JUL 2019	Oracle CPQ 19B Updates	
MAR 2019	Oracle CPQ 18D Updates	
JUL 2017	Oracle CPQ 2017 R1 Updates	
NOV 2016		Initial Document Creation, Oracle CPQ 2016 R1

A: INTRODUCTION

Asset-Based Ordering (ABO), also known as Subscription Ordering, is used to sell tangible assets or subscriptions for services delivered over a period of time (e.g. cell phone service, cable service, office Wi-Fi). The ABO functionality provides the ability to create and store customer assets in CPQ. Assets can be created, modified, suspended, resumed, renewed, and terminated. Customers can use ABO in conjunction with System Configuration, and they can use REST APIs to integrate with external applications.

A1: Purpose

Use this implementation guide together with the following resources to implement Asset-Based Ordering.

- **BOM Mapping Rules** - You must set up Bill of Material (BOM) Mapping Rules to enable ABO. Services and subscriptions are contained in the BOM as sellable objects. For additional information, refer to the *Oracle CPQ 18B BOM Mapping Implementation Guide*.
- **REST APIs** - Create, query, or modify assets using REST APIs and perform asset-based operations such as Terminate, Suspend, Resume, and Renew from an external client application. For additional information, refer to Asset REST APIs in Oracle CPQ Administrator Online Help or REST API Services for Oracle CPQ.
- **Subscription Workbench** (previously called the Customer Assets page) - Display, search, and manage assets.
- **Local Asset Repository** - Store asset information locally in the Oracle CPQ asset repository.
- **ABO Implementation Package** - Use the ABO implementation package to implement Asset-Based Ordering.

A2: ABO Implementation Package Versions

Oracle CPQ creates implementation packages to distribute components that are required to implement new features. Refer to the following files on My Oracle Support (Doc ID 2182966.1) for release specific details.

- **ABO Implementation Guides:**
  - ABO Implementation Guide 19B supports Oracle CPQ release and ABO Packages 19B and later
  - ABO Implementation Guide 18D supports Oracle CPQ releases and ABO Packages 18D through 19A
  - ABO Implementation Guide 2017 R1 supports Oracle CPQ releases and ABO Packages 2017 R1 through 18C
- **ABO RefApp Packages: 24A,** 23D, 23C**,** 23B, 21A, 19C, 19B, 19A, 18D, 18C, 18B, 18A, 17D, 2017R2, and 2017R1

  > **Notes:**
  > - ABO Package 24B is the latest published package for the new RefApp Commerce process that is based on the new Commerce Standard Process included with the CPQ platform release 24B.
  > - ABO Package 21A is the last published package for the RefApp Commerce process included with release 23A and earlier. Features in later releases requiring minor script adjustments are documented in this guide, longer scripts are included in BML text files (e.g. ABO_Final_BML_Actions_21D.zip).
  >
  >   For more details on the new Standard Process, please refer to Oracle CPQ Administrator Online Help > Commerce > Commerce Process > Standard Process.

  - **BML Text Files:**
    - The scripts are also available in txt format for easy comparison and reference in a release specific zip file. For example, the scripts included in the 24B ABO package is available in the zip file - ABO_Final_BML_Actions_24B.zip
    - To compare implementation differences between releases refer to the zip of BML text files that are named in the format - ABO_Final_BML_Actions_<ReleaseVersion>.zip, where ReleaseVersion indicates the CPQ release version. For example - ABO_Final_BML_Actions_24B.zip, ABO_Final_BML_Actions_24A.zip etc.

Oracle CPQ supports ABO by providing a combination of platform features (e.g. the Asset object, REST Operations, BML functions) and an implementation package containing customized components. Customers can implement Asset-Based Ordering in Oracle CPQ in three ways:

- By applying the CPQ ABO Implementation Package to a new site containing the CPQ Reference Application.
- By applying the CPQ ABO Implementation Package to an existing CPQ site, modifying variable references to use the appropriate site variables, and modifying scripts or logic referenced by the Asset-Based Ordering features.
- Implementing customized elements in CPQ that leverage the asset-related platform features.

A3: Upgrade Considerations

> **Note:** To identify your current ABO package version, view the comments in the Admin > BML Library > ORCL_ABO > abo_initializeContext function.

A3.1: ABO Implementation Package for Commerce Standard Process (ABO Platform Release 23B Onwards)

When upgrading to the 23B ABO or **later** platform release, consider the following points in addition to the information in the Upgrading to the 21A or Later ABO Implementation Package from 19B and Earlier Implementations and Upgrading to 18D and Later ABO Implementation Packages from Earlier Implementations sections.

- Customers already using Asset-Based Ordering (prior to ABO release 23B and not using the new commerce standard process), that want to upgrade to the Oracle CPQ 23B or **later** platform release, can continue to use the 21A ABO or earlier ABO implementation packages.
- Oracle CPQ delivers a new ABO implementation package in every release from Oracle CPQ 23B onwards to support the new Commerce Standard Process released in that specific Oracle CPQ platform release. The Standard Process simplifies administrator setup for new ABO implementations by providing a pre-defined set of ABO-related Commerce attributes, actions, library functions, and data columns.
    - Customers can import the new release specific ABO implementation package to add additional content such as Commerce library functions and util libraries.
    - Customers can make UI layout changes by adding attributes and actions as per their requirements.
    - Implementation details that are not relevant while using the new Commerce Standard Process and the latest ABO Implementation package available are noted.
    - All other sections are relevant and should be reviewed carefully during implementation.
- For customers already implementing the new Commerce Standard Process (available from Oracle CPQ 23B platform release onwards), upgrading to the latest Oracle CPQ platform release, importing the release specific ABO implementation package is optional. But post-upgrade if there is a need to import the ABO package again, then it is recommended to use the release specific ABO implementation package that is consistent with your platform release.
- Customers implementing the new Commerce Standard Process, for the first time, must import the release specific ABO implementation package, that is consistent with your platform release and optionally merge their customization manually to the new package after careful planning and testing during implementation.
- The following are some of the points to be noted for each release specific ABO Implementation Package available for the Commerce Standard Process, available since Oracle CPQ Platform Release 23B onwards:

CPQ Platform Release	ABO Implementation Package	What's Changed
Oracle CPQ 23B	ABO_RefApp_Package_2023B	This is the first ABO Implementation Package released for the new commerce standard process released in Oracle CPQ 23B.
Oracle CPQ 23C	ABO_RefApp_Package_2023C	There are no new changes included in the 23C ABO implementation package.

		The artifacts included in the 23C ABO implementation package are similar to the ones available in 23B ABO implementation package.
Oracle CPQ 23D	ABO_RefApp_Package_2023D	Most of the artifacts included in the 23D ABO implementation package are similar to the ones available in 23B and 23C ABO implementation package except for the following new enhancements: • Beginning in Oracle CPQ 23C, sales users can add products with fixed structures (Unconfigurable BOM hierarchies) to a transaction using Quick Add functionality. A product is considered to have a fixed structure when it is associated to a root BOM item and does not have a model configuration defined with it • To support this functionality, couple of minor changes have been made to the following artifacts in the 23D ABO implementation package: o Util library - abo_getConfigurationInstance o Commerce Library Function - oRCL_abo_PostDefaultsOnLineItems
Oracle CPQ 24A	ABO_RefApp_Package_2024A	The artifacts included in the 24A ABO implementation package are similar to the ones available in 23D ABO implementation package, except for a minor update to Util library - abo_getProductModelInfo.
Oracle CPQ 24B	ABO_RefApp_Package_2024B	The artifacts included in the 24B ABO implementation package are similar to the ones available in 24A ABO implementation package, except for minor updates to the following Util libraries: o *abo_copyRootToTransaction* o *abo_generatePAC*

o

A3.2: Upgrading to the 21A ABO Implementation Package from 19B and Earlier Implementations

When upgrading to the 21A ABO implementation package, consider the following points in addition to the information in the Upgrading to the 19B or Later ABO Implementation Package from 18D and Earlier Implementations and Upgrading to 18D and Later ABO Implementation Packages from Earlier Implementations sections.

- Due to the complexity of this package upgrade, we recommend you pursue updating the business-level time zone feature to your existing package and not performing the full 21A ABO package upgrade. Upgrading to the 21A ABO package is **optional and must be carefully planned and tested before implementing.**
- Existing customers wanting to upgrade to the 21A ABO package can upgrade and merge their customization to the new package or review the 21A ABO package text files with their current files and make the necessary modifications.
- If you are on the 19B ABO implementation package and do not want to upgrade to the 21A ABO implementation package but would like support for the 21D Reconfigure Line Item Grid Pricing Behavior for Child Models feature, update your Main and Sub-Document Reconfigure Actions. Refer to: Add ABO Script to Main Document Reconfigure Action and Add ABO Script to Sub-Document Reconfigure Action

  Existing customers wanting the business-level time zone feature but do not want to upgrade to the 21A ABO package will need to determine the differences between the 21A and 19B/C package and copy the new logic from 21A into existing ABO package. Additionally, the required logic within the "`abo_dateTimeConverter`" needs to be extracted and properly copied into the calling function to ensure the customized package has the equivalent logic as the 21A ABO package. Refer to Override Business Time Zone in ABO Implementation.

Within the 21A ABO package, the "`abo_loadDefaultContext`" BML function has been enhanced to include the "`businessTimeZone`" setting.

> **Note:** This above date field philosophy is recommended when the ABO package interacts with CPQ Commerce. However if Oracle CPQ integrates with other systems such as fulfillment or ICS, careful review and planning is recommended to ensure the date field within CPQ Commerce data is handled consistently across all integration channels.

A3.3: Upgrading to the 19B or Later ABO Implementation Package from 18D and Earlier Implementations

When upgrading to the 19B or later ABO implementation package, consider the following points in addition to the information in the Upgrading to 18D and Later ABO Implementation Packages from Earlier Implementations section.

- In Oracle CPQ 19B, the Customer Assets List page was enhanced and renamed to the Subscription Workbench. This enhancement decouples the Subscription Workbench page from the Transaction UI and allows users to view the assets/subscriptions belonging to a given account directly through a navigation link.

  > **Note:** You can set up a user-defined link to view the Subscription Workbench. If the user-defined link is not set up, the Subscription Workbench is only viewable from the Transaction.

- Oracle CPQ 19B introduced multi-select for asset actions. Multi-select of non-modify actions (i.e. renew, resume, suspend, and terminate) works with older ABO packages and does not require upgrade to the 19B ABO implementation package.

- You must install the ABO Implementation Package for Updates 19B and Later to use the multi-select for modify operations and new transaction support for asset-based operations.

  > **IMPORTANT:** The modify action can be customized to directly save to the transaction without launching Configuration UI by specifying "`returnBom=true`" in the modify button setting.

- Customers can perform asset actions from the Subscription Workbench for assets without an associated Transaction ID. When a user invokes an asset action for an asset without an associated Transaction ID, a new transaction is created and associated with the requested operation. Internally the "abo_prepareNewTxn" function is invoked to prepare the payload to create a new transaction.

  > **IMPORTANT:** You must upgrade to the 19B ABO implementation package to use this feature.
  > You should also customize the "abo_prepareNewTxn" and "abo_prepareNewTxn_ext" functions to identify which main document attribute to populate on the new transaction.

- Beginning in CPQ 19B, customers can enable simple product support for ABO to directly add products not associated with related configuration models to a transaction for an asset-based order. The sub-document "`requestDate_l`" attribute default logic was moved to sub-document advanced default to accommodate the simple product support. It populates the same "`requestDate`" for the root and all child items. For a new root line, it will populate the "`requestDate`" to empty if the main document "`defaultRequestDate_t`" is empty.

  > **IMPORTANT:** CPQ strongly recommends that you install the 19B ABO Implementation Package to use simple products.

- BML library functions in the 19B ABO implementation package continue to be imported to the "ORCL_ABO" folder.
- The ABO Data Table (i.e. Oracle_aboPart2Model) is optional in the 19B ABO Implementation package. This table is included in the 19B ABO Implementation package mainly for backward compatibility.
- ABO functions that were previously included in the Transaction and Transaction Line Reconfigure actions were moved to the "oRCL_abo_ReconfigureAction" Commerce library function in the 19B ABO Implementation Package.

  > **IMPORTANT:** You must update the Transaction and Transaction Line Reconfigure actions. For instructions, refer to Add ABO Script to Main Document Reconfigure Action and Add ABO Script to Sub-Document Reconfigure Action.

A3.4: Upgrading to 18D and Later ABO Implementation Packages from Earlier Implementations

Consider the following points when upgrading to 18D and later ABO implementation packages from 18C and earlier implementations:

- Beginning in CPQ 18D, ABO enhancements (such as using System Configuration models with ABO implementations) are not available to customers who continue to use ABO implementation packages from 18C or prior implementations.

- Beginning in the 18D ABO implementation package, the "`_line_bom_id`" and "`_line_bom_parent_id`" BOM attributes retain their original BOM ID values and are not overwritten with the assetKey data during the ABO flows. These BOM attribute values are no longer overwritten with the assetKey data stored in the sub-document "`itemInstanceId_l`" attribute.

  > **IMPORTANT:** If your integration depends on the BOM Id and parent Id attributes, you should evaluate the impact of integration change prior to upgrade.

- If you are using ABO implementation packages from 18D or later implementations, there are significant changes to apply and delta functionality. If you customized apply and delta functionality in releases prior to 18D, you should evaluate apply and delta changes prior to incorporating your customizations. Refer to Appendix C: ABO Feature Summaries, Appendix E: ABO Util Library Functions, and Appendix G: BML System Functions Variables for ABO.
- Beginning in CPQ 18D, you should use the "abo_applybom_ext" and "abo_delta_ext" BML functions to add your customized logic for apply and delta functionality respectively, if applicable. These BML library function extensions were added so customized behavior can be preserved in future upgrades.
- Beginning in the 18D ABO implementation package, some of the initialization parameters that were previously in the "abo_initializeContext" BML function were moved into the "defaultContextJson.txt" file. This file can be accessed under the ABO folder in the File Manager. Refer to Appendix J: Default JSON Text File for additional information.
- The BML text files for 2017 R1, 18D, and 19B are available on My Oracle Support (Doc ID 2182966.1) to allow comparison of the implementation differences between releases.
- The ABO Data Table (Oracle_aboPart2Model) remains the same as in the ABO implementation packages available in earlier releases.

  > **Note:** If you already have data in your own ABO data table, do not migrate or apply the ABO Data Table included in the ABO implementation.

- For customers using the 19B ABO implementation package, the ABO Data Table is optional. In this case, the ABO Data Table is only applicable if external asset repository is being used or for assets created using ABO implementation package from 18C or an earlier release.
- If you choose to apply the Commerce changes included in the ABO implementation package, you must consolidate any customizations you made to the Commerce ABO actions. Otherwise, the customizations will be overwritten when the ABO implementation package is imported.
- Oracle does not recommend using custom JavaScript with Oracle CPQ.
- Beginning in Oracle CPQ 18D, the "`_supplier_company_name`" system variable is available to determine the hosting company name unique to the site.
- The 18D ABO implementation package provided a sample implementation to directly update an asset based on the CPQ transaction. To implement this functionality, refer to Appendix K: Sample Update Assets Action.

Beginning in CPQ 18D, apply and delta functionality is implemented using the following BML functions:

- **applybom** – This BML function is used in ABO implementations to place the apply BOM that is passed as an input argument on top of the base BOM and return a resultant BOM.

  In 18C and earlier releases, this functionality was implemented using the "abo_apply" and "abo_applyXA" BML library functions. Beginning in CPQ 18D, the BML code to implement this functionality was significantly reduced in "abo_apply" by invoking "applybom" from this function.

- **abo_applybom_ext** – This BML library function can be used to customize or make incremental changes to the default apply logic implementation for one particular open order BOM item.

- **calculatedeltabom** - This BML function is used in ABO implementations to compare the final Configuration BOM with the previously calculated Projected Asset Cache (PAC) BOM to calculate delta changes and return a Delta BOM.

  In releases prior to 18D, this functionality was implemented using the following BML library functions: "abo_delta", "abo_processDeltaXA", "abo_processDeltaItem", and "abo_processDeltaBom". Beginning in CPQ 18D, the BML code to implement this functionality was significantly in reduced "abo_delta" by invoking "calculatedeltabo" from this function.

- **abo_delta_ext** - This BML function can be used to further customize and/or make any incremental changes to the default delta logic implementation.

> **IMPORTANT:** Beginning in 18D the following BML functions are deprecated and no longer used in the out-of-the-box package:
>
> - **abo_applyXA** was replaced by the "applybom" function.
> - **abo_processDeltaXA** was replaced by the "calculatedeltabom" function.
> - **abo_processDeltaItem** was replaced by the "calculatedeltabom" function.
> - **abo_processDeltaBom** was replaced by the "calculatedeltabom" function.

Beginning in CPQ 18D, unmapped configuration attributes are preserved during the modify process; this is implemented using the following:

- The **calculateconfiguration** BML function is used to calculate the projected configuration for a list of open order lines passed as input and returns a Configuration key used to load the projected state of the Configuration when the Model Configuration page is launched

- System Variables **BM_CONFIGURATION_KEY** and **BM_PRIOR_CONFIGURATION_KEY** – the configuration key returned by "calculateconfiguration" is sent to configurator via these two keys in "abo_getConfigInstance". The content of the global cache entry pointed to by these keys is the projected configuration corresponding to the "inputBom" and "configBom".

  > **Note:** Configuration data stored in the global cache entries can be examined using the diagnosis log but might change in future release, and should not be customized or manipulated.

- The configuration associated with an asset is stored on root asset and the Update Configuration REST API should be invoked to keep this info up-to-date. For more information, refer to Update Configuration REST API Operation.

A3.5: Using an ABO Implementation Package for 18C or Earlier Release

Customers who implemented ABO in an Oracle CPQ release prior to Oracle CPQ 18D can upgrade to the latest release and continue to use their existing ABO implementation package from the prior release for features available prior to 18C.

> **Note:** ABO enhancements from 18D or later releases (such as using System Configuration models with ABO implementations) are only available when customers implement the ABO implementation package from the applicable release.

B: ASSET-BASED ORDERING IMPLEMENTATION

ABO implementation tasks are organized in the following sections:

- Implementation Prerequisites contains tasks you need to perform prior to implementing or upgrading ABO.
- Import ABO Implementation Package provides instructions to import the ABO implementation package using the CPQ Migration Center.
- Application Modifications contains tasks you need to perform after you import the ABO implementation package.
- Advanced Implementation contains tasks you may need to perform for custom applications, upgrades, or if you choose not import required ABO items.
- Implement Additional Features contains task you need to take to implement additional features.

B1: Implementation Prerequisites

- Set up BOM Mapping Rules and implement a BOM for each configuration model used to create assets. The BOM contains the components that will be included in the associated assets and CPQ transactions used to create and manage assets. For additional BOM information, refer to the *Oracle CPQ 18B BOM Mapping Implementation Guide*.
- If you are upgrading from a prior implementation package, review Upgrade Considerations and Appendix C: ABO Feature Summaries.
- If you are implementing ABO on a custom commerce process (i.e. a process other than the Oracle Quote to Order process from the base reference application), Commerce attributes referenced by the ABO implementation must be included in the target process with matching type and domain. Refer to Appendix D: Commerce and Configuration Items.

B2: Import ABO Implementation Package

The ABO implementation package was designed for use with the Oracle Quote to Order (i.e. oraclecpqo) Commerce Process in the base reference application. Customers with a newly provisioned base reference application should apply all of the attributes, actions, data columns, and library functions from the ABO implementation package. Customers with existing application should review the changes included in the Commerce process, determine if there are existing customizations associated with the changes to the Commerce process, and consolidate the customization as needed.

Customers upgrading to Oracle CPQ 23B or **later** release, who wish to use the new Standard Process, can create a new commerce process that will provide a pre-defined set of attributes, actions, data columns and library functions. The 23B or later release specific ABO Implementation Package can be used to migrate commerce library functions and util libraries to support ABO implementation.

Complete the following steps to import the ABO implementation package to your site using the CPQ Migration Center.

> **Note:** Oracle recommends making note of any customized scripts, attribute menu items, formulas, etc. prior to importing the ABO implementation package.

1. Download the ABO Implementation package (<n>ABO RefApp Package) from My Oracle Support (Doc ID 2182966.1).
   The downloaded filename will be ABO_RefApp_Package_<n>.zip, where <n> represents the Oracle CPQ release.
2. Log in to Oracle CPQ and open the Admin Home page.
3. Click **Migration** in the Utilities section, the Migration Center opens.
4. Select **Import Package** from the **Offline Mode** drop-down menu.

5. In the Upload Package window, click **Browse**.
6. Select the ABO_RefApp_Package_<n>.zip file from your computer, and then click **Open**.
7. (Optional) Choose a target process for Cross Process Migration.

   > **Note:** You must perform the following procedure if you upload the ABO Implementation Package to a custom commerce process (i.e. a process other than Default Migration or Oracle Quote to Order): Enable ABO for a Custom Commerce Process.

8. In the Upload Package window, click **Upload**.
9. Click on the applicable Commerce process (e.g. Oracle Quote to Order) and deselect any Commerce items you do not want to override.

   > **Note:** Refer to Advanced Implementation for instructions to manually install required ABO components.

10. Click **Migrate**.
11. Enter an optional migration description in the Migration window, and then click **Migrate**.
12. Monitor the progress of the migration in the **Status** area at the bottom of the page.
13. When the migration completes, click **Logs** at the bottom of the page.
    The Migration Logs window opens and indicates the success or failure of the migration.

B3: Application Modifications

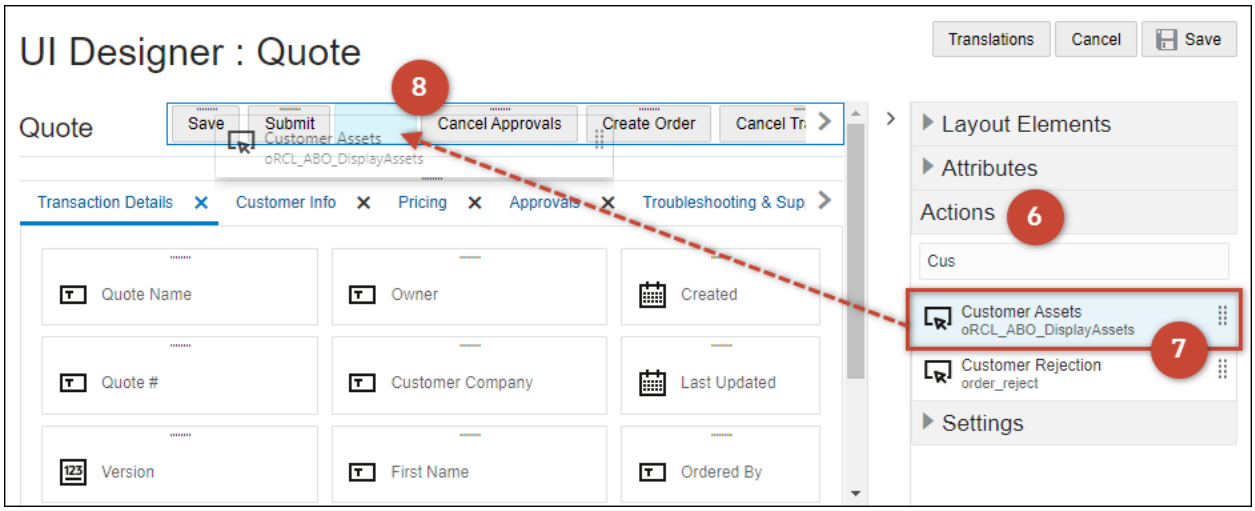
In this section, you will make changes to enable ABO.

B3.1: Add ABO Items to the Commerce Layout

Complete the following steps to add ABO attributes to the Line Item table and ABO actions to the Transaction UI.

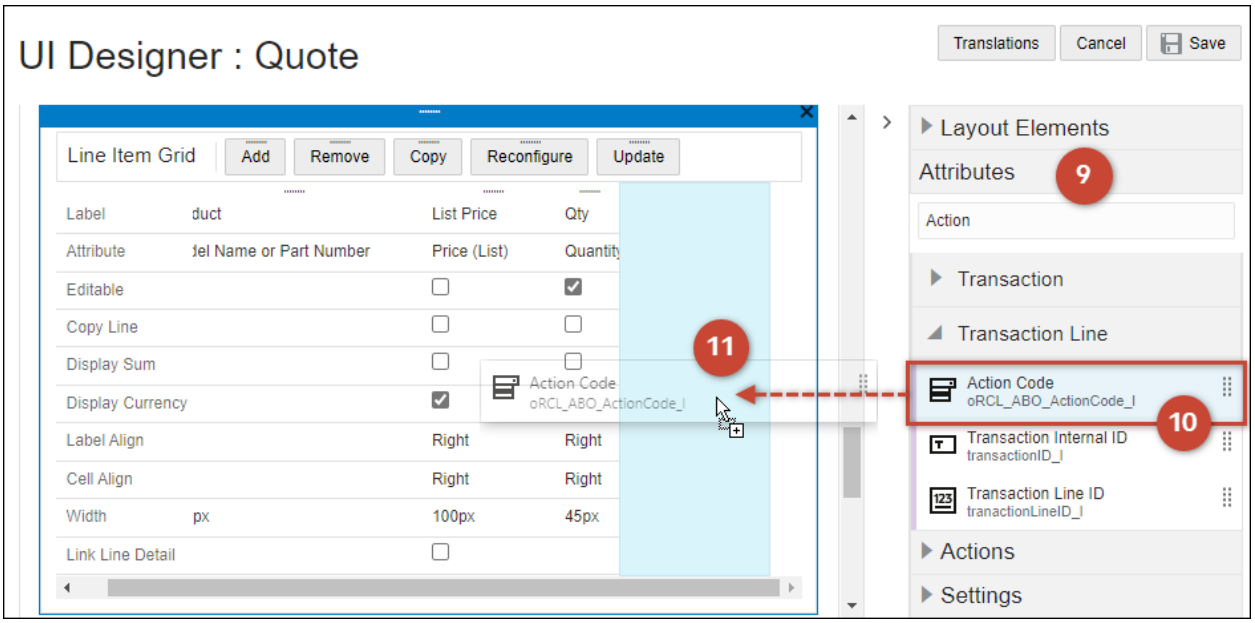
1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **Process Definition**, in the Commerce and Documents section. The Processes page opens.
3. Select **Documents** from the Navigation drop-down menu for the applicable process (e.g. Oracle Quote to Order).
4. Click **List.** The Document List page opens.
5. Select **JET Responsive Layout** from the Navigation menu, and then click **List**.

Add ABO Actions



6. Open the right side panel **Actions** object.
7. Select the **Customer Assets (oRCL_ABO_DisplayAssets)** action from the actions list.
8. Drag and drop the action into the toolbar at the top of the layout.

Add ABO Attributes to the Line Item Grid

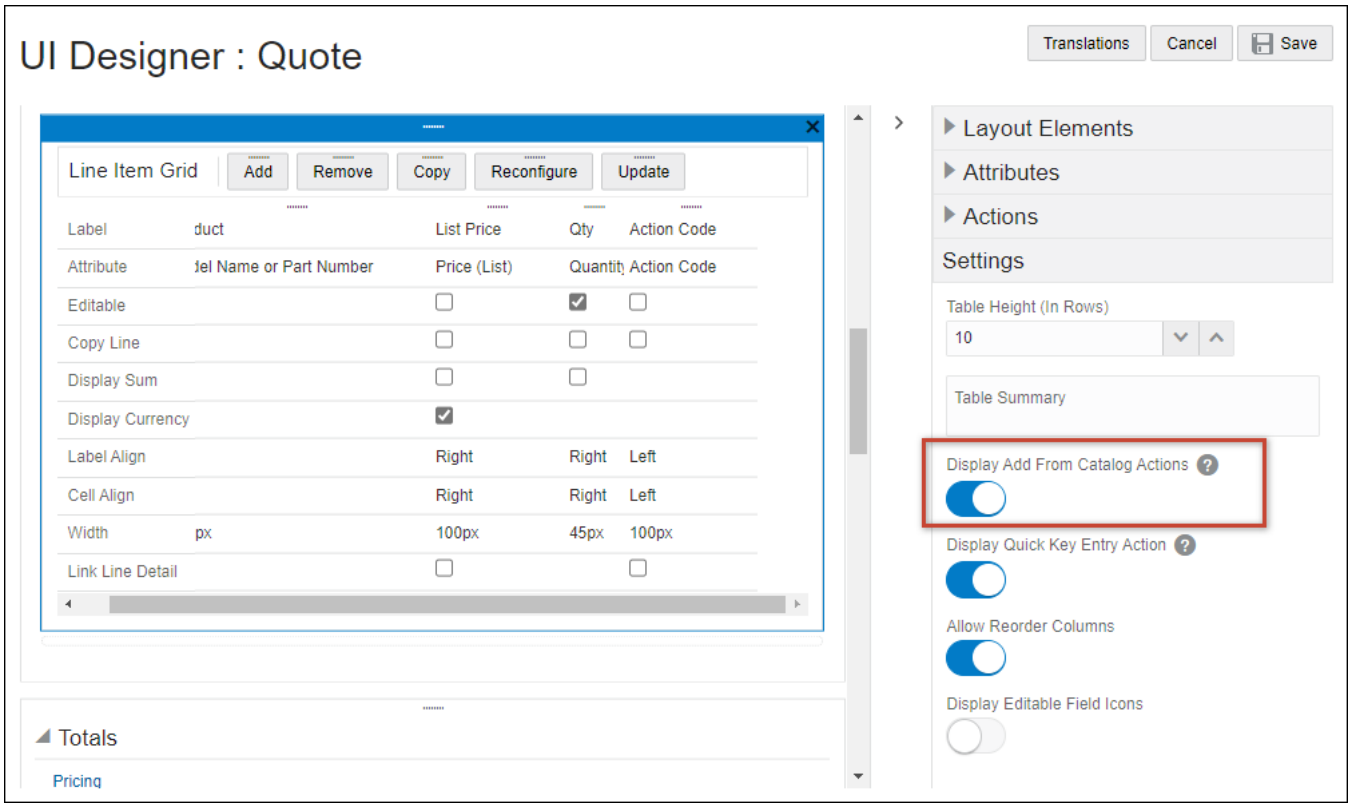


9. Open the right side panel **Attributes** object.
10. Select the appropriate sub-document (e.g. Transaction Line) attribute.
11. Drag and drop the attribute into the Line Item Grid element in the layout editor.
12. Repeat steps **Error! Reference source not found.** - **Error! Reference source not found.** to add the remaining attributes if it i s desirable for business need:

LABEL	ATTRIBUTE MAPPING
Action Code	Action Code (oRCL_ABO_ActionCode_I)
Request Data	Request Date (requestDate_I)
Fulfillment Status	Fulfillment Status (fulfillmentStatus_I)
Instance Name	Instance Name (itemInstanceName_I)
Component Attributes	Component Attributes (oRCL_ABO_Component Attributes_I)

13. If you want to allow users to change request date after line creation, select the **Editable** option for the Request Date option.

> **Note:** If desired, you can also add the following items for diagnosis or hide items for sales users using workflow steps: Instance ID (itemInstanceId_I), Line Item BOM ID (_line_bom_id), and Parent Line Item BOM ID (_line_bom_parent_id)



14. (Optional) Enable the Create Follow-On Order action:
    a. Click on the Line Item Grid object.
    b. Open the right side panel **Settings** object
    c. Enable the **Display Add From Catalog Actions** option.
15. Click **Save**.

B3.2: Add ABO Script to Main Document Reconfigure Action

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

Reconfigure is an existing action provided as a standard feature in CPQ Commerce. You must manually add BML script to the main document (e.g. Transaction) Reconfigure action to support the ABO implementation.

Perform this procedure for initial installation of an ABO implementation.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **Process Definition**, in the Commerce and Documents section. The Processes page opens.
3. Select **Documents** from the Navigation drop-down menu for the applicable process (e.g. Oracle Quote to Order).
4. Click **List.** The Document List page opens.
5. Select **Actions** from the Navigation drop-down menu for the main document (e.g. Transaction).
6. Click **List**. The Action List page opens.
7. Click the **Reconfigure** link in the Action Name column. The Action Admin page opens.
8. Click the Define Advanced Modify – Before Formulas radio button.
9. Click **Define Function**. The Select Attributes page opens.
10. Select the Library Function(s) tab.
11. Select the "`oRCL_abo_ReconfigureAction`" Commerce library function.
12. Click **Next**. The BML Editor opens.
13. Add the following script into the BML window, at the beginning of any existing code.

    > **Note:** If the current reconfiguration script is from a prior ABO package, remove the existing ABO logic and dependencies that are no longer needed

```
//////// BEGINNING of ABO SCRIPT//////////////////
result = commerce.oRCL_abo_ReconfigureAction(true);
////////////////////END OF ABO SCRIPT//////////////////////
```

14. Click **Save and Close**. The Admin Action page opens.
15. Click **Update**. The Action List page opens.

B3.3: Add ABO Script to Sub-Document Reconfigure Action

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

Reconfigure is an existing action provided as a standard feature in CPQ Commerce. You must manually add BML script to the sub-document (e.g. Transaction Line) Reconfigure action to support the ABO implementation.

Perform this procedure for initial installation of an ABO implementation.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **Process Definition**, in the Commerce and Documents section. The Processes page opens.
3. Select **Documents** from the Navigation drop-down menu for the applicable process (e.g. Oracle Quote to Order).
4. Click **List.** The Document List page opens.
5. Select **Actions** from the Navigation drop-down menu for the sub-document (e.g. Transaction Line).
6. Click **List**. The Action List page opens.
7. Click the **Reconfigure** link in the Action Name column. The Action Admin page opens.
8. Click the Define Advanced Modify – Before Formulas option.
9. Click **Define Function**. The Select Attributes page opens.
10. Select the Library Function(s) tab.
11. Select the "`oRCL_abo_ReconfigureAction`" Commerce library function.
12. Click **Next**. The BML Editor opens.
13. Paste the following script into the BML window at the beginning of any existing code.

    > **Note:** If the current reconfiguration script is from a prior ABO package, remove the existing ABO logic and dependencies that are no longer needed

```
//////// BEGINNING of ABO SCRIPT//////////////////
result = commerce.oRCL_abo_ReconfigureAction(false);
////////////////////END OF ABO SCRIPT//////////////////////
```

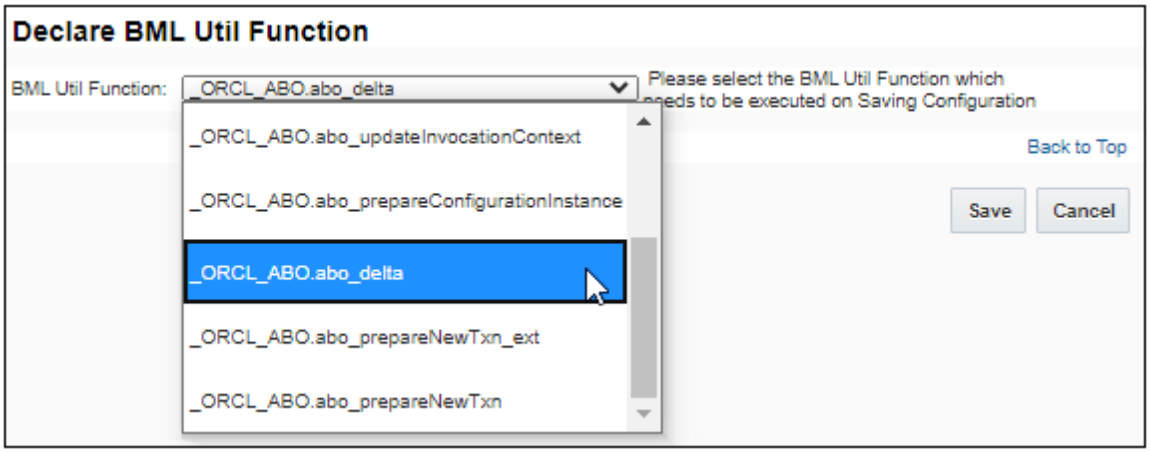
14. Click **Save and Close**. The Admin Action page opens.
15. Click **Update**.

B3.4: Define ABO Function to Execute Upon Saving a Configuration

After installing the ABO implementation package, you must define the ABO function to execute upon saving a configuration. The following procedure is a one-time procedure that you must perform after installing the ABO implementation package.

Complete the following steps:

1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **BOM** in the Products section, the BOM Administration Platform page opens.
3. Click **Declare Util Function** in the BOM Declaration section.
4. Select the **_ORCL_ABO.abo_delta** BML function.



5. Click **Save**.

B4: Advanced Implementation

You can customize ABO implementation using the information provided in this section.

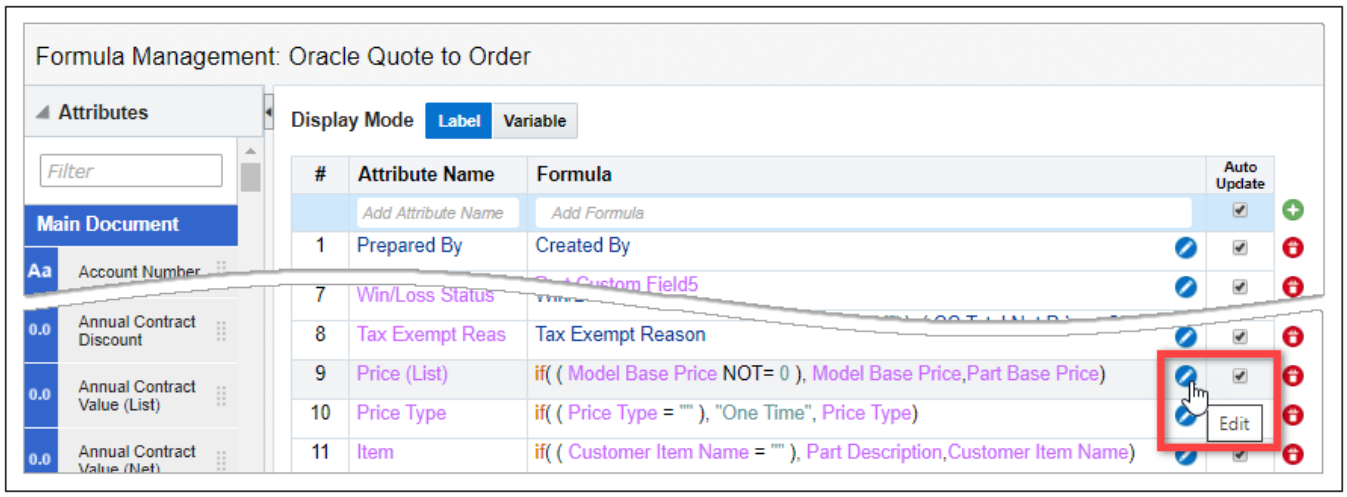
B4.1: Modify the Commerce Price List Formula

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

Complete the following steps to modify the Commerce process formula for the Line Item Price List attribute to support ABO.

> **IMPORTANT:** Complete the following steps if you did not import the Commerce process formula for the Line Item Price List attribute during migration. (I.e. Commerce > *Commerce Process* > Formula(s) > listPrice_l).

1. Log in to Oracle CPQ and open the Admin Home page.
2. Navigate to the Commerce **Processes** page.
   Commerce and Documents > Process Definition
3. Select **Formulas** for the applicable process (e.g. Oracle Order to Quote), and the click **List**.
4. Click the Edit icon ( ) for the **Price (List)** attribute.



5. Add the logic highlighted in red before your existing formula, which is highlighted in blue.

```
if(((oRCL_ABO_ActionCode_l = "DELETE") OR ((oRCL_ABO_ActionCode_l = "TERMINATE"))), 0,
if((_model_base_price NOT= 0), _model_base_price,_part_base_price))
```

> **Notes:**
>
> - The Price (List) formula shown above has been adjusted in 21D for delta pricing support. It no longer provides zero price for suspend, resume, or renew actions.
> - We recommend that the formula be reviewed and revised to consolidate with the latest Ref App formula as additional enhancements or modifications may have been made to the Ref App since the implementation package generation and this document publication.

6. Click **Save**.

B4.2: Add Data Columns for Asset-Based Ordering

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

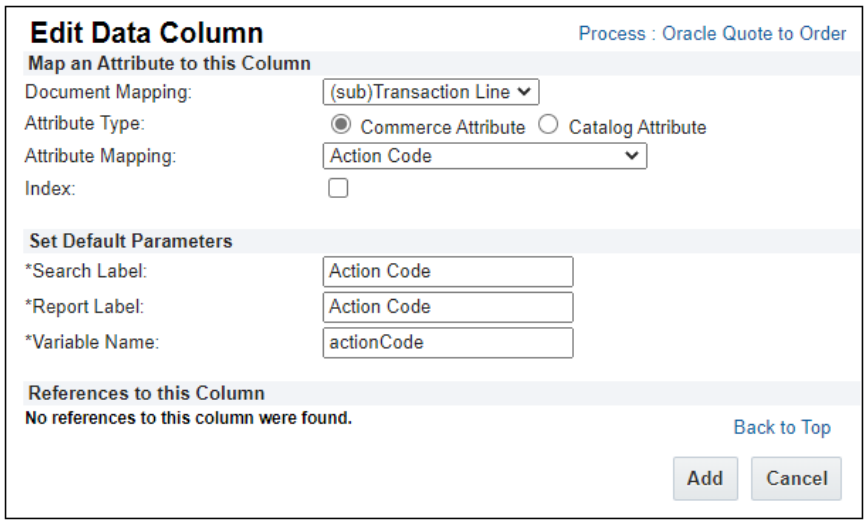
The Commerce attributes listed below require data columns for the ABO implementation. If the data columns already exist, you do not need to recreate them.

> **IMPORTANT:** If you did not install data columns as part of the ABO implementation package import, you must create the following data columns.

DOCUMENT MAPPING	ATTRIBUTE TYPE	ATTRIBUTE MAPPING	SEARCH LABEL	REPORT LABEL	VARIABLE NAME
Sub-Document	Commerce Attribute	Action Code	Action Code	Action Code	aBO_ActionCode
Sub-Document	Commerce Attribute	Request Date	Line Request Date	Line Request Date	requestDate_l
Sub-Document	Commerce Attribute	Instance ID	Item Instance Id	Item Instance Id	iteminstanceId_l
Sub-Document	Commerce Attribute	Fulfillment Status	Fulfill Status	Fulfill Status	fulfilmentStatus_l

Complete the following steps to add data columns for all of the data columns listed in the table above.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Navigate to the **Edit Data Column** page.
   Admin > Commerce and Documents > Process Definition > Data Columns
3. Click **Add**.
4. Select the sub-document (e.g. Transaction Line) from the **Document Mapping** drop-down.



5. Select the **Commerce Attribute** option Attribute Type.
6. Select the applicable attribute from the **Attribute Mapping** drop-down.
7. Select the **Index** option for the sub-document: Instance ID (itemInstanceId_l) data column.

   > **Note:** The other attributes do not require indexing.

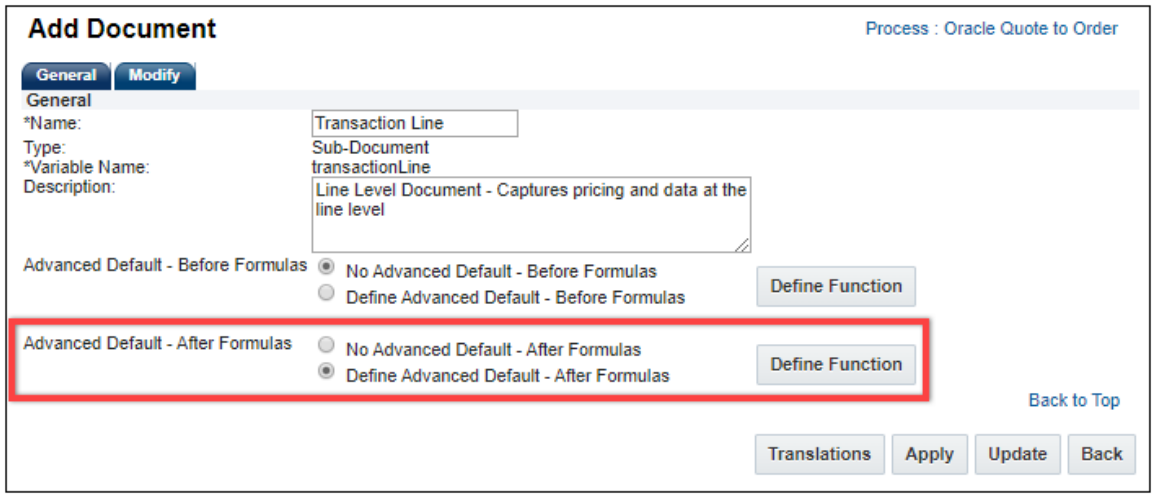
8. Enter the Search Label.
9. Enter the Report Label.
10. Enter the Variable Name.
11. Click **Add** or **Update**.

B4.3: Add Sub-Document Advanced Default Script

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

Complete the following steps to add the sub-document Advanced Default – After Formulas Rule to support Simple Products, Subscription Management, and Request Date Defaulting.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Navigate to the **Commerce Processes** page.
   Commerce and Documents > Process Definition
3. Select **Documents** for the applicable process (e.g. Oracle Order to Quote), and the click **List**.
4. Click on the sub-document Document Name (e.g. Transaction Line).
5. For the Advanced Default – After Formulas, select the **Define Advanced Default – After Formula** option, and then click **Define Function**.



6. Select the Library Function(s) tab.
7. Select the following Commerce library functions:
   - `oRCL_abo_BuildLineItemHierarchy`
   - `oRCL_abo_PostDefaultsOnLineItems`
8. Click **Next**. The BML Editor opens.
9. Add the following script into existing BML code.

```
//ABO transacrionLineAdvancedDefaultAfterFormula Script
//defined in "Process Definition"->"Oracle Quote to Order"->"Documents"->"TransactionLine"-
>"Advanced Default"->"After Formulas"
// sample block that allows ABO field update on new SP Line Items
////////////////////////////////////////
//Function Definition
//
//  Imported Commerce Functions:
//      "aboBuildLineItemHierarchy"
//      "aboPostDefaultsOnLineItems"
//
// Function Body:
returnVal = "";
lineHierInfo = commerce.oRCL_abo_BuildLineItemHierarchy();
returnVal = returnVal + commerce.oRCL_abo_PostDefaultsOnLineItems(lineHierInfo);

return returnVal;
```

10. Click **Save and Close**. The Admin Action page opens.
11. Click **Update**.

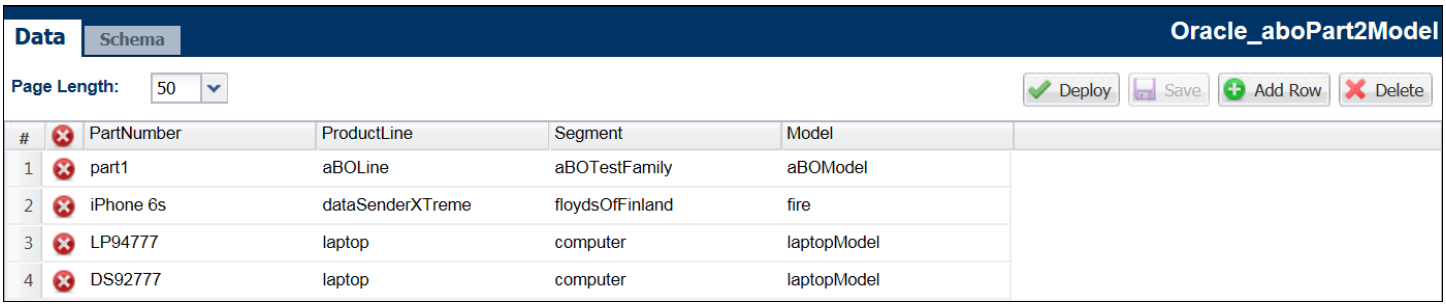
B4.4: Modify the ABO Data Table

> **Note:** Changes to the Oracle_aboPart2Model ABO Data Table are no longer required for 19B or later release ABO implementations. The ABO Data Table is only applicable if you store assets in an external asset repository or for assets created using ABO implementation package from 18C or earlier.

If you use the ABO Data Table, add all asset-related BOMs to the ABO Data Table (i.e. Oracle_aboPart2Model) included in the ABO implementation package. If there are multiple BOMs set up on your site, they can all be set up with ABO. You cannot use non-ABO BOMs for an ABO enabled process.

The ABO Data Table stores the reference table to lookup the model path (i.e. product family/product line/product model) for the part number, and is used to decide which model to launch when modifying an asset for a particular part.

As shown in the below figure, you can also add one BOM to model mapping entry from the root BOM part number to the default model in the Oracle_aboPart2Model data table.



#	PartNumber	ProductLine	Segment	Model
1	part1	aBOLine	aBOTestFamily	aBOModel
2	iPhone 6s	dataSenderXTreme	floydsOfFinland	fire
3	LP94777	laptop	computer	laptopModel
4	DS92777	laptop	computer	laptopModel

> **Notes:**
>
> - The Segment column refers to the Product Family.
> - This data table is used to define which model to use to launch configurator during the modify asset flow. For assets created by 19B and later implantation packages, the model information is captured from the asset itself.
> - If you use the ABO Data Table, add all asset-related BOMs to the ABO Data Table (i.e. Oracle_aboPart2Model) included in the ABO implementation package.
> - If there are multiple BOMs set up on your site, they can all be set up with ABO.
> - If ABO is setup on your site, all BOMs will enforce ABO logic in the ABO enabled process.

B4.5: Enable ABO for a Custom Commerce Process

ABO implementation packages are designed for use with the reference application image that includes a commerce process definition (oraclecpqo), a main document (transaction) and a sub document (transactionLine). Even though the ABO implementation packages are designed for use with the reference application image, but you can enable ABO for other commerce processes.

Complete the tasks to enable ABO for a custom commerce process.

- Ensure the following attributes are present in the target process with the same type and domain as in base reference application process. Refer to Appendix D1: Commerce Main Document Attributes and Appendix D2: Commerce Sub-Document Attributes.
- Import ABO Implementation Package and select your custom commerce process as the target to Cross Process Migration when importing the ABO Implementation Package.
- Register the customer Commerce Process in the ABO initialization module. The recommended way is to override the "`abo_loaddefaultContext`" is to populate an additional entry in the "`IntProcessInfo`" array in default context.

  The following example shows the 19B implementation package "`IntProcessInfo`" array.

```
"IntProcessInfo": [{
    "commerceProcessName": "oraclecpqo",
    "isExternal": "false",
    "main_Doc_Url": "$RESTURL/commerceDocumentsOraclecpqoTransaction",
    "sub_Doc_Url": "$RESTURL/commerceDocumentsOraclecpqoTransactionTransactionLine",
    "postNewSaveAction": "cleanSave_t",
    "subDocAttributes": {
      "oRCL_ABO_ActionCode_l": {
        "type": "menu"
      },
      "fulfillmentStatus_l": {
        "type": "menu"
      }
    }
  }
]
```

  - "`main_Doc_Url`" is in the format of "`$RESTURL/commerceDocuments{ProcessVarName}{MainDocVarName}`"
  - "`sub_Doc_Url`" is in the format of "`$RESTURL/commerceDocuments{ProcessVarName}{MainDocVarName}{subDocVarName}`"
    - `{ProcessVarName}`: The variable name of the Commerce process, the first letter must be capitalized.
    - `{MainDocVarName}`: The variable name of the main document, the first letter must be capitalized.
    - `{subDocVarName}`: The variable name for the sub-document
  - You can verify the URL in the interface catalog or you can replace the $RESTURL with actual site name.
  - "`postNewSaveAction`" is only used when you launch the Subscription Workbench directly without going thru display asset action (i.e. modifying an asset without a CPQ transaction). "`postNewSaveAction`" will be used as the modify action parameter for the new transaction REST API call, and should be a valid modify action on the main document.
  - The other items in the "`IntProcessInfo`" array do not require updates.

  Note: In 18C and earlier package implementations, this information is registered in the "`abo_initialize`" function.

1. Update the commerce process variable name reference in `abo_initialzeContext` call from the new process.Navigate to Admin > Developer Tools > Global Search on BML.
2. Search for the "`abo_initializeContext`" function and note all actions where the "`abo_initializeContext`" function is used.

3. For each action script found, change the "`abo_initializeContext()` call to pass the new process variable name. Typically you need to change the following actions:
   - `oRCL_ABO_CreateFollowOnOrder`
   - `oRCL_ABO_DisplayAssets`
   - Other common action scripts to check are custom actions for Update Asset and ABO Diagnosis.
4. For each Commerce library function, change the "`abo_initializeContext()`" call to pass the new process variable name. Typically you need to change the following library functions:
   - `oRCL_abo_PostDefaultsOnLineItems`
   - `oRCL_abo_ReconfigureAction`
5. For "`oRCL_ABO_DisplayAssets`" action script, change the `backUrl` open main document action variable name "`actionVarName=_open_transaction`" to the customer specific open main document action variable name.

```
backUrl = "/commerce/buyside/document.jsp?formaction=cancelAddFromCatalogCookie&bs_id=" +
bs_id + "&process=" + commerceProcess + "&actionVarName=_open_transaction";
```

6. Review process specific logic within the ABO package and identity if the logic should be duplicated or adjusted.
   - Code blocks like the following example from "`abo_convertDeltaBomtoAsset`" should be reviewed

```
//now it is process specific non-core fields ,
if(commProcName == "oraclecpqo"){  //we only implement for base refApp template
netAmount=jsonget(fieldJson, NET_AMOUNT, "string", "");
```

   - "`abo_updateAsset`" has references to "`abocontext.extraLineFieldForAsset`" which is defined in the defaultcontextjson.txt file, and used to pull data from sub-documents into the BOM structure and convert assets in the Update Asset sample. If applicable, update the `abo_updateAsset` function or `abo_loadDefaultContext` function.
   - The "defaultContextJson" also has the commerceAttributeInDelta property.

```
"deltaBomSvcSetting": {
   …
   "commerceAttributeInDelta" : ["contractStartDate_l", "contractEndDate_l"]
 },
```

     These sub-document attributes are used in delta processing. If the attributes have a different value, the action code of the line will be set to **Update**. In addition, a BOM attribute mapping rule has to be defined for these attributes for them to take effect. This is mainly used for the Subscription Management solution.

B5: Implement Additional Features

This section contains tasks and prerequisites for the following features for ABO:

- Business Time Zone Setting for ABO
- Copy Transaction and Transaction Lines for an Asset
- Delta Pricing for ABO
- Reconfigure Child Models from the Line Item Grid
- Subscription Ordering for Simple Products
- Suspend and Resume for Child Operations

B5.1: Business Time Zone Setting for ABO

In Oracle CPQ 21A, an updated 21A ABO package provides a new administrator-defined business-level time zone setting called `businessTimeZone`. The business time zone is used, rather than the server default time zone, as the basis for all asset-related date and full timestamp translations. This provides administrators the ability to set a time zone that is more relevant to their business.

The updated package interprets the commerce date fields consistently during ABO processing based on the specified business time zone setting. End users may need instruction on how to interpret their user interface display of start, end, and request date fields in relation to the overall business time zone setting.

Prerequisites:

- Oracle CPQ 21A or later
- The Oracle 21A ABO package is required to be installed or the specific feature logic adopted into your 19C or earlier package.

To apply the feature logic, override the "`abo_loadDefaultContext`" BML function and in the final code block change the "`businessTimeZone`" value from **GMT+0** to your desired time zone, as shown below:

```
//Set business timezone to be used in ABO flows to interpret dates and to override the timezone
used for interpretation
//Use GMT+n or GMT-n for absolute timezone
//Use Timezone name for DST Adjusted. For example - "America/Los_Angeles"
jsonput(abocontext, "businessTimeZone", "GMT+0");
```

Tips and Considerations When Using Business Time Zone

- The default "businessTimeZone" setting is GMT+0. This is the same as the default server time zone for newly provisioned Oracle sites.
- The "businessTimeZone" setting is used to interpret date fields in Oracle CPQ Commerce which do not include timestamp or time zone information. For example, an order line with the end date of 2030-10-01 will be interpreted as 2030-10-01 00:00:00.000 for the specified time zone when converting to an asset.
- Only an end user with the same time zone preference as the business time zone will see start and end date matching the business time zone on the asset user interface. End users in different time zones than the designated business time zone will see a different start and end date displayed for an asset based on that end user's time zone preference. End users need to be aware of how the date fields are displayed in relation to the overall company business time zone setting.
    - For sales users with the same user time zone as the "businessTimeZone" setting, the asset will have 2030-10-01 00:00:00 as the end date in the Subscription Workbench, which exactly matches the end date as specified in the original order line.
    - For sales users with a different user time zone as the "businessTimeZone" setting, the end date displayed will be converted to the user time zone and may not fall on the same date as the original order line. In this case sales users need to be instructed to interpret the date value in the Subscription Workbench based on the system business time zone.
    - The business time zone concept only applies to area related to ABO integration. In other integrations areas, the attribute value is a string that doesn't include timestamp/time zone information and it is up to the integration need to interpret the data correctly.

- If the business time zone is Daylight Saving Time(DST) enabled, the date value on asset data will be saved with a different hour in GMT+0 depending on whether that particular date is daylight saving enabled or not.
  For example, if the business time zone is "America/Los_Angeles" when querying asset data via RESTful which always returns in ISO format with GMT+0 as time zone:
  - For an order line with an end date of 2030-12-15, the asset end date in restful response is 2030-12-15T08:00:00.000Z.
  - For an order line with an end date of 2030-07-15, the asset end date in restful response is 2030-07-15T07:00:00.000Z.
  - Similarly when a sales user has DST-disabled time zone set, they will see different hours for winter and summer.
  - Similarly when the sales user is has DST-enabled time zone set, but the business time zone is not DST-enabled, the sales user will see different hours for winter and summer.
- Once set up, asset start and end dates are populated based on business time zone; therefore, changing the business time zone setting to a different time zone should be avoided. If a change to the business time zone is required, it must be carefully planned with detailed analysis regarding the impact to legacy data.

B5.2: Copy Transaction and Transaction Lines for an Asset

Oracle 21B provided customers the ability to easily copy ABO order lines and Transactions. Copying a Transaction creates a new Transaction from an existing Transaction in the Transaction Manager. The copied Transaction will contain the attributes as defined by the Copy action Initialization tab. The new Transaction will have newly assigned Instance IDs that are different from the source Transaction. The Line Items will have the Action Code of Add and the Transaction ID, User Name, and Creation Date are updated accordingly.

- **Copy Transaction** creates a new Transaction from an existing Transaction. The new Transaction's assetkey (`itemInstanceId_l`) and Action Code are assigned accordingly when the initialization for these attributes is set to 'Revert to Default'. The `rootAssetKey_l` is assigned based on the sub-document advanced default logic similar to the Subscription Management (SM)/Fusion Order Management (FOM) package and the initialization set to 'Revert to Default'.
- **Copy Transaction Lines** creates a new Line Item(s) by copying an existing Transaction Line or Lines. When Line Items are copied using any actions in Commerce, all the ABO-required data is copied appropriately and ABO functionality continues to work as expected. The Line Items are copied into the Line Item Grid and are given different instance IDs. The copied Line Items are available for the user to modify and/or save with the Transaction.

Enable Copy Transaction and Transaction Lines for an Asset

> **Note:** For new customers on Oracle CPQ 21B or later, the Enable ABO Field Adjustment for Copy feature is enabled by default and this option is not visible in the Commerce Options page.

Prerequisites:

- Oracle CPQ 21B or later
- Your site must have an ABO package installed with the assetkey and Action Codes defined within the Commerce process.

Complete the following steps to enable copy transaction and transaction lines for an asset.

1. Navigate to the **Commerce Settings** page.
   Home Page > Commerce and Documents > Commerce Settings
2. Click **Yes** for the **Enable ABO Field Adjustment for Copy** option.
3. Click **Update**.

Notes:

- When multiple modify lines of root configurations are copied into a Transaction, each of the newly copied root Line Items is assigned a unique assetkey (itemInstanceId_l) and considered a new and separate configuration from the source configuration.
- When copying Transactions and Transaction Lines the source's Action Code can determine if an item can be copied and the Action Code setting for the new item. Refer to the following guidelines:
    - Child Line Item with Action Code of Suspend, Resume, Renew, or Terminate is copied and the new Line Item Action Code is set to Add.
    - Child Line Item with Action Code of Delete is not copied.
    - Root Line with Action Code of Terminate copies all child Line Item(s), including child Line Items with Action Code of Delete, and sets the new Line Items Action Code to Add.
- When the ABO Copy Transaction and Transaction Lines feature is enabled, note the following regarding the assetkey (itemInstanceId_l), Action Code (oRCL_ABO_ActionCode_l), and rootAssetKey (rootAssetKey_l) attributes:
    - Their default values needs to be set to 'None' at the attribute default level.
    - Within the Copy action Initialization tab, 'Revert to Default' must be set for these attributes.
    - The 'Use Specified Value' and 'Copy From Original' options are ignored and interpreted as 'Revert To Default'.
    - The' Define function' option must be used carefully for these attributes.
- When copying a Transaction, the advanced default values for the Transaction Line attributes—with the exception of the assetkey and rootAssetKey attributes—will only take effect for the first Transaction Line copied. If advanced default values are used, you may need to implement a series of copy Transaction/copy Transaction Lines to achieve the desired result. This limitation is true for use cases outside of the ABO Copy Transaction feature.
- The Copy Transaction and Transaction Line action copies the current instance of the source configuration saved at the point in time the copy function is executed. Any updates within the associated lines to the source configuration, such as follow-on order lines, are not copied or able to be referenced in the new configuration. Therefore, reconfiguration of the newly copied configuration does not incorporate stacked or follow-on Line Item changes made to the source configuration.
- For ABO Implementation Package 18C or earlier, when the root Line Item Action Code is Suspend, Resume, Renew, or Terminate, the Copy Transaction and Copy Transaction Lines feature is only able to copy the root Line Item. This newly copied root Line Item is given a unique assetkey and the Action Code is set to Add. However, it is considered as a part line not a model line and cannot be reconfigured.
- This feature does not adjust every attribute as part of this copy Transaction/Transaction Line function. For example, the fulfillentstatus_l and requestDate_l attributes are not adjusted. Therefore, administrators need to review their attributes in relation to the copy function and as applicable use standard customization practices to make necessary modifications.
- The newly copied configuration is given a unique assetkey with the default format of abo+guid. If you have customized the assetkey format, this change needs to be incorporated as part of the copy action.
- When copied, Transaction Lines created via a non-modify action and not saved from the Configurator UI, the price value from the original source is applied to the newly copied Line Item. If the original Transaction Line has a price value of zero, the user needs to reconfigure the Transaction in order to recalculate the value.

B5.3: Delta Pricing for ABO

Delta pricing shows pricing differences between the current state of an asset and pending asset modifications. View the following sections to enable delta pricing within Oracle CPQ:

- Configuration Delta Pricing for ABO – This Oracle CPQ 21B feature displays delta price for list unit price within the configuration UI. The price does not reflect discounts defined in commerce.
- Commerce Delta Pricing for ABO – This Oracle CPQ 21D feature displays the difference between the prior and current price. The price reflects quantity, rollup price, and discounts

    Note: Commerce delta pricing can be implemented independently from Configuration delta pricing.
- Commerce Delta Pricing for Projected Assets – This Oracle CPQ 22B features displays delta price for pending fulfilled orders. Commerce Delta Pricing must be enabled to enable this feature.

Configuration Delta Pricing for ABO

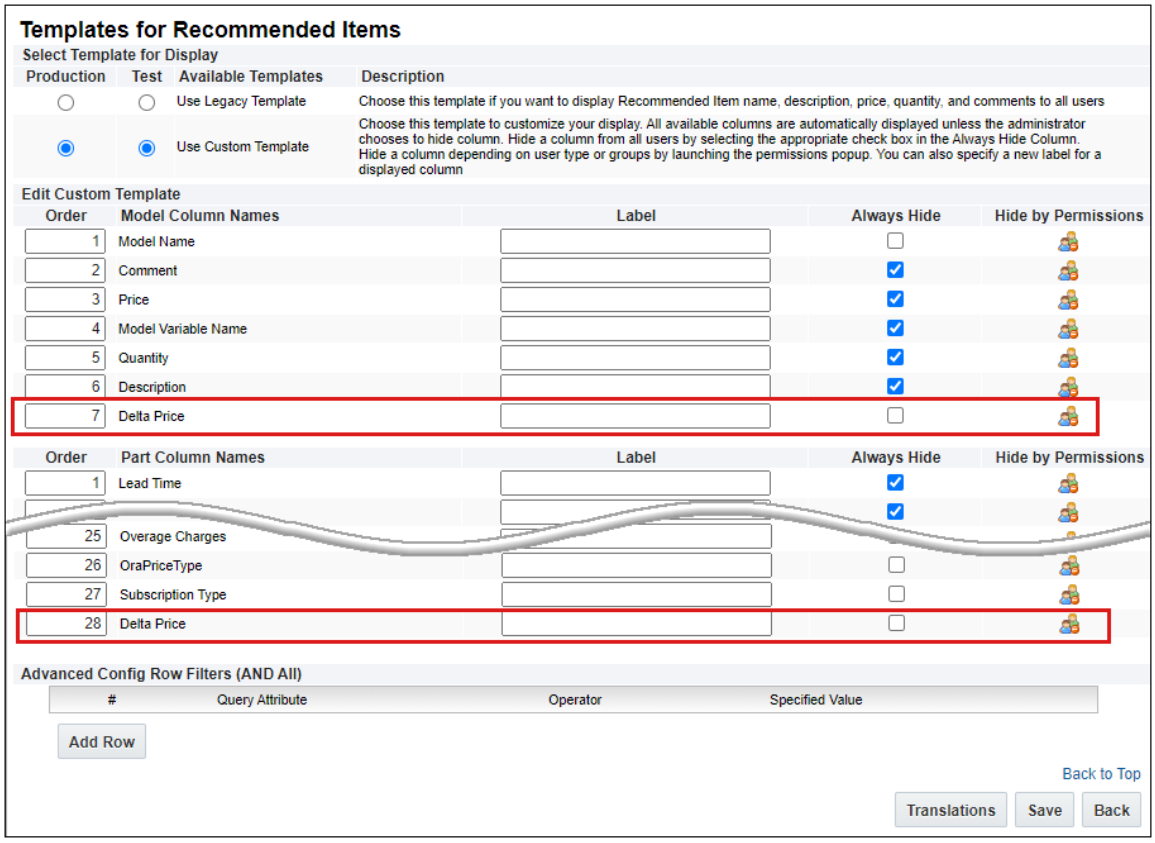
Oracle CPQ 21B enables delta pricing information inside the Configuration User Interface when implemented with ABO. The configuration delta pricing feature is supported only for BOM Items/BOM Mapping in Oracle CPQ.

Prerequisites:

- Oracle CPQ 21B or later
- Your site must have an ABO Implementation Package 18D or later installed

To enable configuration delta pricing, complete the following steps:

1. Navigate to the **Page Templates** page.
   Home Page > Style and Templates > Page Templates
2. Click **Templates for the Recommended Items Page**.



3. Unselect the **Always Hide** checkbox for the **Delta Price** in the **Model Column Names** group and **Part Column Names** group. This makes the Delta Price visible in the page layout.
4. (Optional) Enter a **Label** name for Delta Price.
5. (Optional) Click on the **Users** icon under **Hide by Permissions** column to designate visibility of **Delta Price** based on user type. Permissions are granted to all user types by default.
6. Click **Save**.

Notes:

- The delta price is the difference (+/-) of the current price of the item as compared to the existing price of the asset as currently saved to the configurator.
- Delta price will not be available for order lines in a new configuration. The delta price fields populate upon a Modify on an existing asset from the Subscription Workbench or Assets page. This feature is available for asset modifications and follow-on orders only.
- The delta price is displayed in the same currency as the model/parts in the configurator.
- The Copy Transaction and Transaction Line action copies the current instance of the source configuration saved at the point in time the copy function is executed. Any updates within the associated lines to the source configuration, such as follow-on order lines, are not copied or able to be referenced in the new configuration. Therefore, reconfiguration of the newly copied configuration does not incorporate stacked or follow-on Line Item changes made to the source configuration.
- If a new part is added to the quote that was not part of the existing configuration, the price of the added item is not included in the delta price. This new item displays 0.00 in the part delta pricing column. However, the added part is included in the overall Total Price.
- Without delta pricing enabled, when configuring a system the Total Price of BOM on the root level remains the same—regardless if child models were configured or not. With delta pricing enabled, the Total Price of the entire system updates on the root level if the child models are configured. For more information, refer to Oracle CPQ Administration Online Help > Asset-Based Ordering Implementations – 21B Feature Enhancements.

Commerce Delta Pricing for ABO

Oracle CPQ 21D introduces a new sub-document Delta Pricing Attribute Set and a new Commerce library function to extend the delta pricing to Oracle CPQ Commerce transactions for ABO sites.

Prerequisites:

- Oracle CPQ 21D or later

To enable this feature administrators need perform the following tasks:

- Create a Sub-Document Delta Pricing Attribute Set
- Create Calculate Delta and Price Rollup Function
- Add Calculate Delta Price and Rollup Function to Commerce
- Add Delta Price Attributes to the Transaction UI

Create a Sub-Document Delta Pricing Attribute Set

Note: This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

Perform the following steps to create a sub-document Delta Price Attribute Set.

1. Navigate to the Admin Home Page.
2. Click **Process Definition** in the Commerce and Documents section.
3. Select **Documents** from the applicable commerce process Navigation drop-down, and then click **List**.
4. Select **Attributes** from the sub-document Navigation drop-down, and then click **List**.
5. Click **Add** at the bottom of the Attribute List page.
6. Select **Delta Pricing Set** from the Attribute Type drop-down.
7. Click **Add**. The Delta Pricing Set attributes are displayed: Delta Price, Prior Price, and Rollup Delta Price.

Create Calculate Delta and Price Rollup Function

Note: Changes in this section are no longer relevant to Oracle CPQ 23B or later release while using the new commerce standard process definition and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the new commerce standard process definition.

1. Navigate to the Admin Home Page.
2. Click **Process Definition** in the Commerce and Documents section.
3. Select **Documents** from the applicable commerce process Navigation drop-down, and then click **List**.
4. Select **Library Functions** from the main document Navigation drop-down, and then click **List**.
5. Click **Add** at the bottom of the Commerce BML Library Functions List page.
6. Enter the following properties:
    a. **Name:** Calculate Delta and Rollup Price
    b. **Variable Name**: `calculateDeltaAndRollUpPrice`
    c. **Description**: This function calculates Delta Price values and the Rollup Delta Price value for the root element.
    d. Select **String** from the Return Type drop-down menu.
7. Add the following sub-document attributes:
    - _delta_price
    - _document_number
    - _line_bom_id
    - _line_bom_parent_id
    - _parent_doc_number
    - _prior_price
    - _rollupdelta_price
    - netAmount_l
    - oRCL_ABO_ActionCode_l
8. Add the calculateDeltaandRollupPrice script.

    > **Note**: This script uses the default sub-document "`transactionLine`" variable name. You must update this variable name if you are not using the default sub-document variable name.

9. Click **Add**.

Calculate Delta and Rollup Price

```
//-------------------------------------------------------------
// Commerce Library Function: calculateDeltaAndRollUpPrice
//
// Sets line item delta prices and rolllup delta prices when needed
//-------------------------------------------------------------

str = "";

// Map from document number to parent document number
parentDocNumberMap = dict("string");

// Map from document numbers to rollup delta prices (for root line items only)
rollupPriceMap = dict("float");

// Max depth of line item hierarchical tree.
// Adjust if 50 depths are still not enough for your site.
maxDepth = 50;
its = range(integer(maxDepth));

//
// print delta prices and calculate rollups if changed
//
// Performance will be better if incorporated into existing loops.
// Your site needs implicit string builder turned on.
//
for line in transactionLine {
    if ((line._parent_doc_number == "" AND line.oRCL_ABO_ActionCode_l <> "ADD" AND
line.oRCL_ABO_ActionCode_l <> "")
        OR containskey(parentDocNumberMap, line._parent_doc_number)) {
        put(parentDocNumberMap, line._document_number, line._parent_doc_number);
        if (line._parent_doc_number == "") {
            put(rollupPriceMap, line._document_number, line._rollup_delta_price);
        }
        deltaPrice = line.netAmount_l - line._prior_price;
        deltaChange = deltaPrice - line._delta_price;
        //Print delta price as 0 for blank values.
        if (fabs(deltaChange) > 0.00001 or fabs(line._delta_price) < 0.00001) {
            str = str + line._document_number + "~_delta_price~" + string(deltaPrice) + "|";
        }
        // roll up the delta price change when needed
        if (fabs(deltaChange) > 0.00001) {
            // find the root documenet number by going up the line item hierarchy
            docNumber = line._document_number;
            for it in its {
                parentDocNumber = get(parentDocNumberMap, docNumber);
                if (parentDocNumber == "" or isnull(parentDocNumber)) {
                    break;
                }
                docNumber= parentDocNumber;
            }
            // add the change to the root item
            if (containskey(rollupPriceMap, docNumber)) {
                price = deltaChange + get(rollupPriceMap, docNumber);
                put(rollupPriceMap, docNumber, price);
            }
```

```
        }
    }// end if (fabs(deltaChange) > 0.00001)

}

//
// print rollup prices for all root items.
//
docNumbers = keys(rollupPriceMap);
for docNumber in docNumbers {
    price = get(rollupPriceMap, docNumber);
    str = str + docNumber + "~_rollup_delta_price~" + string(price) + "|";
}

return str;
```

Add Calculate Delta Price and Rollup Function to Commerce Actions

Note: Changes in this section are no longer relevant to Oracle CPQ 23B or later release while using the new commerce standard process definition and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the new commerce standard process definition.

Complete the following steps to add the calculate delta price and rollup function commerce modify actions configured to run the pricing formula action. (E.g. Save, Reconfigure Inbound, etc.).

1. Navigate to the Admin Home Page.
2. Click **Process Definition** in the Commerce and Documents section.
3. Select **Documents** from the applicable commerce process Navigation drop-down, and then click **List**.
4. Select **Actions** from the main document Navigation drop-down, and then click **List**.
5. Click on the applicable commerce action (e.g. Save, Reconfigure Inbound, etc.).
6. Select the Advanced Modify - After Formulas Define **Advanced Modify - After Formulas** option, and then click **Define Function**.
7. Select the **Library Functions** tab.
8. Select the "`calculateDeltaAndRollUpPrice`" function.
9. Click **Next**, at the bottom of the page.
10. Add the following BML function:

```
str1 = "";
str2 = str1 + commerce.calculateDeltaAndRollUpPrice();
return str2;
```

> **Notes:**
> - Implicit string builder must be enabled for this function.
> - If this action already has a defined function, this function should be incorporated into the existing loop.

11. Click **Save and Close**.
12. Click **Update**.
13. Repeat Step 5 - Step 12 for all rollup function commerce modify actions configured to run the pricing formula action. (e.g. Save, Reconfigure Inbound, etc.).

Add Delta Price Attributes to the Transaction UI

Refer to Oracle CPQ Administration Online Help > Commerce Layout Editor - Line Item Grid Attributes for instructions.

> **Notes**
>
> - The delta pricing feature does not require a specific ABO Implementation Package upgrade. However, to implement this feature, the site is required to be on ABO Implementation Package 18D or later.
> - The delta price is the difference (+/-) of the current price of the item as compared to the existing price of the saved asset.
> - Delta price will not be available for order lines in a new configuration. The delta price fields populate upon a Modify on an existing asset from the Subscription Workbench or Assets page. This feature is available for asset modifications and follow-on orders only.
> - The delta price is displayed in the same currency as the model/parts in the Transaction UI.
> - Without delta pricing enabled, when configuring a system the Total Price of BOM on the root level remains the same—regardless of if child models were configured or not. With delta pricing enabled, the Total Price of the entire system updates on the root level if the child models are configured.

Commerce Delta Pricing for Projected Assets

Oracle CPQ 22B extends ABO Commerce Delta Pricing to include projected assets. Therefore, Commerce Delta Pricing must be enabled to provide delta pricing for projected assets. The reference application sub-document "`netAmount_l`" attribute value from the prior order is used to populate the prior price value to enable the delta price calculation for open orders.

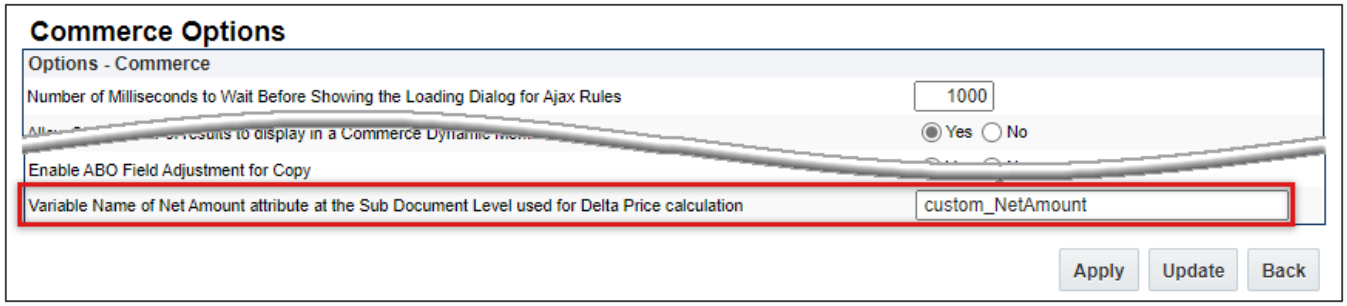
Prerequisites:

- Oracle CPQ 22B or later
- If your site uses the default sub-document "`netAmount_l`" attribute and Commerce Delta Pricing for ABO is enabled, you don't need to do anything to enable Delta Pricing for Projected Assets,
  - If Commerce Delta Pricing for ABO is not enabled, refer to Enable Commerce Delta Pricing for ABO.
  - If your site uses a custom sub-document Net Amount attribute, refer to Specify Sub-Document Net Amount Attribute for Delta Price.

Specify Sub-Document Net Amount Attribute for Delta Price

If your site does not use the reference application sub-document "`netAmount_l`" attribute, perform the following steps to specify a sub-document net amount attribute to enable the delta price calculation.

1. Navigate to the **Commerce Options** page.
   Home Page > Commerce and Documents > Commerce Settings
2. Specify the custom sub-document net amount variable name in the **Name of Net Amount attribute at the Sub-Document Level used for Delta Price** field.



3. Click **Apply** or **Update**.

> **Note:** If this option is not specified and the sub-document Net Amount attribute does not exist, Delta Price will not be calculated for open orders.

B5.4: Reconfigure Child Models from the Line Item Grid

> **Note:** This procedure is not required when using the new Standard Process and the Oracle CPQ 23B or later release specific ABO implementation package, since these changes are already included in the Standard Process definition.

For Oracle 21A and later, customers can directly reconfigure child system models from the Line Item Grid in the Transaction UI. To enable this feature adopt main and sub-document Reconfigure action changes, refer to:

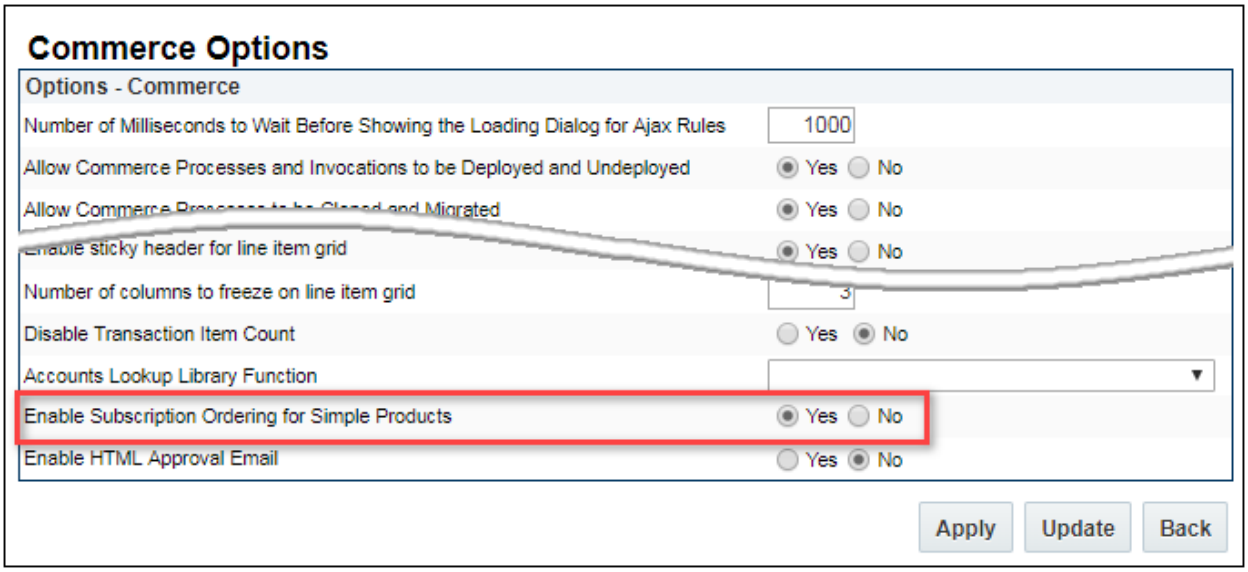
- Add ABO Script to Main Document Reconfigure Action
- Add ABO Script to Sub-Document Reconfigure Action

B5.5: Subscription Ordering for Simple Products

Enable Subscription Ordering for Simple Products to support directly adding simple products to a Commerce transaction for an asset-based order. A simple product is a product that does not have its part number associated with any of the related configuration models. When enabled, users can use Quick Add or part search to add simple products without navigating away from the transaction page.

Complete the following steps to enable the Subscription Ordering for Simple Products.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Navigate to the **Commerce Processes** page.
   Commerce and Documents > Commerce Settings
3. Set the Enable Subscription Ordering for Simple Products option to **Yes**.



4. Click **Update**.

> **IMPORTANT:** If you did not import the sub-document (e.g. Transaction Line) Advanced Default – After Formula, you must perform the following procedure: Manually Add Sub-Document Advanced Default Script.

B5.6: Suspend and Resume for Child Operations

The following section provides a list of typical setup information for Suspend and Resume on child and grandchild models and parts.

1. Create two Configuration attributes for each model or part you are enabling the Suspend and Resume operations.

   > **Note:** If you are enabling this enhancement for all models, you can create the Configuration attributes at the Product Family level and add them to the Configuration flow layout at the child and grandchild level.

   a. Create a Configuration attribute to capture the user intent to suspend or resume.

      - **Variable Name:** oRCL_ABO_Action
      - **Data Type:** Text
      - **Display Type:** Single Select Menu
      - **Menu Values:** SUSPEND and RESUME

   b. Create a Configuration attribute to capture the projected state of the asset that is calculated and stored in the attribute at runtime. This can be created as a hidden attribute.

      - **Variable Name:** oRCL_ABO_PriorAssetState
      - **Data Type:** Text or Single Select Menu
      - If you are defining a Single Select Menu, then the menu values should be: ACTIVE and SUSPENDED

   > **Note:** The variable name of the Configuration attribute is not significant, but it must match the BOM Mapping rules created later.

2. Add the **oRCL_ABO_Action** and **oRCL_ABO_PriorAssetState** attributes to the child/grandchild levels of the Configuration UI layout, so that Suspend/Resume actions are available on child models to users.

3. Create constraint rule(s) to enable or disable the Suspend and Resume actions in the Model Configuration page session depending on the value of the projected state of the asset.

   > **Note:** If you are enabling this enhancement for all models, you can create the rule at the Product Family level.

   Sample script:

```
if (oRCL_ABO_PriorAssetState == "SUSPENDED") {return "RESUME|^|";}
if (oRCL_ABO_PriorAssetState == "ACTIVE") {
  return "SUSPEND|^|";
}
if (oRCL_ABO_PriorAssetState == "") {
  return "SUSPEND|^|";
}
return "SUSPEND|^|RESUME|^|";
```

4. Navigate to Admin > Data Tables, for each model or part you are enabling the Suspend and Resume operations., create relevant BOM Data Table mappings in the Oracle_BomAttrMap Data Table between the Commerce line attribute `oRCL_ABO_ActionCode_l` and the Configuration attribute for capturing user intent to Suspend/Resume, that is `oRCL_ABO_Action` in the above example.

5. Optionally, create relevant BOM Data Table mappings in Oracle_BomAttrMap between BOM attribute `oRCL_ABO_PriorAssetState` and the configuration attribute created to capture the projected state of the asset that is `oRCL_ABO_PriorAssetState` in the above example.

C: BEST PRACTICES

This section contains best practices that may help to troubleshoot issues with the ABO implementation.

Oracle recommends the following best practices:

- Examine all available logs.
- Capture the REST API payload and response with the browser debugger, and use a service tester (e.g. REST Client) to make a similar call to diagnose particular step within the flow.
- If you are performing your own error checking in your script using the `throwerror()` function, the `throwerror` function has an optional parameter to indicate whether it is a user error or a system error, the default is user error.
    - For user errors, the error message will show up on the Subscription Workbench when a script operation fails.
    - The `throwerror` user error type is also useful to display a more meaningful message to the end user when an unexpected data encountered.
    - For system errors, a generic error will be displayed on the Subscription Workbench, and the detailed error will show up in log.
- Avoid using a print statement with a complex parameter in your BML script.

  For example: `print "my prefix"+ jsontostr(bigjson)`

  Print statements will always execute, regardless of the BML print logging setting. However, it may not always be possible to see the output. Since executing the `jsontostr()` function is a potentially resource intensive operation, Oracle recommends commenting out or deleting the print statements.
- Create a standard test function to verify the specific BML customization. Since most abo utility functions expect that the "`abo_initialize`" call has already been invoked, you may need to add the "`abo_initialize`" call to the beginning of the test function.

C1: Enable BML Print Logging

Examine all available logs and print the output of BML print statements to the log file for easier debugging of errors related to functions referenced in ABO scripts.

To enable this feature, navigate to **Admin > General Site Options** and set the **Enable BML print logging** option to **Yes**.

Basic details about the BML Print Logging enhancement:

- Oracle CPQ **General Site Options** contain an "**Enable BML print logging**" option. Use this option to enable or disable the logging of print statements in BML. The option is by default disabled.
- Log BML print values even when a user is running the script from the debugger mode in the BML Editor.
- BML Print is not recommended for diagnosis of BOM-related or ABO Process functionality, use the ABO Diagnosis Framework outlined below instead for such complex cases.

C2: ABO Diagnosis Framework

Use the ABO Diagnosis Framework to diagnose ABO BML scripting issues. The "`abo_addDiagnosticInfo`" library is included in the ABO implementation package to provide enhanced capabilities unavailable in the BML Print Log. The BML print statements have limitations that are more prominent during complex implementations such as ABO.

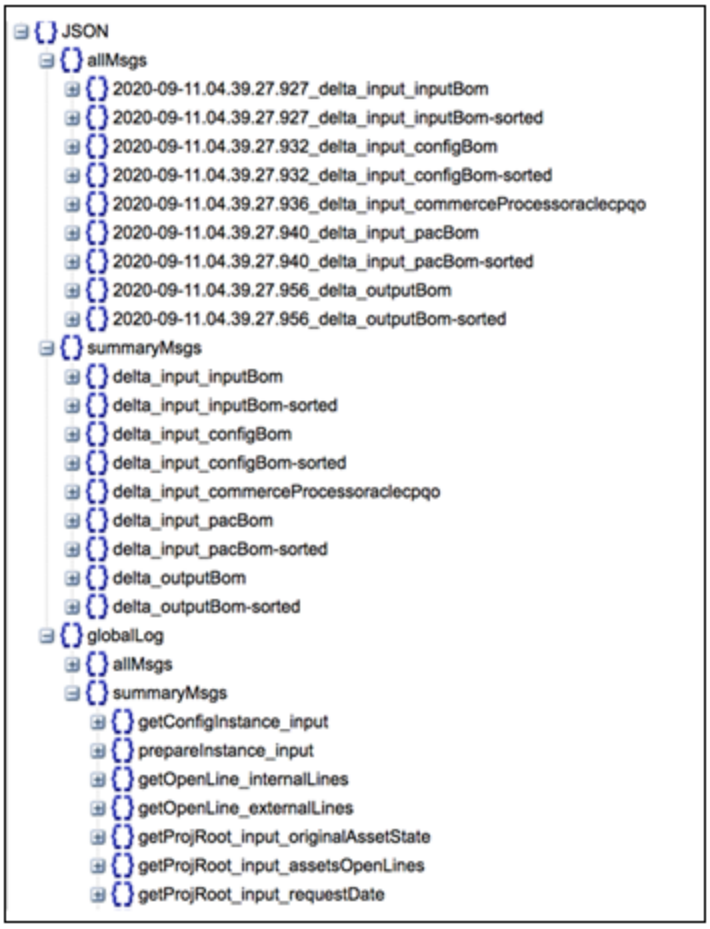
- When the **Enable BML print logging** option is set to **Yes**, the function is available on all user sessions. As a result, the performance of all users is impacted.
- All outputs are combined together and placed in the same log file. As a result, determining the sequence of events leading to a particular issue is a challenge.
- In addition, the "Print" function contains a limit of 4000 characters for BOM JSON and is often truncated.

The ABO Diagnosis Framework captures the log entry as a JSON structure. The key functions in an ABO package include conditional blocks to invoke "`abo_addDiagnosticInfo`" to capture input, output, and local variables at critical steps for key functions. Such calls add an entry into the diagnosis log JSON with a timestamp prefix, and allow examination of key variable values at critical ABO processing steps.

Beginning in Oracle CPQ 20A, a size restriction is placed on the underlying storage for the `usersessionset` API that is leveraged by the ABO Diagnosis Framework. The user will see the "`USERSESSIONSET value parameter exceeds the maximum length; contact your system administrator`" error when the ABO Diagnosis Framework is enabled. To work around this limitation, update your `abo_addDiagnosisInfo` function with the text provided in the ABO_Final_BML_Actions_21D.zip bugFixes/30662496_diagnosisLog_userSessionLimit folder and setup an external service for backup storage of the diagnosis log in integration center.

The ABO diagnosis leverages both user session (`userssionset` API) and global (`globaldict` API) caches, and frequently clears the diagnosis entries in the user session cache and appends those entries to the global cache. The entries from the user session cache are placed under "`allMsg`" and "`summaryMsg`" properties. The `allMsg` entries have a timestamp prefix, while the `summaryMsg` entries don't have a timestamp prefix, therefore for the same type of log entry only the last entry will be kept.

The entries from the global cache are placed under two similar entries under the "`globalLog`" property. When reviewing the overall picture, you should look at the union between them.



With the updated `abo_addDiagnosisInfo` function, included in the ABO_Final_BML_Actions_21D.zip bug fixes, the user session log will be periodically truncated and appended to the global session log. Additionally, when backup storage is setup in integration center, the entries in the `globalMsg` will periodically be copied to the backup storage. The log entries in the backup storage will be placed under two similar properties under the "`globalLogBackup`" property.

Notes:

- When the user operation is successful, the `globalLog` and `globalLogBackup` will have same content.
- When the user operation fails, the `globalLog` will have truncated content because all log entries since the last database will be lost. The `globalLogBackup` will contain the lost rolled-back log entries if the diagnosis log was collected immediately after the failed operation.
- Note if the log isn't collected shortly after a failed function and the user performs other operations, the entries in `globalLogBackup` will be overwritten by the subsequent successful operations. So it is important to collect the log immediately after failed operations.

C2.1: ABO Diagnosis Use and Best Practices

ABO Diagnosis is enabled, disabled, and printed by passing different parameters to the "`abo_addDiagnosisInfo`" function. It is assumed that the "`abo_inititalization`" function has been invoked. On the test environment, it is usually convenient to create a commerce test attribute and individual actions to enable, disable, and print the diagnosis information to this test attribute to display to the end user. A better solution, which minimizes extra items in the commerce definition, is to create a test utility function. Use the following sample to create the function and use the debugger to run it with different modes to enable, disable, or print.

```
if (mode == 1) {
  context = util._ORCL_ABO.abo_initializeContext("oraclecpqo");
  dummy = util._ORCL_ABO.abo_addDiagnosticInfo("", json(), "enable");
}
if (mode == 2) {
  dummy = util._ORCL_ABO.abo_addDiagnosticInfo("", json(), "get");
  print dummy;
}
if (mode == 3) {
  dummy = util._ORCL_ABO.abo_addDiagnosticInfo("", json(), "disable");
}
if (mode == 0) {
  dummy = util._ORCL_ABO.abo_getContext();
  print dummy;
}
return "";
```

The following steps outline a typical diagnosis process:

1. Setup the required data and isolate the reproduction flow and key steps to diagnose.
2. Initialize ABO context directly or indirectly by invoking an action such as Display Asset.
3. Enable diagnosis.
4. Go thru the key flow or step.
5. Print diagnosis and copy the diagnosis JSON.
6. Disable diagnosis as soon as possible
7. Use a JSON viewer to examine the diagnosis JSON.

Notes:

- For 19B or earlier ABO implementations, initialize context in debugger will fail, since the "`supplier_company_name`" is empty in debugger. A workaround for these releases is to hardcode the "`supplier_company_name`" in "`abo_loadDefaultContext`" or use display asset to initialize context.
- The diagnosis log is placed in a user session cache, global session cache, remote backup storage, and consumes a significant amount of memory resources and will slow down the application. Even though the diagnosis only affects the current user, the UI will become extremely slower as the JSON data accumulates and grows in size and adds new log entries to the diagnosis JSON. To minimize slow performance, enable the diagnosis log before the step you are testing, print and copy the diagnosis log for offline analysis, and then immediately disable diagnosis.
- When enabled or disabled, the diagnosis log is truncated. You can repeatedly call enable-log, run your test, and then print the log to collect the log for each step in a complex scenario.

C2.2: ABO Diagnosis Limitations

- Only one session should have diagnosis enabled since the diagnosis is captured based on user login. If the same login is used from different sessions with diagnosis enabled, the log entries will be mixed together.
- The diagnosis typically works with operations started on the CPQ UI.
- For modify REST services initiated from the CPQ UI, the "`_debug`" prefix is added to the "`sourceIdentifier`", this will trigger the diagnosis for the rest session. Refer to the script in the `displayAsset` action.
- The same "`_debug`" prefix approach can be used for diagnosis of a direct REST API calls from a service tester (e.g. REST Client).
- If the `abo_addDiagnosis` function is not updated using the script from the ABO_Final_BML_Actions_21D.zip, using the ABO diagnosis log will result in a USERSESSIONSET error.
- For SOAP web services or diagnosis of complex scenarios, the diagnosis log can be enabled temporarily, by including the following entry in "`abo_loadDefaultContext`" function:

```
if (_user_login == 'testUser') {
   jsonput(abocontext, "AboDiagnosticDisabled", false);
}
```

- Beginning in Oracle CPQ 20A and later, the following error will occur when using the diagnosis log::

  "`USERSESSIONSET value parameter exceeds the maximum length; contact your system administrator.`"

  To bypass this issue, refer to C2.3 Override abo_adddiagnosticinfo USERSESSIONSET Error. You can resolve this issue by implementing the `abo_addDiagnosis` function included in the ABO_Final_BML_Actions_21D.zip and setting up an external service as backup log storage in integration center.

IMPORTANT:

- The changes to the "`abo_loadDefaultContext`" function should be removed as soon as the log is collected.
- You should always add a condition to restrict when the log is enabled (such as to check for user login) to avoid impact to the overall system performance and other user sessions.
- **You should NEVER blindly enable the log in** `abo_loadDefaultContext`.
- **You should NEVER enable the log by toggling the AboDiagnosticDisabled flag inside the defaultContextJson.txt**
- Also, note that the `abocontext` is cached for the current user session, so if this approach is used for UI diagnosis, the user needs to logout and login to pick up the `abo_loadDefaultContext` change.

C2.3: Override abo_addDiagnosticinfo USERSESSIONSET Error

Beginning in Update 20A, CPQ enforces a limit on the `usersessionset` BML function. This affects the ABO diagnosis functionality, since the diagnosis data tends to be very large. Perform the following tasks to continue leveraging the ABO diagnosis functionality:

- Override the abo_addDiagnosticInfo function to store the diagnosis information in the global cache instead of the `usersessioncache`.
- Set up a Generic Integration to an external backup storage service to periodically back up the global cache entries to an external source since the global cache will be overwritten by the subsequent successful operations.

Override the abo_addDiagnosticInfo Function

Note: Download the latest ABO_Final_BML_Actions_21D.zip file from My Oracle Support (Doc ID 2182966.1).

1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **BML Library** in the Developer Tools section.
3. Perform one of the following options for the `abo_addDiagnosticInfo` function:
   a. If the function hasn't been previously overridden, click on the **Create** link under the Override Column.
   b. If the function has already been overridden, click on **Edit**.
4. Copy the content from the abo_addDiagnosticInfo_workaround_for_UserSessionSizeLimit.txt file in the ABO_Final_BML_Actions_21D.zip bugFixes/30662496_diagnosisLog_userSessionLimit folder into the `abo_addDiagnosticInfo` function.
5. If `abo_addDiagnosticInfo` function has already been overridden, you need to compare the existing function with the changes in the ABO_Final_BML_Actions_21D.zip bugFixes/30662496_diagnosisLog_userSessionLimit folder and merge the changes into the existing function.
6. Click **Update**.
7. Deploy the Commerce process to implement your updates.

Set Up a Generic Integration to an External Backup Storage Service

1. Log in to Oracle CPQ and open the Admin Home page
2. Click **Integration Center** in the Integration Platform section.
3. Click **Create Integration**, and then select **Generic Integration** from the type drop-down.
4. Enter “aboDiagnosisBackupLog” for the Name.

   IMPORTANT: This name is required, since is referenced by the updated `abo_addDiagnosticInfo` function

5. For the **Request URL**, specify an external POST endpoint with a request body property capable of accepting long text strings.
   - If you are using a dummy inactive asset record for the backup storage, the URL can be in the following format:
     `https://{hostname}/rest/v16/assets/{id}`
   - If you are using a service other than the dummy asset, make sure the following line within `abo_addDiagnosisInfo` function is modified to point to the property in the endpoint which will store the long text string:

     ```
     backupProperty = "attributes";
     ```

     The asset rest service has a field named "attributes" whose value is expected to be a JSON string.
6. Specify the **Username** and **Password**.
7. Select the **Enable Integration** option.
8. Click **Save**.

Note: If the Generic Integration is not setup correctly, you might not be able to see the entry in `GlobalLogBackup` and you may get error when diagnosis is turned on.

D: ABO LIMITATIONS

The following is a list of ABO limitations:

- The asset repository only stores the latest asset information.
- The Asset import feature has the following limitation when BOM mappings are used on configuration array attributes.
    - When assets are created using normal configuration steps, "conditionIndex" (an internal BOM Item property) is set correctly for BOM items with BOM mappings based on the configuration array attributes.
    - But when assets are imported, "conditionIndex" is set to the default value of 0. For reverse BOM mapping to work with array type configuration attributes, the appropriate "conditionIndex" must be set and should not be 0.
    - Due to this limitation, reverse BOM mapping doesn't work as expected when modifying an imported asset that has more than one child BOM item, since "conditionIndex" is set to 0.
    - When the "conditionIndex" is set to 0, reverse BOM mapping is not able to convert the BOM structure back to configuration and the corresponding BOM items are not available during configuration.
    - Since these BOM items are not available in configuration, when this configuration is saved to commerce, the action codes for these BOM items are set to "Delete" instead of "-" (NO_UPDATE).
- Nested model support limitations:
    - Assets with nested models can only be created from CPQ. Assets with nested models imported from an external system cannot directly launch a configurator session using the modify process.
    - For 21A and earlier, the "copyLines" action is for simple models and does not work for lines created through ABO because of the changes to the "instanceId_I" initialization. Therefore, you should add error checking to the copyLine action. This limitation is removed starting in 21B by enabling the feature in the CPQ admin, setting the proper attribute defaults and in the Initialization tab for the "copyLines" action. Refer to the Oracle CPQ Administration Online Help or the Oracle CPQ 21B What's New for details.
    - For 21A or earlier, the "copyTransaction" action has the same limitation as the "copyLines" action and does not work for transactions containing lines created through ABO. Similarly, this limitation is removed beginning in 21B.
- For 20D or earlier, when there is a set type recommendation rule defined during the modify asset flow, the attribute value stored in asset will be replaced with the value recommended by the set type recommendation rule. This limitation is removed beginning in 21A.
- The Commerce SOAP service creates a new asset order and does not support the following business processes: Modify, Follow-On Order, or Terminate Order.
- Beginning in update 20A, CPQ enforces a limit on the "usersessionset" BML function that is leveraged by ABO functionality.
- A modify line item and a resume or renew line item for the same asset cannot be created on the same date. If the user attempts to perform these actions, a validation error will be reported.
- Limitations for 18C and earlier ABO implementation packages:
    - For ABO Implementation Packages prior to 18D, the Modify and Follow-On Order operations are not guaranteed to return the same selection state as the original order line used to create the asset. The operations rely on BOM Mapping to recreate the configuration selection. If the configuration selection attribute does not map to the BOM, there is not enough information available to determine the value for the configuration attribute. This behavior is different from the regular Reconfigure flow. For ABO Implementation Packages beginning in 18D, the configuration attributes are no longer required to have BOM Mappings.
    - Reconfigure is not available for terminated, suspended, resumed, or renewed lines prior to CPQ 18D. The customer has to create a follow-on order to make changes on resumed or renewed lines.
    - For ABO Implementation Packages prior to 18D, then the BOM's root part number (not the model) always displays when suspending, resuming, or renewing an asset.
- Based on the session language used when the Transaction was initially created, the Attribute Summary does not populate for asset lines removed from the current order (e.g. asset lines with action = delete).
- The Follow-On order operation is only available on the desktop UI and is not available on a mobile UI. Customers cannot create Follow-On orders using the mobile UI.

APPENDIX A: KEY CONCEPTS AND DEFINITIONS

This section provides a high-level overview of key concepts applicable to Oracle CPQ ABO implementations. Oracle recommends customers review these key concepts prior to implementing their Oracle CPQ Asset-Based Ordering solution.

Assets

An ABO asset is created when a subscription product order from CPQ is fulfilled. When an order is fulfilled, an asset is created in the CPQ asset repository. Once created, customers can use CPQ transactions and the Subscription Workbench (previously called the Customer Assets page) to view and manage their assets. The CPQ transaction will capture the delta change of the asset when the transaction is fulfilled, only the delta change will be fulfilled. The asset will be updated with the delta change to bring it to the same state as last transaction.

Asset Management

Action Codes

Action Codes track changes to assets during the asset lifetime. There are two types of action codes:

- **Basic**: Basic action codes keep track of changes. Add, Update, and Delete are examples of basic action codes.
- **Business-Specific**: Business-specific action codes keep track of business processes. Business-specific action codes include Terminate, Suspend, Resume, and Renew.

The following action codes are available for line items on quotes and orders.

- **Add** - Adds a new product to the customer's assets.
- **Update** - Updates either an attribute value or delta field value for an existing product (e.g. asset or order).
- **Delete** - Deletes, deactivates, or disconnects an existing product (e.g. asset or order).
- **Hyphen (-)** - Tracks the No Changes to an Existing Asset action using the hyphen character (-).
- **Terminate** - Terminates a subscription service.
- **Suspend –** Suspends a subscription service.
- **Resume** – Resumes a subscription service.
- **Renew** – Renews a subscription service.

> **Note:** You must add the Action Code attribute to the Transaction UI layout after importing the ABO implementation package. Refer to Add ABO Items to the Commerce Layout, you can customize where action codes display, such as displaying the action codes on sub-document layouts or main document Line Item Grids.

Fulfillment Status

The fulfillment status identifies the current state of an order and the associated saved BOM instance, which is the asset information obtained from the CPQ Model Configuration page.

The fulfillment status can contain four possible values:

- **CREATED** – Upon creating the configured lines, the fulfillment status is set to CREATED and indicates the order has not been submitted for fulfillment.
- **BEING_FULFILLED** - Indicates the order was submitted to the fulfillment system and CPQ has not yet been notified of the order's fulfillment.
- **FULFILLED**: Indicates the order is fulfilled and assets have been created in CPQ.
- **CLOSED**: Indicates the order is closed or cancelled.

> **Note:** The values for Fulfilment Status in the configuredBomInstance for the external order use case are slightly different than what is used in the CPQ Commerce Line Item Grid, an Empty value is used to represent the "CREATED" status, and "CANCELLED" is used to represent the 'CLOSED" status.

Asset Creation

Creating an asset generates a traceable item that integrates with your fulfillment system. The fulfillment system or external client integration use synchronize REST APIs to create and update assets in the CPQ asset repository. Transaction based REST APIs are used to update the fulfillment status on the transaction line item. Transaction line Item status can be Created, Being Fulfilled, or Fulfilled.

Example Process Flow:

1. When sales user adds subscription products to a CPQ transaction, the transaction line items display a "Created" status.
2. When the sales user submits the transaction order to the fulfillment system, the transaction line item status changes from "Created" to "Being Fulfilled".
3. When the fulfillment system fulfills the order and notifies CPQ, the transaction line item status changes from "Being Fulfilled" to "Fulfilled".
    - For assets created in CPQ, the fulfillment system fulfills the order and notifies CPQ via an integration flow that uses the synchronize REST API and transaction based REST APIs to update the fulfillment status on the transaction line item.
    - For assets are created in an external client application, the fulfillment system fulfills the external order and notifies the external client application via an integration flow that uses the Configuration BOM Instance REST API to update the fulfillment status on the transaction line item.
4. When the order is fulfilled, the asset is recorded in the CPQ asset repository.

> **Notes:**
>
> - When using REST APIs to directly update the asset repository, refer to Asset Population via Direct REST Service to ensure proper asset structure.
> - To convert the BOM structure directly from CPQ Commerce to the asset, refer to Appendix K: Sample Update Asset Action.

Asset Modification

Sales users can modify the details of an asset) using the following methods:

- If the order has not been fulfilled (i.e. the status is "Created" or "Being Fulfilled"), the user can create a follow-on order or reconfigure to update an order prior to fulfillment.
- After the order has been fulfilled, the user can use the ABO Modify action on the Subscription Workbench. Sales users can create a modify order by adding, deleting, or updating a component of a fulfilled asset.

The Modify action calculates the changes or deltas to each component in the asset's BOM and creates rows with action codes indicating the changes.

For example, when a user modifies a phone plan subscription from a Prepaid 40 plan to a Prepaid 50 plan two rows are created. One a row to delete the Prepaid 40 asset BOM component and another row to add the Prepaid 50 asset BOM component.

> **Notes**:
>
> - Upon modifying an order, the Subscription Workbench will not reflect the update until the order is fulfilled.
> - You cannot modify an asset with a future start date from the Subscription Workbench. However, it is possible to achieve this using a modify REST service or by launching the configurator with the assetKey and the future request date.

Follow-On Order

A Follow-On Order is a change to an unfulfilled order. When a user creates a Follow-On Order, ABO automatically creates the action codes for each transaction line item based on the difference between the expected state of the asset on the request date and the new configuration.

Reconfigure Action

Customers can also use the Reconfigure action to change an unfulfilled order. The Reconfigure action compares the asset with a reconfigured order line to reflect user-intended net changes in a subscription or asset. Pending updates to order lines that occur before the Reconfigure action request date are included in the comparison. Pending order lines have one of the two conditions. The item is "Being Fulfilled" in another order, or the item is "Being Fulfilled" or "Created" in the current order.

Multiple Open Orders

After a customer places an order, they can change a pending asset configuration before the order is fulfilled and reflected into the asset by modifying an asset or creating a follow-on order for a future date. There are typically two scenarios related to multiple open orders:

- **Scenario 1:** A Customer orders an asset or service to begin at the beginning of the month. This is Order 1.
  Two weeks later the customer puts in a termination order for that service for the end of the month. This is Order 2.

  For example, a customer adds a sports network to their cable service to begin August. 1. This is order 1.
  On August 15, the customer terminates the sports network to take effect on August 31. This is order 2.
- **Scenario 2:** A customer orders an asset or service to begin at the beginning of the month. This is Order 1.
  The next day, the customer orders an additional item for the pending fulfilled service to begin two weeks later. This is Order 2.

  For example, a customer adds a cooking network to their cable service to begin on August 1. This is order 1.
  The customer calls back the next day and adds a sports network to their cable service to begin on August 15. This is order 2.

> **Note:** The open order stacking described in these scenarios is a key concept in ABO. Each order captures the delta made by the customer and sends it to the fulfillment system. Customers can project what the state of an asset will be on a given date. For more information on this concept, refer to Projected Asset Cache BOM and Open Order Stacking.

Asset Termination

The Terminate action on the Subscription Workbench is used to end a subscription. Sales users create a transaction and specify the request date, and then they access the Subscription Workbench to select and terminate the asset. The transaction request date becomes the end date for the asset.

For example, a customer terminates a cable television subscription. The sales user creates a Commerce transaction for the customer and specifies the request date for the termination. The sales user selects the Customer Assets action to display the Subscription Workbench. After the sales user selects and terminates the asset, transaction displays 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.

Asset Repository

In the out-of-the-box implementation of ABO, asset information is stored locally in the CPQ repository. The local asset repository is not versioned and only keeps track of the latest image of the asset. The sample Update Assets action follows the Apply business process that applies all the lines in the current order to the same asset and aggregates the lines to generate a Projected Asset Cache (PAC). The PAC generates the future state of the asset that is compared against the current state of the asset. The PAC generates the delta change to apply on the local asset repository. You can use the out-of-the-box Update Assets sample action to copy the full order hierarchy to an asset.

The PAC projection for the sample Update Asset script is slightly different from the PAC projection for the modify process. During the modify process, transaction lines with a pending fulfillment status from other transactions are considered. During the sample Update Asset process, only the transaction line from the current transaction is considered. For example, On September 1 there is a pending order on transaction line (L1) to change attribute A's value to v1, then on October 1 there is a pending order on transaction line (L2) to change attribute B's value to v2. In the sample Update Asset process, the updated asset for L2 will only update attribute B to value 2. It will not update attribute A, but you would see that attribute A has been updated to v1 if you tried to reconfigure L2.

This mimics the expected ERP logic where fulfillment and the update asset will only update the delta change as indicated in the current order line. In other words, since users can reconfigure a transaction line with an earlier requested date, the last requested transaction line for the asset may not reflect some changes. Differences may occur between the PAC and the last requested transaction line. The transaction lines from other open orders are not considered when using the Apply business process, which ensures all changes are incorporated in the final projected configuration.

Pricing fields are recalculated whenever an asset is loaded into a configuration session. A one-way synchronization occurs from order to asset. Therefore, the price value is not usually carried over from asset to configuration. Beginning in 19B, the unit price of the item is tracked internally. For external order renewals, the price is set to the point when the pending order line was added to the shopping cart. For CPQ Commerce integrations, we expect the price to be recalculated each time the user creates a new transaction to modify or renew an asset.

The transaction lines reflect one of four states: Created, Being Fulfilled, Fulfilled, and Cancelled. The sample implementation of the Update Assets action applies all transaction lines in the current transaction that are not yet Fulfilled by order of requested date. If the transaction line is requested for a future date, the future dated transaction line configuration is synchronized to an asset. In the case of newly created transaction lines, the start date of the asset is set to the requested date of the new transaction line.

There are limitations in current application to store assets outside of CPQ, which is only supported via an override for simple model cases (i.e. models that do not have nested child models).

Asset Population via Direct REST Service

Consider the following tips to ensure assets are populated correctly when updated by direct REST services. This does not include the BOM projection approach in the sample update asset script.

- To ensure the Subscription Workbench page functions properly, the "Customer", "DisplayKey", "Part", and "Quantity" fields are mandatory fields that should be populated with valid data.
- Populate the "rootAsset" and "parentAsset" fields based on hierarchy:
    - The fields should both reference a valid asset.
    - For the root asset record, the parent asset should be null.
    - Use the "assetKey" component fields to update the "rootAsset" and "parentAsset" fields.
- The parent asset cannot be its own ancestor.

  For example, assume A2 has A1 as a parent and A1 has A3 as a parent, therefore A2 would have A3 as a grandparent. It would be invalid for A3 to have A2 as a parent, because it would create a cycle in the hierarchy where A3 has A1 as a grandparent and A3 has A3 as a great grandparent.
- If an order line is marked as deleted, use the asset repository to delete or end-date the asset.
- With the exception of the action code available in the transaction_line_bom_attribute, the Attributes field should contain the same information as the transaction_line_bom_attribute.
- The abo_updateAsset function available in the 18D and later ABO package supports the ability to copy information from a Commerce order line and paste it into an asset.
- If ABO implementation package from 18D or later release is being used and abo_updateAsset is not being used to create/update assets, then Update Configuration REST API action should be invoked explicitly after asset creation or update.

BOM Instance

A Configuration BOM instance is represented by a JSON object. BOM items can be classified as sales items, manufacturing items, or both. These items are defined in the BOM Item Definition table. This BOM instance is referenced during reconfiguration and is used to generate Sales BOMs and Manufacturing BOMs, which can be sent to a back-end system for order fulfillment.

CPQ uses sales items to create a Sales BOM instance during Configuration, sales items are used for quotes, and the Sales BOM instance is used during reconfiguration. In other words, ABO packages only use Sales BOMs.

> **Note**: Sales Items must have corresponding Part Numbers defined in the CPQ Parts Database.

For ABO implementations, the BOM instance contains the delta configuration of the change compared with the projected asset. The changed BOM item will an applicable action code of add, delete, or update.

Projected Asset Cache BOM and Open Order Stacking

The Projected Asset Cache (PAC) tracks the state of an asset across time, it is also known as the Projected Asset State. The PAC process is a function that loads into memory all assets and open orders matching a specific search. The projected configuration can be different based on the transaction's requested effective date. For example, consider the following shirt configuration:

- An asset has the following configuration: Color=Red and Size=XL
- Order 1 is placed to change the Color to Blue, it is pending to take effect on August 1
- Order 2 is placed to change the Size to Small, it is pending to take effect on September 1
- If we create a new modify third order for the same asset to take effect on August 15, the projected configuration when we open the configurator would be Color=Blue, Size = XL
- If we create a new modify third order for the same asset to take effect on September 15, the projected configuration when we open the configurator would be Color=Blue, Size = Small

In the default implementation, an open order is referring to an order in the “Being Fulfilled” state. After retrieving the data, the PAC builds the future requested state of the asset product instances. This is accomplished by taking into consideration the asset matching the search specification and applying all open orders due to complete before the specified date. The process then applies the current quote to generate the future requested state.

There are two key steps in calculating projected assets:

1. Retrieve relevant data:
    - Load the asset data
    - Find and load all open orders associated with the asset
    - Load the current changes made to quote line items associated with the asset.
2. Build the future projected requested state
    - After retrieving the data, build the future requested state of the asset.
    - Consider all assets matching the search specification and apply changes from all open orders due to complete prior to the specified date.
    - Apply the changes made in the current quote to generate the projected requested state of the asset.

Delta Functionality

When users request a change to an existing asset (i.e. product or service) or a projected future asset, they usually initiate a Delta quote. Delta functionality compares a projected original asset BOM instance with a new asset BOM instance and creates a new BOM instance with action codes indicating the components that changed.

The delta functionality is implemented in abo_delta BML, which is defined in BOM Admin > Declare Bom utility.

BOMs for Delta Functionality

There are four BOMs used for comparison to calculate the differences:

- **ConfigBom**: Corresponding to the Configuration user selections in Configurator UI
- **PacBom**: (also known as the priorBom) this is the base against which delta calculation will be conducted. It is read from user session cache within the abo_delta BML script, and the entry in the user session cache is prepared before Configurator launch.
- **InputBom** is what was passed to the Configurator to launch the UI
  - The InputBom does not participate in the delta process directly. The only reason it is supplied is for reconfiguration.
  - If the line is newly created during the prior save, we want to carry over the information from the prior save instead of creating the line from scratch, especially to preserve the assetkey from the prior save of the current line.
- **DeltaBom** is the output of the delta process and is used to generate transaction lines in commerce.
  - When ABO is not enabled, the deltaBom will be the same as the configBom
  - When ABO is enabled for an existing configuration, the deltaBom might contain additional/different lines than the configBom. The new lines represent the lines in the original configuration that are unselected by the user.
  - In commerce, those extra lines will be represented as a line with an actionCode of Delete.
  - In addition to the action code, the delta process also assigns the assetkey for the newly added BOM item.

Projected BOMs - PacBom vs InputBom

The projected asset cache contains two BOMs, the PacBom and the InputBom.

- For Modify operations the PacBom is the same as the InputBom
- For Reconfiguration the PacBom and InputBom are different
  - The PacBom is the projection before applying the current line reconfiguration
  - The InputBom is the projection after applying the current line reconfiguration

    For example:
    - An asset has the following configuration: Color=Red, Size=Large
    - The current line has a change to Color=Blue
  - When we try to reconfigure the current line the PacBOM has Color=Red, while the InputBOM has Color=Blue
  - Strictly speaking, the InputBOM is used to bring the configurator up to the same configuration as last saved, and the PacBOM is used for delta comparison at Update Transaction time.
  - In the above example, if the user changes the color to red and the size to small during reconfiguration, and then saves.

    The delta configuration saved to Commerce will be:
    - Color No-change since the latest configuration has the same value as the pacBom and the asset
    - Size – Change to Small

> **Notes:**
>
> - In 18D and later ABO implementations, there is a similar concept for PAC Configuration and Input Configuration. It corresponds to the structure for storing configuration attribute values that are not mapped in BOM mapping and have an internal property of bomitem.
> - Until the order is submitted, no matter how many times the user reconfigures and saves the line, the delta comparison is always against the projected stage before the current order, because the delta is about the aggregate change the user made within current order. The comparison between different reconfigurations within the same order is not important.

Delta Process Logic

The following items describe the logic within the delta process.

- When a component only exists in a new instance, add the component to the delta BOM using the Add action code.
- When a component only exists in the original instance, add the component to the delta BOM using the Delete action code.
- When a component exists in both the original and new instance, add the component to the delta BOM and compare the component attributes.
    - If there are no changes, set the component action code to hyphen (-).
    - If changes were made or an attribute was added or deleted, set the component action code to Update. The component action code is set to Update in the following scenarios:
        - The addition or removal of a BOM attribute
        - The update of a BOM attribute
        - The update of one of the three key attributes: `partnum`, `qty`, `parentId`
        - The update of one of the attributes listed in `commerceAttributeInDelta` setting.

          The "`commerceAttributeInDelta`" property was added to the "deltaBomSvcSetting" section in the "defaultContextJson.txt" file in the 19B ABO Implementation Package.

          This property determines which Commerce fields are used for comparison during the delta process. The reference package specifies `contractStartDate` and `contraceEndDate` as the list of fields for comparison.

          ```
          "commerceAttributeInDelta" : ["contractStartDate_l", "contractEndDate_l"]
          ```

        - Refer to Appendix J: Default JSON Context File for additional information.
- The child action codes reflect changes to the child entity but not the parent items.
- Attribute-level changes display new and updated attributes summarized in the Attributes field in a comma-delimited format. Deleted attributes do not display in the Attributes field.
- Apply simulates the impact of a set of changes on an original asset. The output is a future asset and is the inverse function of Delta, which projects the future state of an asset. If A1 delta A2 = C, then A1 apply C = A2.

Matching the Child BOM Item between the Original and New Instance

CPQ uses the "assetKey" attribute to compare the old and new instance of a given BOM item. An internal BOM ID gets associated to a given assetKey and is generated dynamically based on the user's selection on Configuration UI and the BOM Mapping rule. For scenarios where a user deselects and reselects an attribute's value, driving the selection of the same asset, the system cannot conclusively determine the user's intention, which could be the reselection of the same original asset or deletion of the original asset and addition of a new, but the same, asset.

- For the 19B ABO Package with CPQ update 19B or later, CPQ identifies the same BOM item with the original assetKey if the de-selection and selection is change on a non-array attribute. However, for array attribute-based BOM item selection, the assetKey might change. To meet the desired outcome, customers can consider updating the logic in abo_delta BML library to change the behavior.
- Prior to the 18C ABO Package, the matching algorithm was Part Number driven and across the entire hierarchy. There caused an undesirable match issue for both array and non-array attribute-based BOM item selection when the same part appeared multiple times in the BOM Item Definition. The algorithm was updated in the 19B ABO package, see the above item for the behavior description.

System Configuration

System Configuration refers to a product offering which is made of multiple sub products, each of which can have a set of rules to define the structure leading to an n-level nested configuration to define the overall offering or permutations of configurable attributes which go together to make the sub product and ultimately the overall product.

System Configuration refers to the way Oracle CPQ is used to configure and bundle the product or set of products that are sold using a group of related models that together define an entire system. A system is a hierarchical arrangement of connected configurable models with a system root containing all of the other models.

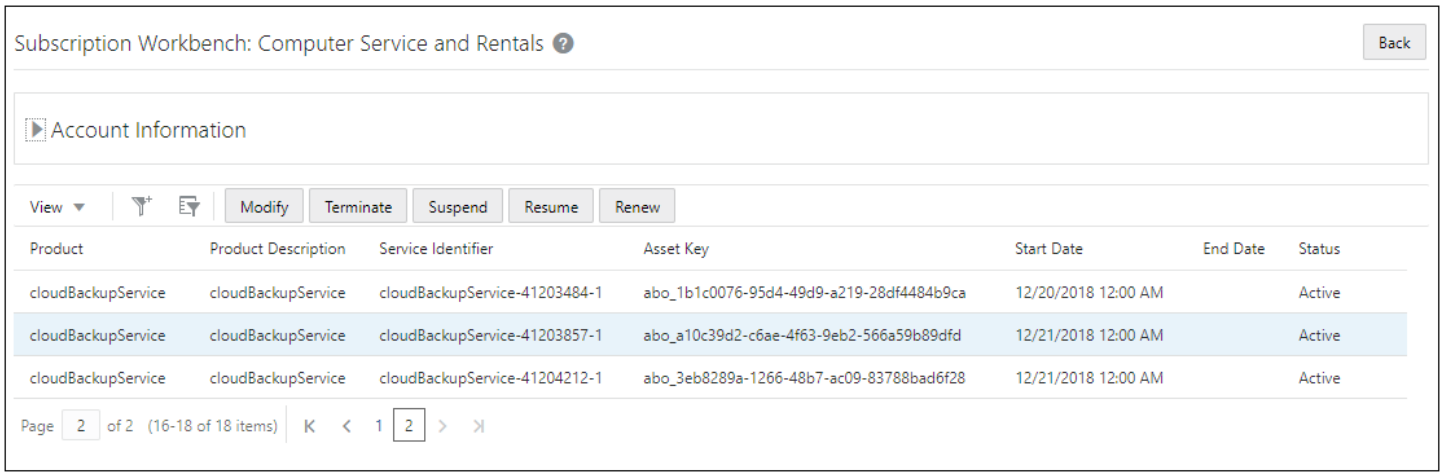
When ABO is enabled, it saves a delta configuration at the end of the configuration session. The delta configuration is stored in the Configuration BOM Instance, but it is not exposed for direct access. Similar to BOM applications, open order stacking will occur for the delta configuration when launching a configuration session. The PAC configuration and input configuration are passed to the configuration session in the BM_PRIOR_CONFIGURATION_KEY and BM_CONFIGURATION_KEY global cache entries.

> **Note:** To use System Configuration with ABO implementations, you must install an 18D or later ABO implementation package.

APPENDIX B: SUBSCRIPTION WORKBENCH

The Subscription Workbench (previously called the Customer Assets page) allows sales users to view and manage active assets. The page has predefined display columns, but you can configure the page using the Customer Assets List available in UI Designer. Sales users can also hide and reorganize columns. The changes will remain active until the user's session expires.

To access the Subscription Workbench, open a quote with an associated customer Id and click **Customer Assets.**



The Customer Assets Commerce action is used to retrieve and display assets for the associated Customer ID in Subscription Workbench. The action is available in the ABO implementation package as a main document (e.g. Transaction) action.

> **Note:** For this flow, you need to add the Customer Assets action to the Transaction UI layout after importing the ABO implementation package. Refer to Add ABO Items to the Commerce Layout.

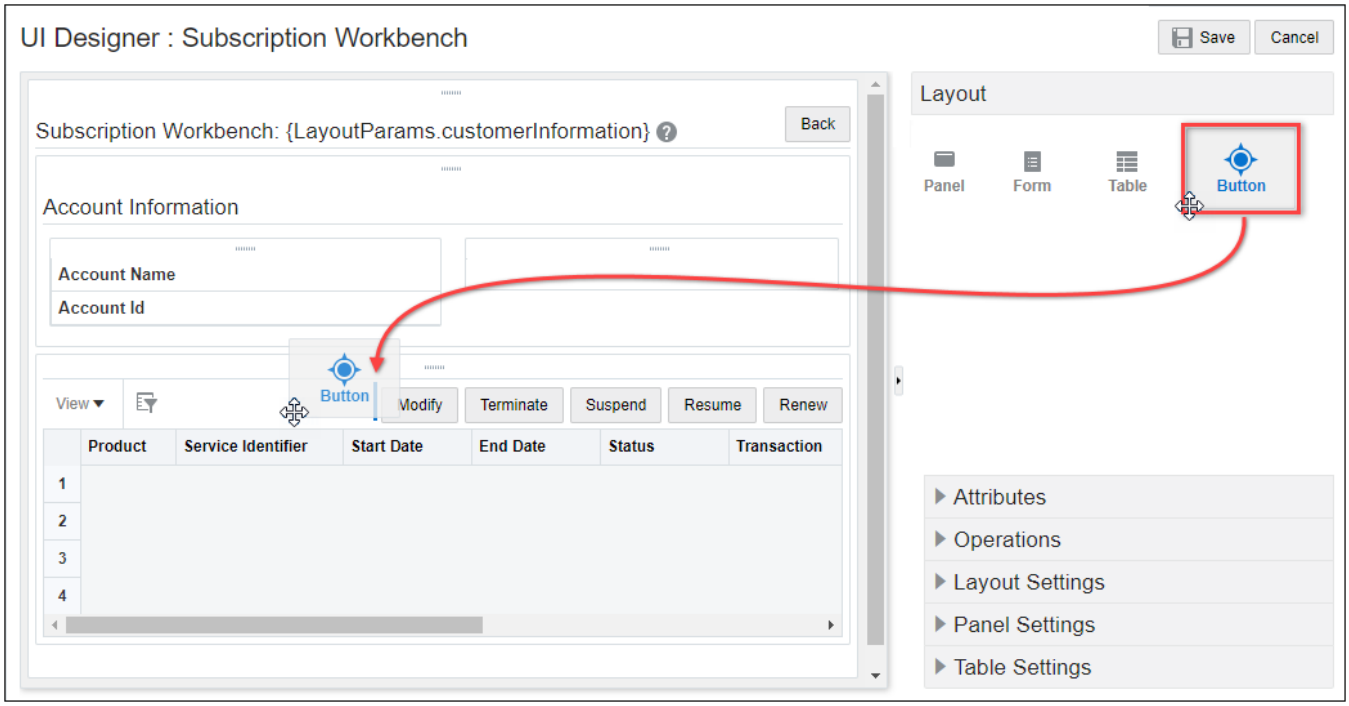
The out-of-the-box Subscription Workbench displays active root assets based on the request date associated with the current Transaction. The page does not display terminated assets or the end date of assets.

Customize the Subscription Workbench

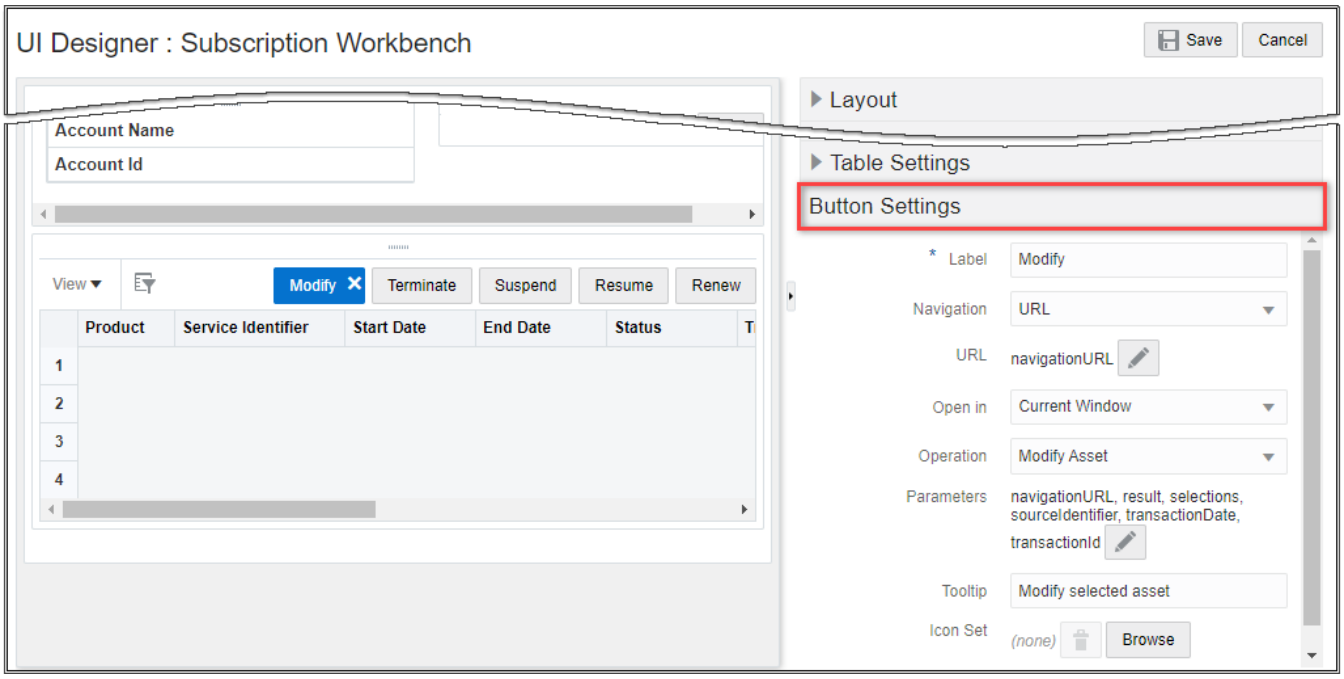
Use the UI Designer to customize the Subscription Workbench. You can customize which attributes are displayed and change the search criteria on the asset table. For additional information about UI Designer, refer to the Oracle CPQ Administrator Online Help.

Complete the following steps to access the Subscription Workbench layout in the UI Designer:

1. Navigate to Admin > General > UI Designer.
2. Select Customer Assets List.
3. To add a button, drag and drop the button icon from the Layout tab to the UI panel.



4. To set or edit the button options, click on the UI panel button and then click on the Button Settings tab.



5. Set the following options for ABO operations:

   a. **Navigation:** select URL
   b. **URL:** enter navagationURL
   c. **Operation:** select the applicable asset operation (e.g. Modify Asset)

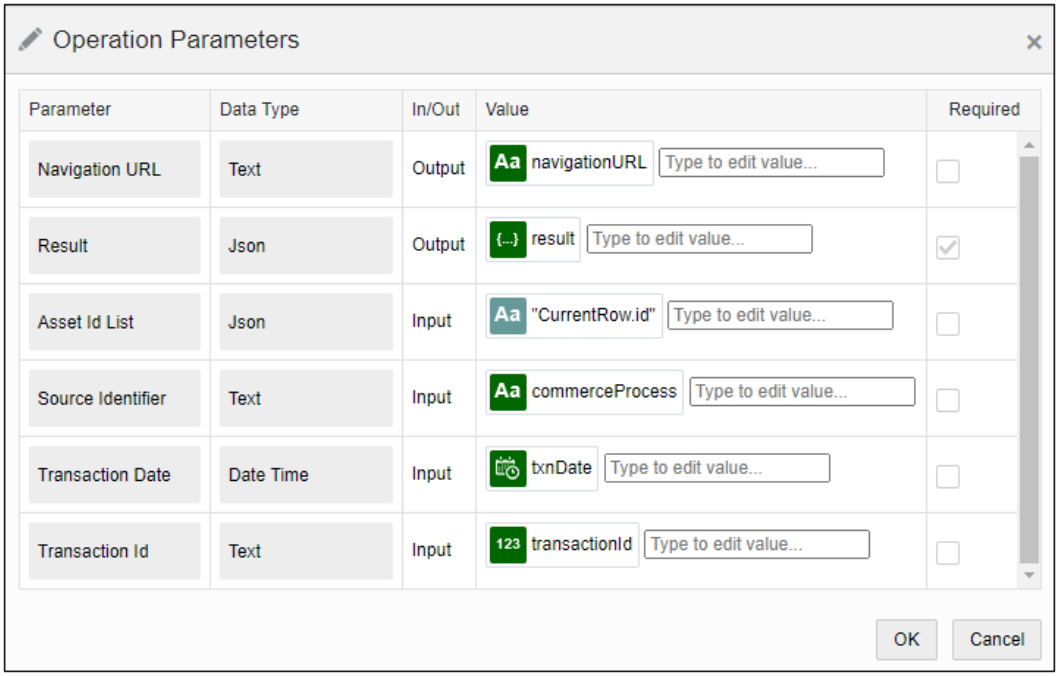
> **IMPORTANT:** Beginning in Oracle CPQ 18D, the Subscription Workbench will always invoke the latest REST service for asset operations. Therefore, even though the "getConfigInstance" and "copyConfigInstance" operations are still available in layout editor, they should not be used in the UI since the "getConfigInstance" and "copyConfigInstance" REST operations were deprecated in 2017 R1 and are only available in v3 and earlier REST services.

d. Click on the **Parameters** edit icon and verify the following settings:

- **Navigation URL:** navigationURL
- **Result:** result
- **Asset Id List:** "CurrentRow.id"

> **Note:** When a user selects multiple rows, this parameter will automatically expand to list the ids of all selected rows.

- **SourceIdentifier:** commerceProcess
- **Transaction Date:** txnDate
- **Transaction Id:** transactionId



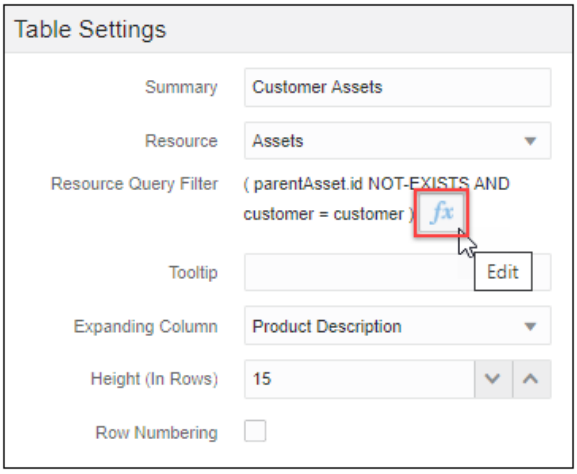
e. Click **OK** to close the Operation Parameters.

6. To edit the table filter:

a. Click anywhere within the table.

b. Open the Table Settings in section in the right panel.

c. Click on the Resource Query Filter edit icon to define the filter applied to the table.



7. Click **Save**.

APPENDIX C: ABO FEATURE SUMMARIES

Appendix C1: 18C and Earlier ABO Features

This section provides information for 18C and earlier ABO enhancements that are still supported in the current release.

The 18C ABO solution depends on the BOM Mapping feature and is built off the "oracelcpqo" reference application process. It relies on the configurator to convert the user selection into a BOM structure, and then convert the BOM structure to a homogeneous asset hierarchy. It allows the user to modify the fulfilled asset configuration and bring it back to the configurator and use reverse BOM mapping to restore the configuration. When the user makes a selection in configuration, the changes are captured in the BOM data.

The 18C ABO solution supports open order stacking and BOM structure projection. For more information, refer to the Projected Asset Cache BOM and Open Order Stacking section in Appendix A.

The 18C solution does not preserve the configurator attribute selection unless BOM mapping is defined for the configuration attribute.

The 18C ABO implementation package supports five asset operations: Modify, Renew, Suspend, Resume, and Terminate.

- Modify Asset will launch the configurator UI with the BOM data converted from the asset.
- In the out-of-the-box package the Renew, Suspend, Resume, and Terminate operations will save the converted BOM data directly to the transaction with a corresponding action code.

The asset operations can be initiated from the Subscription Workbench, or by invoking a REST service on an asset and then using the returned URL to open the configuration UI or transaction UI. For a pending fulfilled future asset, these two approaches are not available, the user has to create a follow-order order to make and submit additional changes to the pending fulfilled configuration.

By default, the delta BOM for ABO operations are saved as a regular transaction line in CPQ commerce. When client side integration is enabled for configuration, the delta configuration is saved in the configBomInstance and is returned in the payload of the client side integration action.

A client side integration is also known as an external order since it is stored in an external order system instead of storing the entire transaction line in CPQ Commerce. CPQ commerce can retrieve the saved delta BOM with the BML getBom function. For external orders, the saved delta BOM can be retrieved from the configBomInstance using the Get Configuration BOM Rest API.

To reconfigure a previously saved configuration, open order stacking logic and the pacBom projection calculation are used for both internal applications and client side integrations. The pacBom calculation ensures any additional changes submitted for this asset in other orders are reflected in the configurator. For reconfiguration, the delta calculation is against the pre-modify projection, not the configuration saved on the current line. For more information, refer to the Projected BOMs section in Appendix A

- For internal applications using CPQ Commerce, you can reconfigure an asset using the Reconfigure action on the transaction Line Item Grid or you can invoke the Reconfiguration API and then launch the returned URL.
- For external orders, you can reconfigure an asset by launching the external_reconfig.jsp using the configId returned from the prior session or can invoke the configBomInstance Reconfiguration REST API and then launch returned URL.

Similar operations are used to create a follow-on order.

- For internal applications using CPQ Commerce, you can initiate a follow-on order from the transaction Line Item Grid or you can invoke the Follow-On Order REST API and then launch the returned URL with minor adjustment of the action BML.
- For external orders, you can initiate a follow-on-order by invoking the Follow-On Order REST API and then launching the returned URL.
- Both internal applications and external orders can also initiate follow-on orders by directly launching the model_config.jsp using the assetKey as a parameter in the URL

In summary, for almost all operations in the 18C ABO implementation, customers can use two different methods to launch the configurator:

- **Method One**: Invoke the appropriate REST API, which returns a URL to launch the configurator. The URL contains a configContextKey that is generated by ABO functionality. The configContextKey provides information used to launch the configurator to the projected selection state. Refer to Appendix H1 for more information on Asset REST services.
  The following example shows a sample URL returned from a REST call.

```
"configuratorURL": https://sitename/commerce/new_equipment/products/model_configs.jsp?
_from_partner=true&product_line=laptop&model=laptopModel&segment=computer&bm_sales_root_bom
_item_id=BOM_RootBomItemId&configContextKey=be2c0b20-49e4-4642-adfc-a207b529b282
```

- **Method Two:** Directly launch the configurator using the assetKey or configId. This will internally invoke the same ABO functionality used for REST calls, and the generated configContextKey is used to launch configurator. Refer to Appendix I to see additional optional parameters. The following example shows a sample URL to directly launch the configurator.

```
https://sitename/commerce/new_equipment/products/model_configs.jsp?_from_partner=true
&assetKey=abo_73426cef-373f-4e82-849b-d979e03263f2&transactionDate=2020-02-06T00:00:000Z
```

Additional Information

- For external orders:
  - To enable client side integration the "`_from_partner=true`" parameter needs to be included.
  - For guest users, additional security access tokens are required when using method two to directly launch the configurator. The following example shows the security access tokens.

```
accessToken=<accessToken>&accessTokenData=<accessTokenData>&publicKeyVarName=<publicKeyVarName>
```

  - For backward compatibility on sites with client side integrations without ABO, the access tokens can be omitted when the configId is for a new configuration instead of a configuration modification.
- For internal application using CPQ commerce, only authenticated users who are logged into the CPQ application can use the additional URL parameters to directly launch the configurator.
- The asset repository is a non-versioned snapshot of data. It supports three ways to populate asset data:
  - The Industry-standard REST API for single asset update.
  - The Oracle CPQ standard REST API synchronize action for asset hierarchy updates. This action is leverages in the out-of-the-box sample update asset script in the ABO package
  - Asset import REST API that allows bulk import of asset data in a csv file.

> **Note:** No matter which approach you use, you should follow the Asset Population via Direct REST Service guidelines in Appendix A.

Appendix C2: 18D ABO Feature Summary

The features in this section are some of the ABO Enhancements that are available beginning in CPQ 18D. You must implement the ABO Implementation Package from 18D or a later release for the following ABO Enhancements.

Support for System Configuration Models

Beginning in Oracle CPQ Release 18D, customers can use ABO in conjunction with System Configuration and the Oracle Commerce Cloud integration. The ABO Implementation Package from 18D or a later release provides support for system assets, allowing customers to use ABO flows on System Configuration models in both Oracle CPQ and Oracle Commerce Cloud order scenarios.

- When using ABO with System Configuration, only root models are available on the Customer Assets page.
- Users can Suspend, Resume, and Renew assets containing root, child, or grandchild models. The models are then added as new line items to the Line Item Grid and can be fulfilled.
- Clicking Terminate from the Customer Assets page will terminate the root asset and all child assets and create a Transaction Line in Commerce. When the Terminate action is fulfilled, the End Date is updated to reflect the date the root asset was terminated.
- This feature can be supported only if assets are maintained in the Oracle CPQ Repository. If assets are stored externally then this feature cannot be supported. In other words, in order for this feature to work as expected, assets data should be created in Oracle CPQ.

Support for Suspend and Resume Actions on Child and Grandchild Models and Parts

Beginning in Oracle CPQ 18D, when using modify, reconfigure, or follow-on order ABO flows on assets, users can Suspend and Resume child and grandchild models and parts in a configurator session. In prior releases, these operations were only supported on the asset as a whole.

This enhancement provides a basic framework that provides customers the ability to customize the behavior by creating Configuration attributes, BOM mappings, and constraints to support the Suspend and Resume operations on child models, according to their business requirements.

Refer to Suspend and Resume for Child Operations for setup information on child and grandchild models and parts.

Retain Configuration Attributes without Mapping in BOM Tables

In releases prior to Oracle CPQ 18D, Configuration attributes were not preserved in ABO flows unless there was a BOM Mapping rule defined for them. Beginning in Oracle CPQ Release 18D, the ABO setup has been simplified by allowing you to add Configuration attributes to the Model Configuration page layout without mapping the attributes in the BOM Mapping tables. The Configuration attribute values are retained and retrieved when an asset is modified or a new follow-on order is created.

- A Sales representative no longer needs to be presented with default values in the configurator UI for attributes that are not included in BOM mapping when modifying an asset or creating a follow-on order.
- With multiple open orders, prior configurations are accurately preserved and retrieved.
- Supports scenarios where orders are stored within Oracle CPQ and outside of Oracle CPQ (e.g. Oracle Commerce Cloud).

Update Configuration REST API Operation

Beginning in Oracle CPQ Release 18D, in order to preserve unmapped configuration attributes and to support system configuration, an additional REST call is required to copy these configuration attribute values from order line and store them on the root asset after fulfillment.

The "updateConfiguration" REST API operation accumulates a projection of all configuration attributes from all relevant open order lines related to an asset and stores this information in the "configAttrInfo" attribute on the root asset.

For additional information, refer to Asset REST APIs in Oracle CPQ Administrator Online Help.

- When orders are stored in Oracle CPQ, invoke the update configuration REST API operation, after invoking the synchronize operation to create or update assets in the Oracle CPQ Asset repository during the fulfillment flow. The following tasks are required to update configuration values:
    1. Invoke the synchronize REST API operation. For example:
       `https://sitename.oracle.com/rest/v7/assets/actions/synchronize`
    2. Invoke the update configuration REST API operation. For example:
       `https://sitename.oracle.com/rest/v7/assets/actions/updateConfiguration`
- When orders are stored outside of Oracle CPQ (e.g. in Commerce Cloud), the "updateConfiguration" action is automatically invoked when the REST API is invoked to update the "fulfillmentStatus" attribute of the "configBomInstance"' resource to "FULFILLED". For orders stored outside of Oracle CPQ, the following flow is expected during fulfillment:
    - When the order is submitted for fulfillment, invoke a REST API call to update the "fulfillmentStatus" attribute of the "configBomInstance" resource to "BEING_FULFILLED".
        - After the order is fulfilled by the fulfillment system, invoke the synchronize REST API operation.
          `https://sitename.oracle.com/rest/v7/assets/actions/synchronize`
        - After the order is fulfilled, invoke a REST API call to update the "fulfillmentStatus" attribute of the "configBomInstance" resource to "FULFILLED".

> **Note:** The "updateConfiguration" action is automatically invoked and an additional API call is not required to update configuration values in assets when the fulfillment status is changed to 'Fulfilled'.

Reconfigure an Asset after Resume or Renew Operations

In releases prior to Oracle CPQ 18D, reconfiguring an asset after performing a Renew or Resume operation will throw an error. Beginning in Oracle CPQ 18D, reconfiguring renewed or resumed order lines is supported.

The customer can now reconfigure the resumed or renewed service from the Oracle CPQ Model Configuration page. This enhancement helps a cable company user make changes to an asset after the contract date is renewed or resumed post service break.

Appendix C3: 19B ABO Feature Summary

You must implement the 19B ABO Implementation Package for the following 19B ABO Enhancements.

Simple Product Support

Beginning in Oracle CPQ 19B, customers can enable Subscription Ordering support to directly add simple products to a Commerce Transaction for an asset-based order. A simple product is a product that does not have its part number associated with any of the related configuration models. When enabled, users can use Quick Add to add simple products to a Transaction without navigating away from the Transaction page. They can also add simple products using a parts search. Simple product assets can be modified, renewed, resumed, suspended, and terminated for internal applications. Navigate to Admin > Commerce and Documents > Commerce Settings to enable the "Enable Subscription Ordering for Simple Products" option.

For more information, refer to the Oracle CPQ Administration Online Help > Asset-Based Ordering Implementations: Simple Product Support section.

> **Note:** If a part is associated with a model and part-model mapping has been created in the Oracle_BomItemDef data table, then the following discrepancy in behavior will be noticed:
>
> - When an adhoc new line item is created for a simple product using Quick Add or Quick Key, it is added as a non-model part. In this case, the part number will be displayed under the Product # column of the Line Item Grid.
> - When ABO operations such as modify, renew, etc. are invoked, the part will be saved as a model line item and the model name will be displayed instead of part number under the Product # column of the Line Item Grid, since the part-model mapping information is retrieved for this scenario.

New Transaction Support for Asset Operations

Beginning in Oracle CPQ 19B, customers can perform asset operations from the Subscription Workbench, previously called the Customer Assets page, for assets without an associated Transaction ID. When a user invokes an asset operation for an asset without an associated Transaction ID, a new transaction is created and associated with the requested operation. This enhancement has been provided to support and integrate with Oracle Subscription Management Cloud in 19B.

Customers can customize the behavior of the new transaction creation process by making changes to the library function abo_prepareNewTxn_ext. Refer to Appendix E: ABO Util Library Functions for additional information.

Multi-Select Support for Asset Operations

Beginning in Oracle CPQ 19B, users can select multiple assets on the Subscription Workbench, previously called the Customer Assets page, to modify, renew, resume, suspend, or terminate. After the user invokes the desired asset operation, the Transaction UI page opens and the appropriate action is displayed for the selected assets.

For more information, refer to the Oracle CPQ Administration Online Help > Asset-Based Ordering Implementations: Multi-Select Support for Asset Operations section.

> **Note:** Support for the multi-select modify operation request 19B ABO Implementation Package. Support for multi-select suspend, resume, renew, and terminate operations does not require the 19B ABO Implementation Package.

Appendix C4: 21A ABO Feature Summary

Business Time Zone Setting for ABO

The Oracle CPQ Ref App used by the ABO package stores the commerce start, end, and request dates as date-only format without timestamp and time zone, while asset start and end dates are stored with full timestamp information. In Oracle CPQ 20C and earlier, during the translation of the date information between commerce and asset, the commerce date is always based on the "0" hour of the server default time zone (usually GMT+0).

End users that are in the same time zone as the server default see the commerce and asset date and time information consistently. However, when an end user is in a different time zone than the server default and their user profile is set to display their local time zone, the translation between the server time zone and their local time zone may overlap dates and cause the commerce date display to be different than the asset date display. This may cause confusion and result in undesirable business date handling.

In Oracle CPQ 21A, an updated 21A ABO package provides a new administrator-defined business-level time zone setting called "businessTimeZone". The business time zone is used, rather than the server default time zone, as the basis for all asset-related date and full timestamp translations. This provides administrators the ability to set a time zone that is more relevant to their business.

The updated package interprets the commerce date fields consistently during ABO processing based on the specified business time zone setting. End users may need instruction on how to interpret their user interface display of start, end, and request date fields in relation to the overall business time zone setting.

> **Notes:**
>
> - The default "businessTimeZone" setting is GMT+0.
> - Once set up, asset start and end dates are populated asset start and end dates are populated based on business time zone; therefore, changing the business time zone setting to a different time zone should be avoided. If a change to the business time zone is required, it must be carefully planned with detailed analysis regarding the impact to legacy data.
> - Only an end user with the same time zone preference as the business time zone will see start and end date matching the business time zone on the asset user interface. End users in different time zones than the designated business time zone will see a different start and end date displayed for an asset based on that end user's time zone preference. End users need to be aware of how the date fields are displayed in relation to the overall company business time zone setting.
> - Refer to Enable ABO for a Custom Commerce Process for customization information.

Appendix C4: Oracle CPQ 21B ABO Feature Summary

Configuration Delta Pricing for ABO

Beginning in Oracle CPQ 21B, delta configuration pricing information is supported inside the Configuration User Interface when implemented with ABO. The delta pricing feature is supported only for BOM Items/BOM Mapping in Oracle CPQ with Oracle CX Commerce integrations. This feature allows for delta pricing information to be easily available to end users.

For more information, refer to the Oracle CPQ Administration Online Help > Asset-Based Ordering Implementations: Configure Delta Pricing for ABO section.

Appendix C5: Oracle CPQ 21D ABO Feature Summary

Reconfigure Line Item Grid Pricing Behavior for Child Model

Beginning in Oracle CPQ 21D, users who have ABO enabled can directly reconfigure child system models from the Line Item Grid in the Transaction UI. In previous releases, users could only reconfigure the child models of a system by first reconfiguring the root model, then navigating to the child to make changes. This feature makes reconfiguring child system models easier for end users.

For an ABO enabled site with system configuration implemented, review the following options to identify steps to enable this feature:

- If you upgraded to the 21A ABO Implementation package, you don't need to do anything to enable Reconfigure Line Item Grid Pricing Behavior for Child Models.
- If you are on the 19B ABO implementation package and do not want to upgrade to the 21A ABO implementation package but would like support for the 21D Reconfigure Line Item Grid Pricing Behavior for Child Models feature, you can update the main and sub-document Reconfigure Actions as specified in the ABO_Final_BML_Actions_21D.zip commerce\action\reconfigure and reconfigure_line files.

Commerce Delta Pricing for ABO

Oracle CPQ 21B provided configuration delta pricing information inside the Configuration UI when implemented with Asset-Based Ordering (ABO). Oracle CPQ 21D introduces a new sub-document Delta Pricing Attribute Set and new Commerce library functions to extend the delta pricing to Oracle CPQ Commerce transactions for ABO sites.

The sub-document Delta Pricing Set contains the following Currency type attributes:

- The Delta Price attribute (_delta_price) displays the price difference for the updated line item.
- The Prior Price attribute (_prior_price) displays the prior price for the updated line item.
- The Rollup Delta Price attribute (_rollup_delta_price) displays the model level price difference of all child items for the updated model.

This feature allows for delta pricing information to be easily available to end users by enabling the following valuable operations:

- View delta pricing information on the Line Item Grid and the sub-document page for both legacy and JET Transaction UIs.
- Support delta pricing on the Transaction UI for Add, Update, Delete, Renew, and Terminate ABO flows.
- Provide rollup delta pricing for new, updated, and modified models.

For more information, refer to the Oracle CPQ Administration Online Help > Asset-Based Ordering Implementations: Commerce Delta Pricing for ABO section.

Search Projected Assets by Customer REST API

This web service provides a consolidated list of fulfilled and pending order asset lines in a hierarchical list for a specific customer. In addition, users can perform search and sort operations on the consolidated list of projected assets.

For more information, refer to refer Oracle CPQ Administrator Online Help > Asset REST APIs or REST API Services for Oracle CPQ.

Appendix C6: Oracle CPQ 22B ABO Feature Summary

Commerce Delta Price for Projected Assets

Oracle CPQ 22B extends ABO delta pricing functionality to include projected assets. The prior price is populated for open orders to establish delta price, roll-up delta, and delta price based on request date when there are multiple orders for the same asset. The reference application sub-document "`netAmount_1`" attribute value is used to populate the prior price value to enable the Delta Price calculation for open orders.

If your site uses the default sub-document "`netAmount_1`" attribute and Commerce Delta Pricing for ABO is enabled, you don't need to do anything to enable Delta Pricing for Projected Assets.

- If your site uses a custom sub-document net amount attribute, you need to specify custom vairable name, refer to Specify Sub-Document Net Amount Attribute for Delta Price.

  > **Note:** If this option is not specified and the sub-document Net Amount attribute does not exist, Delta Price will not be calculated for projected assets.

- If Commerce Delta Pricing for ABO is not enabled, you need to enable Commerce Delta Pricing, refer to Commerce Delta Pricing for ABO.

Appendix C7: Oracle CPQ 23B or Later ABO Feature Summary

Oracle CPQ 23B or later release, adds support for the new commerce standard process implementation. No new enhancements have been added to the ABO implementation in this release apart from adding support to the new commerce process.

Since the new commerce standard process already includes all the attributes, actions, data columns necessary for ABO implementation, the same is not included in the new 23B ABO or later release specific implementation package. The new 23B ABO or later release specific implementation package includes the following:

- Appendix E: ABO Util Library Functions
- Appendix D4: ABO Commerce Library Functions
- Appendix J: Default JSON Context File

APPENDIX D: CPQ COMMERCE AND CONFIGURATION ITEMS

Appendix D1: Commerce Main Document Attributes

Note: All the relevant attributes listed in this section are already included in the new commerce standard process definition available in Oracle CPQ 23B or later release. Hence, they are not included in the 23B ABO or later release specific Implementation Package. The following main document attributes are added, updated, or referenced by the ABO implementation package.

ATTRIBUTE NAME	VARIABLE NAME	TYPE	ABO USAGE	DESCRIPTION
Customer Id	`_customer_id`	Text	This is an important ABO field that is used to populate the asset.customer_refernce	This system attribute is used in CPQ platform-supplied CRM Integrations, this is a special Customer ID where Customer ID as the attribute type. This attribute is associated with the automatically created Auto-fill, Browse, Select Alternative Address, and View actions.
Customer Company Name	`_customer_t_company_name`	Text	For display purpose only, used in display asset action for rendering company summary title on the Subscription Workbench	This system attribute is the company name of the buying organization for this transaction.
Customer First Name	`_customer_t_first_name`	Text	For display purpose only, used in display asset action for rendering company summary title on the Subscription Workbench	This system attribute is the first name of the customer for this transaction
Customer Last Name	`_customer_t_last_name`	Text	For display purpose only, used in display asset action for rendering company summary title on the Subscription Workbench	This system attribute is the last name of the customer for this transaction
Document Number	`_document_number`	Text		This system attribute is a unique document number to be used as document identifier in a process
Transaction Internal Id	`_system_buyside_id`	Text		This system attribute is the current unique buy-side ID. This is the internal identifier for a transaction within a CPQ site.
Selected Document Number	`_system_selected_document_number`	Text		This system attribute is the number of the currently selected sub-document item within a transaction on a CPQ site.
Currency	`currency_t`	Single Select Menu	This is an important ABO field that drives all pricing related attributes	This refApp attribute is used to price and invoice this transaction. • USD (USD • GBP (GBP) • EUR (EUR)

ATTRIBUTE NAME	VARIABLE NAME	TYPE	ABO USAGE	DESCRIPTION
Default Request Date	defaultRequestDate_t	Date	This is an important ABO field that drives the date used for projection and open order stacking	This refApp attribute identifies the date the user requested transaction fulfillment to occur.
Last Updated By	lastUpdatedBy_t	Text	No specific ABO usage. Pulled in due to cross-reference	This refApp attribute identifies the person that last updated this transaction.
Payment Terms	paymentTerms_t	Single Select Menu	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute identifies the payment terms the customer will receive on their invoice. • Net 30 (Net 30) • Net 60 (Net 60)
Reference Application Release Date	refApReleaseDate	Date	No specific ABO usage. Pulled in due to cross-reference	This refApp attribute identifies the date and time on which this version of the Base Reference Application was released.

Appendix D2: Commerce Sub-Document Attributes

Note: All the relevant attributes listed in this section are already included in the new commerce standard process definition available in Oracle CPQ 23B or later release. Hence, they are not included in the 23B ABO or later release specific Implementation Package

The ABO implementation package (ABO_RefApp_Basic_Package_3.zip) adds the following Commerce sub-document attributes.

ATTRIBUTE NAME	VARIABLE NAME	TYPE	ABO USAGE	DESCRIPTION
Document Number	_document_number	Text	This attribute is a critical attribute for ABO logic	This system attribute is a unique document number to be used as document identifier in a process
Price Quantity	_price_quantity	Integer	This attribute is a critical attribute for ABO logic	This system attribute is used for the quantity of a line item
Unit Price	_price_unit_price_each		This attribute is a critical attribute for ABO logic	This system attribute is used for the unit price of a line item
Contract End Date	contractEndDate_l	Date	This attribute is Important for ABO, and critical for Subscription Management	This refApp attribute denote the date that the customer stops to receive the service.
Contract Start Date	contractStartDate_l	Date	This attribute is Important for ABO, and critical for Subscription Management	This refApp attribute denotes the date that the customer starts to receive the service.
Amount (Discount)	discountAmount_l	Currency	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute denotes the discount amount for this line.

ATTRIBUTE NAME	VARIABLE NAME	TYPE	ABO USAGE	DESCRIPTION
Fulfillment Status	`fulfillmentStatus_l`	Single Select Menu	This attribute is a critical attribute for ABO that drives the date used for projection and open order stacking The ABO package overrides the domain for this attribute.	This refApp attribute denotes the status of this line supplied by the fulfillment system after the Transaction is submitted as a sales order. This attribute should have the following domain values. The value inside parenthesis is the variable name and the value outside parenthesis is the displayed text: • Created (CREATED) • Being Fulfilled (BEING_FULFILLED) • Fulfilled (FULFILLED) • Closed (CLOSED)
Instance ID	`itemInstanceId_l`	Text	This attribute is a critical attribute for ABO logic	This refApp attribute is a unique, invariant, and internal identifier for a product instance. The identifier links sales order lines to the assets they change in an asset based order as well as the key stored in a sales order line association. This Id is assigned when the product instance is treated either in an external federated asset repository or in CPQ upon creating a sales order line.
Instance Name	`itemInstanceName_l`	Text	This attribute is a core ABO field, but it is only used for display purposes and it does not drive any business logic for ABO	This refApp attribute is the public identifier of an instance of a product. For goods, this is typically the serial number. For services, the identifier might be another identifier such as telephone number. A unique identifier is not necessary across all product instances.
Last Updated By	`lastUpdatedBy_l`	Text	No specific ABO usage. Pulled in due to cross-reference	This refApp attribute denotes the person that last updated this transaction Line.
Price (List)	`listPrice_l`	Currency	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute denotes the list price of the item.
Amount (Net)	`netAmount_l`	Currency	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute the extended list amount minus any discounts for this line.

ATTRIBUTE NAME	VARIABLE NAME	TYPE	ABO USAGE	DESCRIPTION
Action Code	oRCL_ABO_ActionCode_l	Single Select Menu	This is a core ABO logic field This attribute is created by the ABO implementation package	The value inside parenthesis is the variable name and the value outside parenthesis is the displayed text: • (ADD) Add • (DELETE) Delete • (UPDATE) Update • (TERMINATE) Terminate • (SUSPEND) Suspend • (RESUME) Resume • (RENEW) Renew • (NO_UPDATE) –
Component Attributes	oRCL_ABO_Component Attributes_l	Text Area	This attribute is a core ABO field, but it is only used for display purposes and it does not drive any business logic for ABO This attribute is created by the ABO implementation package	Indicates the change required by the Transaction Line. A comma separated collection of key/value pair attributes that define the configuration of an asset or component plus an action code indicating whether the attribute has changed.
Part Number	part_number		This attribute is a critical attribute for ABO logic	This system attribute
Period	pricePeriod_l	Single Select Menu	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute denotes the period of service purchased at this price. • Per Month (Per Month) • Per Year (Per Year)
Price Type	priceType_l	Single Select Menu	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute indicates whether the amount on this line is charged once or periodically. Add a new domain value to the existing list of domain values.
Request Date	requestDate_l	Date	This attribute is a critical attribute for ABO that drives the date used for projection and open order stacking	This refApp attribute denotes the date the user requested Transaction Line fulfillment to occur.
Unit of Measure	requestedUnitOfMeasure_l	Single Select Menu	Optional, within conditional block of asset mapping logic for refApp process.	This refApp attribute denotes the unit of measure of the item requested by the customer.
Root Asset Key	rootAssetKey_l	Text	**Note:** This attribute is not part of the ABO package, but the Subscription Management package populate this field in the sub-document .default logic This logic is purely ABO driven based on the BOM hierarchy	

Appendix D3: BOM-Related Attributes

The following table shows a list of BOM attributes that are referenced for ABO. These attributes are system attributes that are automatically available to all processes.

FIELD NAME	VARIABLE NAME	TYPE	DESCRIPTION
Line Item BOM ID	`_line_bom_id`	Text	The BOM item instance ID. Format: `BOM-{$UID}` **Note:** 18D and later ABO Implementation packages no longer store the asset key in the `_line_bom_id` field. The asset key is stored in the sub-document `itemInstanceId_l` attribute.
Parent Line Item BOM ID	`_line_bom_parent_id`	Text	The parent BOM item instance ID. **Note:** 18D and later ABO Implementation packages no longer store the parent asset key in the `_line_bom_parent_id` field.
Line Item BOM Attributes	`_line_bom_attributes`	Text	BOM attributes stored as JSON string.
Line BOM Part Number	`_line_bom_part_number`	Text	The part number of the root BOM item. Stores the root BOM Item Part Number in the Model line. Applicable to the root BOM item only.
Line BOM Item Quantity	`_line_bom_item_quantity`	Number	The BOM item line quantity, which is the unexploded line quantity whereas `_price_quantity` stores the exploded quantity.
Line BOM Level	`_line_bom_level`	Number	The BOM item depth (level) in the quote. The value is 0 for the root BOM item. **Note:** System attribute `_linebom_level` is an integer type attribute. • The value is "0" for the root BOM item, i.e. the Model line. • The value is "1" for first level child BOM items • The value is "2" for second level child BOM items, etc. . The value is empty for non-BOM quote lines
Line BOM Effective Date	`_line_bom_effective_date`	DateTime	The BOM effective date. If null, interpreted as the current time. The system attribute is to be stored in the model line effective date for driving the effectiveness of dates in all BOM Mapping.

Appendix D4: ABO Commerce Library Functions

The ABO Implementation package adds the following commerce library functions to the Oracle Quote to Order Commerce process under the main document. The code for these library functions can be found in the release specific ABO_Final_BML_Actions.zip file in My Oracle Support (Doc ID 2182966.1).

Function Name	Description
`oRCL_abo_BuildLineItemHierarchy`	This function iterates through Transaction Line collection and constructs several maps reflecting line items and subscription order parent/child relationships. • This function is called in Sub-Document > Define Advanced Default - After Formulas script • The contents of this function can be found in AboBuildLineItemHierarchy.txt file within the Commerce - Library folder in the release specific ABO_Final_BML_Actions.zip file in My Oracle Support (Doc ID 2182966.1)
`oRCL_abo_PostDefaultsOnLineItems`	This function performs initialization of subscription order related fields for newly created line items • This function is called in Sub-Document > Define Advanced Default - After Formulas script • The contents of this function can be found in AboPostDefaultsOnLineItems.txt file within the Commerce - Library folder in the release specific ABO_Final_BML_Actions.zip file in My Oracle Support (Doc ID 2182966.1)
`oRCL_abo_ReconfigureAction`	This function contains the ABO specific implementation for reconfigure and reconfigures line action scripts respectively. • This function is called in Define Advanced Modify - Before Formulas section of `_reconfigure_action` in Main Document and Sub-Document. • The contents of this function can be found in AboReconfigureAction.txt file within the Commerce - Library folder in the release specific ABO_Final_BML_Actions.zip file in My Oracle Support (Doc ID 2182966.1)
updateAsset	This function creates assets based on the information provided by transaction lines • This function is only available in the 23B or later release specific ABO Implementation package • This function is called in Define Advanced Modify - Before Formulas section of `updateAsset action` in Main Document and Sub-Document. • The contents of this function can be found in updateAsset.txt file within the Commerce - Library folder in the ABO_Final_BML_Actions_<ReleaseVersion>.zip file in My Oracle Support (Doc ID 2182966.1)
oRCL_abo_FollowOnOrderAction	This function contains the implementation for the follow on order transaction line action • This function is only available in the 23B ABO or later release specific Implementation package • This function is called in Define Destination Rule – Define Function section of *oRCL_ABO_CreateFollowOnOrder* `action` in Sub-Document. • The contents of this function can be found in updateAsset.txt file within the Commerce - Library folder in the ABO_Final_BML_Actions__<ReleaseVersion>.zip file in My Oracle Support (Doc ID 2182966.1)

Function Name	Description
oRCL_abo_DisplayAssetsAction	This function is used to launch the customer assets UI page • This function is only available in the 23B or later release specific ABO Implementation package. • This function is called in Define Destination Rule – Define Function section of *oRCL_ABO_DisplayAssets* `action` in Main-Document. • The contents of this function can be found in updateAsset.txt file within the Commerce - Library folder in the ABO_Final_BML_Actions___<ReleaseVersion>.zip file in My Oracle Support (Doc ID 2182966.1)

Appendix D5: ABO Commerce Actions

The ABO implementation package (ABO_RefApp_Basic_Package_3.zip) creates the following Commerce actions.

> **Note:** The commerce actions listed in this section are already included in the new commerce standard process definition available in Oracle CPQ 23B or later release. Therefore, they are not included in the 23B ABO or later release specific Implementation Package.

Customer Assets Action

DOCUMENT	ACTION NAME	VARIABLE NAME	TYPE	DESCRIPTION
Main Document	Customer Assets	`oRCL_ABO_DisplayAssets`	Add From Catalog	This action opens the Subscription Workbench for the transaction's account.

Use the information below to define a Destination Rule for the main document "Customer Assets" action.

The following attributes should be selected:

- System Attribute: `_system_buyside_id`
- Commerce Main Document Attributes:
  - `_transaction_customer_id` (_transaction prefix is from the main document variable name)
  - `defaultRequestDate_t`
- ORCL_ABO Util Library Functions :
  - `abo_addDiagnosticInfo`
  - `abo_initializeContext`
  - `adf_BuildAdfURL`

> **Note:** The attachment "BML and Action Script Files" available on My Oracle Support (Doc ID 2182966.1) contains the "customerAssets.txt" file under the commerce\actions folder.

Create Follow-On Order

DOCUMENT	ACTION NAME	VARIABLE NAME	TYPE	DESCRIPTION
Sub-Document	Create Follow-on Order	oRCL_ABO_CreateFollowOnOrder	Add from Catalog	This action performs an Add from Catalog starting in the selected asset's model configurator flow

Use the information provided below to define a Destination Rule for the sub-document "Create Follow-On Order Assets" action.

The following attributes should be selected:

- System Attributes:
  - _system_buyside_id
  - _system_selected_document_number
- Main Document Attribute: defaultRequestDate_t
- Sub-Document Attributes:
  - _document_number
  - itemInstanceId_l
  - _line_bom_parent_id

> **Note:** The attachment "BML and Action Script Files" available on My Oracle Support (Doc ID 2182966.1) contains the "followonOrder.txt" file under the commerce\actions folder.

APPENDIX E: ABO UTIL LIBRARY FUNCTIONS

The following table identifies the library functions added upon installing the ABO implementation package. These functions are in the ORCL_ABO folder. The BML text files for the following functions can be found on My Oracle Support (Doc ID 2182966.1) in the ABO_Final_BML_Actions_19B.zip file

FUNCTION NAME	DESCRIPTION
`abo_addDiagnosticInfo`	Used to manage diagnosis, which provides in-depth log information when enabled. The function also provides the capability to add, retrieve, and reset the diagnostic log. **Inputs:** `msgTitle` (String), `msgJson` (Json), `action` (String) **Output:** `storageJson` (Json)
`abo_apply`	This function has the core ABO apply logic. Given a list of BOM configurations of the same product instance, this function applies them on top of each other in the specified order and calculates the projected configuration BOM. The following ABO operations use this function: Modify, Reconfigure, Follow-On Order, and Terminate. **Inputs:** `baseBomIn` (Json), `arrBomtoApply` (JsonArray) **Output:** `baseBom` (Json)
`abo_applybom_ext`	This function is an extension point for ABO apply logic to make additional changes on top of the default apply logic for one particular open order BOM. **Inputs:** `origBaseBom` (Json), `toApplyBom` (Json), `appliedBom` (Json) **Output:** `finalBom` (Json)
`abo_applyXA`	**DEPRECATED IN THE 18D ABO IMPLEMENTATION PACKAGE.** The logic for this function was moved to the new BML function applybom.
`abo_bomRollDown`	Invoked from the `abo_loadAsset` function, this function populates additional fields such as 'explodedQuantity'. **Input:** `baseBom` (Json) **Output:** `baseBom` (Json)
`abo_convertDeltaBomItemToAssetItem_ext`	This is extension point for abo `deltaBom` to asset mapping logic to make additional change to map from single bomitem to single asset **Inputs:** `assetItem` (Json), `deltaBomItem` (Json), `rootDeltaBom` (Json), `txtJson` (Json)
`abo_convertDeltaBomtoAsset`	Used to convert a delta BOM to asset format. The asset JSON can be used to synchronize asset data to an asset resource. **Inputs:** `deltaBom` (Json), `txn_json` (Json) **Output:** `rootAsset` (Json)

FUNCTION NAME	DESCRIPTION
abo_copyRootToTransaction	Used in all ABO operations that do not launch the configurator (i.e. Terminate, Suspend, and Resume), this function is invoked when any of the above operations are executed on the asset resource. **Inputs:** actionCode (String), assetKey (String), transactionDate (Date), transactionId (String), commerceProcess (String) **Output:** resultDict (String Dictionary)
abo_dateTimeConverter	This function was added in 21A package, it converts date and time between ISO format and Commerce script format interpreted as business time zone defined for ABO context. The mode parameter controls if it is converting from ISO format to business time zone, or vice versa or getting current day as of business time zone. The operation parameter controls how the timestamp segment (hour:minute:seconds) is handled during conversion. Inputs: inputDateTime (String), mode (String), operation (String) Output: outputDateTime (String)
abo_delta	This function has ABO core delta logic to compare two BOMs. Invoked from a configurator save event, this function compares the final configuration BOM with the previously calculated Projected Asset Cache (PAC) BOM to determine the delta changes and produce a delta BOM. **Inputs:** configBom (Json), commerceProcess (String), inputBom (Json) **Output:** flatConfigBom (Json)
abo_delta_ext	This is an extension point for abo delta logic to make additional changes on top of the default delta logic. **Inputs:** pacBom (Json), inputBom (Json), configBom (Json), deltaBom (Json) **Output:** finalDeltaBom (Json)
abo_generatePAC	Used for generating Projected Asset Cache (PAC) for BOM, this function loads assets and opens order lines to the specified date. This function also applies open order changes to assets to calculate a projected state of the asset as of the requested date. **Inputs:** assetKey (String), transactionDate (Date), transactionId (string), lineNum_to_exclude (Integer), requestAction (String) **Output:** resultBom (Json)
abo_getConfigurationInstance	Used in the ABO Modify process, this function is invoked as a Modify operation on the asset resource. **Inputs:** commerceProcess (String), assetKey (String), transactionDate (Date), transactionId (String), reConfigLineId (String) Output: resultMap (String Dictionary)

FUNCTION NAME	DESCRIPTION
abo_getContext	Used to retrieve the ABO context for the current process from session. **Input:** None **Output:** Abocontext (Json)
abo_getExternalOpenOrderLines	Used to retrieve quote lines from a defined external process for the list of assetKeys passed as input. **Inputs:** arrAssetKeys (String[]), transactionDate (Date), transactionId (String) **Output:** openLinesbyAsset (Json)
abo_getInternalOpenOrderLines	Used to retrieve all open order lines for all internal processes for the array of assetKey passed as input. **Inputs:** arrAssetKeys (String[]), transactionDate (Date), transactionId (String) **Output:** openLinesbyAsset (Json)
abo_getOneAssetState	This function determines the current status of an asset based on the request date. **Inputs:** oneAssetItem (Json), requestDate (Date) **Output:** oneAssetState (Json)
abo_getOpenOrderBoms	Gets open orders for an assetKey passed as input and gets state information, checks conflict, and gets BOM information. **Inputs:** assetKey (String), transactionDate (Date), transactionId (String), lineNum_to_exclude (Integer), requestedAction (String) **Output:** arrQuoteBom (JsonArray)
abo_getOpenOrderLines	This utility BML retrieves the open order lines for given list of assetKey from both internal and external processes **Inputs:** arrAssetKeys (String[]), transactionDate (Date), transactionId (String), lineNum_to_exclude (Integer) **Output:** openLinesbyAsset (Json)
abo_getProductModelInfo	Used to lookup the model data for a particular part number from the Oracle_aboPart2Model Data Table and determines which model to use to launch the configurator based on the part number of the asset. **Input:** partNumber (String) **Output:** resultDict (String Dictionary)
abo_getProjectedRootAssetState	Iterates through open order lines and retrieves projected asset state JSON with current status and potential conflict line information. **Inputs**: originalAssetState (Json), assetsOpenLines (Json), requestDate (Date) **Output:** projectedAssetsState (Json)

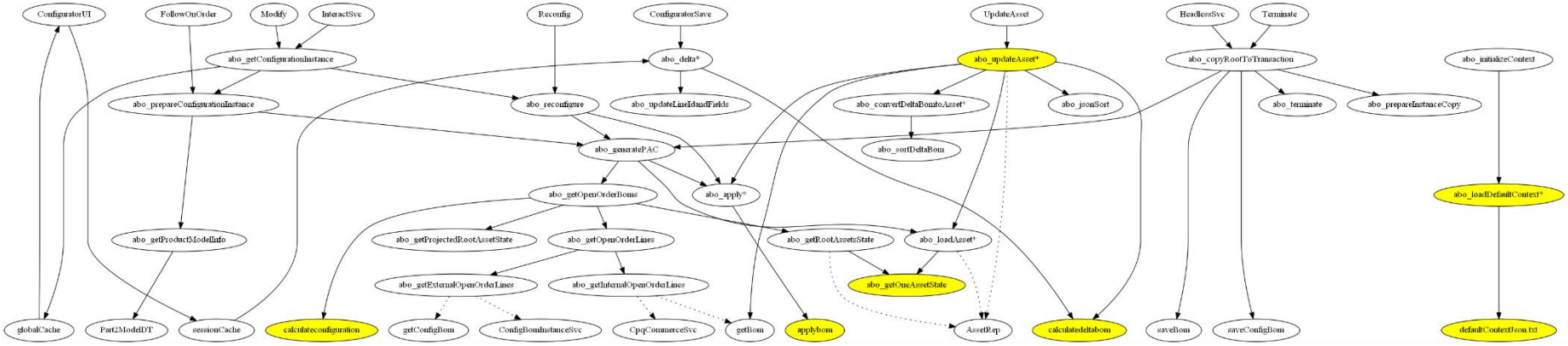
FUNCTION NAME	DESCRIPTION
abo_getRootAssetsState	Used to get the root asset's state in a JSON. **Inputs:** arrAssetKeys (String[]), requestDate (Date) **Output:** assetsState (Json)
abo_initializeContext	Initializes the ABO context and is invoked in the beginning of all ABO flows. **Input:** commerceProcessVarName (String) **Output:** context (Json)
abo_invokeRESTService	Used to execute get/post REST service calls. **Inputs:** method (String), url (String), payloadJson (Json) **Output:** responseJson (Json)
abo_isjsonnull	This function returns true if the key does not exist or the input JSON is null. **Inputs:** inputjson (Json), key (String) **Output:** true/false (Boolean)
abo_jsonCompare	Supporting function for abo_jsonSort that compares two homogeneous JSON objects based on the criteria specified. **Inputs:** jsonObj1 (Json), jsonObj2 (Json), jsonCritArray (JsonArray) **Output:** returnInt (Integer)
abo_jsonSort	Used for sorting records in the input jsonarray based on some input criteria. **Inputs:** unsortedJsonArray (JsonArray), critArray (JsonArray) **Output:** sortedJsonArray (JsonArray)
abo_loadAsset	Used to load asset data from the asset resource and convert to the BOM structure for generating the Projected Asset Cache (PAC). **Input:** assetKey (String) **Output:** baseBom (Json)
abo_loadAssetItem_ext	This is an extension point for abo load asset logic to make additional changes to map from single asset to single bomitem **Inputs:** assetItem (Json), bomItem (Json), rootAsset (Json), rootBom (Json) **Output:**
abo_loadDefaultContext	This utility function loads default context from file. **Input:** commerceProcessVarName (String) **Output:** Abocontext (Json)

FUNCTION NAME	DESCRIPTION
abo_prepareConfigurationInstance	Used during the ABO Modify process and the ABO Follow-On Order process, this function will calculate the projected state for a specified product instance as of requested date and store this information in user session cache. **Inputs:** assetKey (String), transactionDate (Date), transactionId (String) **Output:** resultDict (String Dictionary)
abo_prepareInstanceCopy	Invoked from the abo_copyRootToTransaction function, this function has logic specific to the Suspend, Resume, and Renew operations. **Inputs:** configBom (Json), txnDate (Date), actionCode (String) **Output:** configBom (Json)
abo_prepareNewTxn	This function is used in the new transaction creation process and is called during ABO operations such as terminate, suspend, resume, renew, and modify. **Inputs:** assetKeys (String[]), transactionDate (Date), sourceIdentifier (String) **Output:** resultJson (Json)
abo_prepareNewTxn_ext	This function is provided as an extension point for abo_prepareNewTxn function. It can be used to enhance the logic and make incremental changes to the new transaction creation process. **Inputs:** assetKeys (String[]), assetJson (Json), newTxnPayloadJson (Json) **Output:** newTxnPayloadJson (Json)
abo_processDeltaBom	DEPRECATED IN THE 18D ABO IMPLEMENTATION PACKAGE AND NO LONGER USED IN DELTA PROESS. The logic for this function was moved to the calculatedeltaBom function.
abo_processDeltaItem	DEPRECATED IN THE 18D ABO IMPLEMENTATION PACKAGE AND NO LONGER USED IN DELTA PROESS. The logic for this function was moved to the calculatedeltaBom function.
abo_processDeltaXA	DEPRECATED IN THE 18D ABO IMPLEMENTATION PACKAGE AND NO LONGER USED IN DELTA PROESS. The logic for this function was moved to the calculatedeltaBom function.
abo_reconfigure	Used in the Reconfigure operation, this function captures the core logic specific to this operation. **Inputs:** lineBomId (String), lineTrx_Date (String), bs_Id (String), line_Num (String) **Output:** resultMap (String Dictionary)
abo_sortDeltaBom	This utility is used to sort the Update Asset's deltaBom so that parents appear before children for non-delete lines, and children appear before parents for delete lines. **Input:** deltaBom (Json) **Output:** deltaBom (Json)

FUNCTION NAME	DESCRIPTION
abo_terminate	Invoked from the abo_copyRootToTransaction function, this function has logic specific to the terminate operation. **Inputs:** pacBom (Json), txnDate (Date) **Output:** pacBom (Json)
abo_updateInvocationContext	This utility function is used to retrieve the context for current process from the session, set the invocation context json in context, and return context. The parameters in invocationContext should be set before calling this function. **Input:** invocationContext (Json) **Output:** Abocontext (Json)
abo_updateLineIdandFields	**DEPRECATED IN THE 18D ABO IMPLEMENTATION PACKAGE AND NO LONGER USED IN DELTA PROESS.** The logic for this function was moved to the calculatedeltaBom function.
adf_BuildAdfURL	Used to build the URL to access the ADF Runtime page for any layout with appropriate URL encoding, removes any illegal characters from a URL, and replaces them with URL-friendly character codes. **Inputs:** layoutCode (String), layoutParams (String Dictionary), buysideLayout (Boolean) **Output:** result (String)
adf_EncodeURIComponent	Used to encode the URL parameter value based on the HTTP URL standard. **Input:** parameter (String) **Output:** str (String)
abo_updateAsset	Entry level utility BML for internal order update asset functionality, also used by external order update asset functionality. **Input:** txn_json (Json), lines_json_array (JsonArray) **Output:** responseJson (Json)
abo_updateAssetForExternalOrder	Entry level utility BML for external update asset functionality. **Input:** arrConfigIds (String[]), inputJson (JsonArray) **Output:** responseJson (Json)

APPENDIX F: ABO BML FUNCTION DEPENDENCY DIAGRAM

The following diagram identifies the BML library functions used by an Oracle CPQ 18D ABO implementation and the actions and REST services that invoke the library functions.



Note: The library functions and REST services introduced in the Oracle CPQ 18D ABO implementation are highlighted in yellow. The arrows represent input passed from one REST service or BML function to another.

APPENDIX G: BML SYSTEM FUNCTIONS AND VARIABLES FOR ABO

Get BOM Function (getBOM)

Syntax: `Json getbom(Integer bsId, Integer lineNumber [, String[] lineFields [, Boolean validateBomModel [, Boolean flattenChildProducts [, Boolean isSalesBom]]]])`

Description: Used by ABO to retrieve the saved sales BOM from open orders.

- `validateBomModel` should be false, otherwise, it will not retrieve lines with a "Delete" actionCode.
- `isSalesBom` should false, for manufacture BOM, whole BOM will be regenerated thus ABO information stored in commerce line will not be present in response BOM
- `flattenChildProducts` is usually true for ABO implementations.
- `lineFields` identify additional line attributes retrieved from the transaction line, which are then stored in BOM fields property.
    - The attributes loaded via this function during the open order loading step of PAC calculation are passed thru the configurator so they will be included in the delta process even if they do not have a defined BOM attribute mapping.
    - Transaction line attributes participating in BOM mapping should be included during the `getBom` call as well.
    - The `getBom` function is also used to retrieve the BOM from transaction lines during the sample Update Asset process.
    - Additional attributes, such as `list_price` and `price_type`, are included to provide typical information that needs to be copied from the transaction to an asset during this process.

Save BOM Function (savebom)

Syntax: `Integer savebom(Integer bsId, Json bomJson [,String configurationKey])`

Description: Used by ABO to save the flattened BOM to a transaction and return the document number of the created root line.

`saveBOM` does not create the full configuration structure as from the configurator and it does not validate the BOM structure against the latest model definition.
The reconfiguration actions are required to re-validate the saved line.

- (Optional) The `configurationKey` is returned from the `calculatedeltabom` function and is used to identify the global cache entry for the projected configuration.
- You should either leave this value empty or only pass a valid projected configuration key.

Save Configuration BOM Function (saveconfigbom)

Syntax:
`Integer saveconfigbom(Json configBomJson [,Dictionary instanceAttributes [, configurationKey]])`

Description: Saves a client integration BOM instance (I.e. configBomInstance) and returns the `configId`.

Similar to the `saveBom` function, the configuration BOM is not validated against the latest configuration as part of this call.

- (Optional) The `configurationKey` is returned from the `calculatedeltabom` function and is used to identify the global cache entry for the projected configuration. You should either leave this value empty or only pass a valid projected configuration key.
- `instanceAttributes` contains input attributes (e.g. `sourceIdentifer`, `transactionDate`, and `transactionId`) for the asset REST action initiated by the `saveconfigBom` call.

  The `instanceAttributes` contents are saved into the configBomInstance so the original request context can be used to re-instantiate the session when the returned `configId` is used in an external reconfiguration process.

  This information is not needed for `saveBom` in the CPQ commerce case, since it can be derived from transaction line itself.

Get Configuration BOM Function (getconfigbom)

Syntax: `Json getConfigBom(Integer configId, Boolean flattenChildItems)`

Description: Retrieves the stored configbom via the saveConfigBom API or the configBom created via an external client application Configurator UI session.

- `configId` is the Configuration ID for the client side integration action.

  Note: The configId parameter is not the same as the configuration_id system attribute.

  - For UI integrations, the client side integration action returns the `config_id` in the response JSON.
  - REST APIs for other services such as Terminate, Renew, Suspend, and Resume orders use the `saveBomConfig` BML function
    and return the "configId" in the `lineId` field.
- `flattenChildProducts` – set this Boolean parameter to "true" for ABO implementations to flatten child items and return all descendant BOM items as direct children of the root BOM item.

Calculate Delta BOM Function (calculatedeltabom)

Syntax: `Json calculatedeltabom(Json priorBom, Json currentBom, Json inputBom [, Json setting])`

Description: This function compares the `priorBom` with `currentBom` and then returns the difference between the two with appropriate action code for each item.

`Calculatedltabom` invoked from the `abo_delta` function when a user clicks Add to Transaction or Update Transaction to exit the configurator. It replaces the `abo_processDeltaBom` and `abo_updateLineFields` functions in 18C or earlier implementations. The comparison is mainly between `priorBom` and `currentBom`, but if the item exists in both `currentBom` and `inputBom` and doesn't exist in `priorBom`, it will reuse most information from `inputBom`, especially the asset key.

All of the BOMs used in the caculatezdeltaBom function are all flattened BOMs for out-of-the-box ABO implementations, but this function is capable of handling hierarchical BOMs.

- `priorBom` is often called the pacBom,
- `currentBom` is often called the configBom and it corresponds to the configuration selections when the user exits the configurator.
- `inputBom` is usually the initial information that is sent to the configurator during launch.
- (Optional) The JSON `setting` parameter is used to define delta BOM service settings. Refer to the deltaBomSvcSetting section in Appendix J: Default JSON Context File for more information.

  For example, the `assetKeyprefix` option can be used to customize prefix for the generated assetKey value.

  If the assetKey format is much more complicated or the comparison result is not exactly as expected, the BML abo_delta_ext function is the recommended method to customize the delta BOM behavior after default implementation from `caculateDeltaBom`.

Apply BOM Function (applybom)

Syntax: `Json applybom(Json baseBom, Json oneBomToApply [, Json setting])`

Description: Apply the delta BOM from an open order to construct the projected BOM. The internal logic is very similar to the abo_apply_bom function in 18C and earlier implementation packages.

All of the BOMs used in the `applyBom` function are all flattened BOMs for out-of-the-box ABO implementations, but this function is capable of handling hierarchical BOMs

- `baseBom` can come from the following items: an asset, an initial order of new configuration before it is fulfilled, or the result of an earlier `applybom` call to apply the changes from an open order with an earlier date.
- `oneBomToApply` is the delta BOM loaded from and open order line from a `getBom` or `getConfigBom` call.
- (Optional) The JSON `setting` parameter is used to define delta BOM service settings. Refer to the deltaBomSvcSetting section in Appendix J: Default JSON Context File for more information.

Calculate Configuration Function (calculateconfiguration)

Syntax: `String calculateconfiguration(String baseConfigurationKey, JsonArray linesToApply)`

Description: Even though the name is similar to `calculatedeltabom`, this function is actually similar to `applyBom`.

The `calculateconfiguration` function applies a delta configuration set from open lines on top of the asset configuration to produce the projected configuration for all configuration attributes including attributes that are not mapped to the configurator.

Notice there is not a BML function to calculate the delta Configuration since this logic is conducted in code.

- Both the `baseConfigurationKey` input parameter and return value are a key to a global cache entry, whose value is an unpublished JSON structure used to store the configuration for a root asset and its content including internal fields and all configuration attributes including unmapped attributes.

  When the configuration key is empty, it means there is not any configuration information (i.e. the configuration is blank).

- The `linesToApply` element can contain the following items: asset, internal order line in CPQ Commerce, or the configBomInstance for external orders (e.g. Oracle Commerce Cloud). The following sample payload shows the three use cases.

```
[{
    "type": "asset",
    "assetKey": "abo_ec3092ab-e9e3-4cff-8d1d-28be0d20178d"
  }, {
    "type": "internalOrder",
    "_bs_id": 20965158,
    "_document_number": 20
  }, ….{
    "type": "externalOrder",
    "configId": 20966014
  }
```

  Notice all the entries must belong to the same root asset, otherwise the call will be ignored or throw an error. There could be more than one internal or external order but they should be in the ascending order of the reqeustDate, since the order in the array is the order the delta configuration item get applied.

  For the same reason if the asset is present it should be the first item in the array since it is the starting point. In addition, the other baseConfiguration input should be blank for this use case of passing the asset in the linesToApply array.

Convert Hierarchical BOM to Flattened BOM Function (convertbomtoflat)

Syntax: `Json convertbomtoflat(Json bomJson)`

Description: This BML function converts a hierarchical BOM to a flattened BOM.

For ABO implementations, BOMs are usually stored as a two-level flattened structure and are only converted to a hierarchical structure during interaction with the configurator. The `convertbomtohier` function is used to the flattened structure to a hierarchical structure. In the out-of-the-box package, we only convert to a hierarchical BOM when passing an input BOM to Configuration using the bm_sales_root_bom_item parameter.

Convert Flattened BOM to Hierarchical BOM Function (convertbomtohier)

Syntax: `Json convertbomtohier(Json bomJson)`

Description: This BML function converts a flattened BOM to a hierarchical BOM.

The conversion is based on the bom.id and bom.parentId not the bom.fields.instanceId_l (i.e. assetKey). Therefore, if there was any manipulation of the asset key, it needs to be managed separately.

Subscription Enabled Attribute

Variable Name: `_subscription_order_enabled`

Description: This system attribute indicates if ABO delta is enabled (i.e. any BML function is defined at Admin > Bom > Declare Utility.

APPENDIX H: REST SERVICES

The ABO solution leverages a few CPQ REST services in the implementation package. This section identifies key REST service features used in ABO implementations. For more information, refer to the Asset REST APIs topic in Oracle CPQ Administration Online Help or REST API Services for Oracle CPQ.

Appendix H1: Assets

Loading Data into Local Asset Repository

When assets are loaded, we use the expand query parameter to load the entire hierarchy using the descendentAsset option. The descendentAsset option will leverage the root asset FK, The expand descendentAsset query is only feasible when loading the entire hierarchy and cannot be used to load a sub-tree under a child asset. The result of such expand queries is logically equivalent to a two-level flattened BOM.

In the sample Update Asset script, we leverage the synchronize action to update the entire asset hierarchy in a single request. Compared to standard asset action REST APIs, the synchronize action will commit all the changes for a single root asset hierarchy in one database transaction, which is better for maintaining integrity. Note the following few key items for the asset synchronize action behavior:

- The modify_action within the synchronize payload is used to indicate which operation to use on a particular row. With the exception of the delete operation, it is easier to not supply that value so that it will be treated as upsert by default. Explicitly supplying add/update as the value will trigger additional checking such as insert with same asset key and will error out.
- The key used for validation is the assetKey. The Asset id is optional. It can be supplied when available and will trigger additional cross-validation.
- When using descdentAssets in the synchronize payload, the rootAsset field on the child asset should not be supplied because the value will be deducted from the parent entity in the payload based on the descendent asset relationship. The root asset FK on the parent entity (i.e. the root itself) should be supplied.
- When using descdentAssets in the synchronize payload, the parentAsset field should always be supplied
- Within the descendentAssets payload, the processing is linear, thus the order of assets is important.
    - When inserting assets, the parent asset should appear before the child asset.
    - When deleting assets, the child asset should appear before the parent asset, otherwise the validation may fail.

> **Note:** The Asset Population via Direct REST Service topic in Appendix A applies the same approach and those guidelines are also followed in the sample Update Asset script.

Asset Action Types

The asset provides two main types of actions:

- Modify actions that directly launch the configuration UI, internally these actions will invoke `abo_getConfigInstance` BML function.
- Non-modify actions (e.g. renew, suspend, resume, and terminate) which directly save to commerce, internally these actions will `invoke abo_copyInstanceToTransaction` BML function.

Single and Multi-Select Asset Action REST APIs

Before CPQ 19B, REST API actions could only be invoked on a single asset since the asset ID was part of the URL. Internally we look up the assetKey using the asset ID, and then pass the assetKey to the underlying BML. Since the asset ID must be provided in the URL for single asset REST API actions, this method cannot be used for follow-on orders because an asset has not been created yet and is not available in database.

When the multi-selection feature was introduced in 19B, it provided an additional method to invoke actions on assets without passing the asset ID. For the multi-select actions, the assetKey is provided in the payload instead of the URL. Similar to the single asset actions, multi-select asset REST API actions are not available for follow-on orders because there is not an assetKey to provide in the payload.

Refer to refer to Asset REST APIs in Oracle CPQ Administrator Online Help or REST API Services for Oracle CPQ for the exact syntax, sample payload, and response

The following section highlights available parameters and key information. The examples are for modify and renew REST APIs, but all other APIs has similar behaviors.

URI Endpoint:

- **For Single Select Asset**: `https://sitename.oracle.com/rest/v10/assets/{id}/actions/renew`
- **For Multi-Select Assets**: `https://sitename.oracle.com/rest/v10/assets/actions/renew`

Request Parameters applicable to both Single and Multi-Select Assets

NAME	DESCRIPTION
`sourceIdentifier`	The identifier for the integration process • When this parameter is not specified, the default value is "_external_order" • For internal applications using CPQ commerce, this is the variable name of the CPQ commerce process
`transactionDate`	The date and time that the service request needs to be processed or fulfilled.
`TransactionId`	Optional, The current Transaction identifier for external process integrations or bs_id for internal applications using CPQ commerce.

Request Parameters specific to Multi-Select Assets

NAME	DESCRIPTION
`returnBom`	Optional, set this item "true" to return the BOM structure for the requested assets in the response body. The default value is "false".
`flattenHierarchy`	• "true" returns a flattened BOM structure, this is the default value. • "false" returns a hierarchical BOM structure
`assetKeys`	An array of asset keys selected for the specified operation. When "assetKeys" are used, do not use "selections".
`selections`	An array of asset ids selected for the specified operation. When "`selections`" are used, do not use "`assetKeys`".

Response Body Parameters Specific to Multi-Select REST APIs

NAME	DESCRIPTION
`resultTransactionId`	The Commerce Transaction ID, if the Transaction ID is not provided, a new Transaction is created. This does not apply to external order case.
`navigationURL`	• For assets created via CPQ, a successful response includes a navigation URL to the Transaction UI or to launch the configurator. • For assets created via an external order, a successful response is empty or includes a URL to launch the configurator. A URL to launch configurator only returned if all the following conditions are satisfied: • It is a Modify request • A single asset as passed • returnBom = false
`processedList`	When the request is successful, the output contains the processed list of assets: • For internal applications using CPQ commerce, the assets are added as new lines to a transaction. • For external orders, the assets are saved to the configBomInstance. When the navigationURL contains a URL to launch configurator, this field will be empty because no lines have been added to the database at this time.

Response Parameters for Single Select Asset REST APIs and Multi-Select REST APIs with a Single Asset

The response contains two levels, the immediate child is Results and the remaining properties are children of Results.

NAME	DESCRIPTION
Response for Non-Modify REST APIs or Modify REST APIs with returnBom=true	
`Results`	This field is populated by the abo_copyInstanceToTranaction BML function If this function is customized to include additional properties in the results dictionary, the response will also contain these additional properties.
• `LineId`	The Configuration ID for the saved Configuration BOM Instance for external orders,
• `linenum`	The document_number of the saved CPQ transaction line.
• `bomkey`	The id of the root BOM item of the saved CPQ transaction line.

NAME	DESCRIPTION
Response for Modify REST APIs with returnbom=false with a Single Asset	
Results	The following properties are for Modify REST APIs that return a URL to launch the configurator. (i.e. Single Select REST APIs or Multi-Select REST APIs with a single asset)
• product_line	The Product Line variable name used to uniquely identify the model associated with the root asset
• model	The Model variable name used to uniquely identify the model associated with the root asset
• segment	The Product Family variable name used to uniquely identify the model associated with the root asset
• configContextKey	A key to the global cache entry storing the projected configuration information which will be retrieved by the configurator during launch
• bomkey	The id of the root BOM item for the inputBom passed inside the globalcache entry referenced in the configContextKey. This is needed for configurator launching validation.
• configuratorURL	The URL to launch the configurator

Sample Response for Modify REST APIs with a Single Asset returnbom=false

The following sample shows the response for a Modify Asset REST API with a single asset.

- When the response is for a Single Select REST API, the properties highlighted by blue text are returned.
- When the response is for a Multi-Select REST API, the properties highlighted by red text are returned in addition to the properties highlighted by blue text.


```
{
  "result": {
    "product_line": "laptop",
    "model": "laptopModel",
    "configContextKey": "be2c0b20-49e4-4642-adfc-a207b529b282",
    "segment": "computer",
    "bomkey":"BOM_ABOSampleRoot",
    "configuratorURL":
"https://sitename.oracle.com/commerce/new_equipment/products/model_configs.jsp?_from_punchin=true
&actionVarName=addLineItem_t
&id=21325897&product_line=aBOAssetBasedOrdering&model=basicAsset&segment=testbed&bm_sales_root_bo
m_item_id=BOM_ABOSampleRoot
&configContextKey=773ecd36-87a1-4fcf-84df-87f9b2eb36fc"
  },
  "resultTransactionId": "21325897",
  "navigationURL":
"http://sitename.oracle.com/commerce/new_equipment/products/model_configs.jsp?_from_punchin=true&
actionVarName=addLineItem_t
&id=21325897&product_line=aBOAssetBasedOrdering&model=basicAsset&segment=testbed&bm_sales_root_bo
m_item_id=BOM_ABOSampleRoot
&configContextKey=773ecd36-87a1-4fcf-84df-87f9b2eb36fc",
  "processedList": []
}
```

Sample Response for Multi-Select Modify REST APIs with Multiple Assets


```
{
  "resultTransactionId": "21275813",
  "navigationURL":
"https://sitename.oracle.com/commerce/buyside/document.jsp?formaction=cancelAddFromCatalogCookie&
bs_id=21479248&actionVarName=_open_transaction&commerceProcess=oraclecpqo&process=oraclecpqo&txnD
ate=2019-01-27 00:00:00&transactionId=21275813",
  "processedList": [{
      "lineId": "8",
      "assetKey": "abo_05b05644-82de-4240-86eb-8feee4aa90d6"
    }, {
      "lineId": "11",
      "assetKey": "abo_d60556a9-8639-403b-aea0-d0fbd3d73c29"
    }
  ]
}
```


Notes: The following items describe how the asset REST API response information relates to the Subscription Workbench.

- In the out-of-the-box asset layout, asset REST API response items are mapped to the button setting parameters. These parameters can be viewed by navigating to:
  Admin > UI Designer > Asset List > Button > Button Setting > Parameters.
- The layout parameters are directly mapped to the input parameter of the action.
  - The configuration of selections is the literal value of the “CurrentRow.Id”. This special reserved keyword automatically expands to include the ids for all the selected assets on runtime. If you create a new button, you must enter the exact value.
  - Not all available inputs are mapped in the out-of-the-box button configuration, and the list of available inputs is based on when a button was created.
  - If you need to specify the value for a parameter you do not see (e.g. returnBom), you must remove and re-select the same operation.

    IMPORTANT: Before you remove the operation, you should record the assigned value for existing parameters so you can re-enter the same values.
- Results will be assigned to results layout parameter for backward compatibility.
- The navigationURL is assigned to a layout parameter, by default the new navigationURL is used as the navigation target for the button.
  For 19A or earlier, the result.configuratorURL is used for modify action and the backURL is used for non-modify action buttons.

Appendix H2: Configuration BOM Instance Resource

When using CPQ commerce, the BOM structure associated with the configuration is stored indirectly in the commerce transaction line. When a user customizes a product using an external client application to capture the order, the external client application opens the Oracle CPQ Model Configuration page in an embedded iFrame. Once the user customizes the product, the resulting BOM structure along with the unique identifier of configuration (i.e. configId) is returned to the external client application. Both the BOM structure and the configId are also stored by CPQ in the configBomInstance resource and they are accessible via REST services and the getConfigBom BML function.

The configBomInstance function is used to support open order stacking during PAC projection when the corresponding configId is not yet fulfilled. During open order stacking projection, the configId is used to load the associated BOM.

The BOM structure can also be accessible using the Get Configuration BOM REST API on this resource.

Beginning in 18D, the configBomInstance also stores the internal delta configuration associated with the configId. BML or REST APIs cannot directly access this delta configuration. This information is used internally by calculateConfiguration BML when the external configId is supplied in the linesToApply parameter.

The configBomInstance also provides reconfiguration and follow-on order actions, which provides capabilities similar to the asset modify action. When a configId is passed in a REST API action, the result will contain a URL with embedded information that can be used to launch the configurator with the projected state. Beginning in 19B, you do not have to use a REST API for reconfiguration, the reconfiguration can be achieved by using the external_reconfig.jsp and passing the configId as a URL parameter. A follow-on order action can be invoked using a multi-select asset modify action with the assetKey or by launching the model_config.jsp with assetKey as a URL parameter.

Attributes in ConfigBomInstance Entity

ATTRIBUTE	DESCRIPTION
Configuration Id	The configuration Identifier that uniquely identifies the configuration BOM instance. • The Configuration ID (i.e. `configId`) is for the client-side integration action and is not the same as the Commerce system attribute (i.e. `_configuration_id`). • For UI integrations, the client-side integration action returns the `configId` in the response JSON. For other actions such as Terminate, Renew, Suspend, and Resume, REST calls generated from the `saveconfigbom` BML function return the `lineId`.
BOM Instance Id	An invariant key that uniquely identifies the root component of the BOM. Also known as the assetKey.
Configuration BOM Instance	The Configuration BOM Instance data. The Configuration BOM associated with the configBomInstance
Fulfillment Status	Fulfillment Status of the configuration BOM instance. This attribute can be updated.
Action Code	Indicates the change required for this root configuration instance. For example - TERMINATE, SUSPEND, RESUME, RENEW.
Source Identifier	The variable name of the commerce process or external application identifier.
Transaction Date	The date and time when the service request was made. This attribute can be updated.
Transaction Id	The current transaction identifier. This attribute can be updated.

Appendix H3: ABO Usage of Commerce REST Services

There are two uses for CPQ Commerce REST services in ABO:

- ABO leverages REST queries on transaction lines across all transactions to obtain a list of pending fulfillment lines.

  The URL for sub-document REST services is registered as the "sub_Doc_url" in the defaultContextJson file.

  The "`sub_Doc_Url`" used the following format:
  "`$RESTURL/commerceDocuments{ProcessVarName}{MainDocVarName}{subDocVarName}`"

  - `{ProcessVarName}`: The variable name of the Commerce process, the first letter must be capitalized.
  - `{MainDocVarName}`: The variable name of the main document, the first letter must be capitalized.
  - `{subDocVarName}`: The variable name for the sub-document.
- All the attributes referenced for selection, search, or sort specifications in REST criteria must be mapped in a data column.
- To avoid slow performance for SQL queries, the itemInstnace_l attribute, which stores the assetKey, is indexed.
- Beginning in CPQ 19B, the transactionId parameter is optional for asset REST services. When the transactionId is not included, the abo_prepareNewTxn BML function is internally invoked to prepare a payload based on the selected assets. Then the returned payload is used as a document parameter to invoke the same underlying new transaction REST service functionality with a modify action to create the transaction. The modify action name is defined in the defaultContxtJson file.

APPENDIX I: PARAMETERS FOR LAUNCHING THE CONFIGURATOR

ABO provides three methods to launch configuration to change an asset configuration.

1. Invoke a REST service modify action on asset, configBomInstance, or commerce action, which will return a URL to launch the model_config.jsp with the product family/product line/product model variable name, ConfigContextKey, and root item Id. Refer to the Single and Multi-Select Asset Actions REST APIs section in Appendix H for input parameters and sample responses.

   Sample Configurator URL

```
https://sitename.oracle.com
/commerce/new_equipment/products/model_configs.jsp?_from_punchin=true&actionVarName=addLine
Item_t&id=21325897&product_line=
aBOAssetBasedOrdering&model=basicAsset&segment=testbed&bm_sales_root_bom_item_id=BOM_ABOSam
pleRoot&configContextKey=773ecd36-87a1-4fcf-84df-87f9b2eb36fc
```

   When launching the configurator, the projected BOM and configuration will be extracted from the global cache entry referenced in the ConfigContextKey variable and used to instantiate the configurator selection. The following URL parameters are used by ABO in the URL returned by the modify asset REST service.

PARAMETER	DESCRIPTION
`product_line`, `model`, and `segment`	The product line, model, and product line variable name used to determine which model to display.
`configContextKey`	The key to a global cache entry, which contains projected BOMs and configurations, is used to instantiate the configurator selection. This entry is populated by the `globaldictset` BML function in `abo_getConfigInstance`. In the out-of-the-box package, the value of this global cache entry is a JSON object with the following properties: • "`bm_sales_root_bom_item`" - the inputBom used to instantiate the configurator session. If the inputBom is not present, it will be considered a new configurator session. The inputBom must have the property `instruction= preserveInputBom` • "`bm_abo_pac_root_json`" - the priorBOM is used for delta comparison. For modify operation, the pacBom will be the same as the inputBom • "`ConfigurationKey`" - a key to the globalcache entry that contains the projected configuration value corresponding to the inputBom, it is returned by the `calculateconfiguration` BML function • "`PriorCofigurationKey`" - a key to the globalcache entry that contains the projected configuration value corresponding to the pacBom, it is returned by the `calculateconfiguration` BML function. At configurator launch time, all the content in the JSON entry within the globaldict entry referenced by the ConfigContextKey is copied to the user session cache and can be accessed by the usersessionget call during configurator launch.
`bm_sales_root_bom_item_id`	When using the configContextKey to launch the configurator, this is a mandatory parameter. The value must match the Bom.Id for the inputBom within globalcache referenced by the configContextKey. Otherwise, the inputBom is ignored and a blank configuration is presented.
`actionVarName`	The action variable name can be provided in the URL parameter for the "addFromCatalog" action, during add to transaction. Previously, only the "actionId" could be used for this action.

PARAMETER	DESCRIPTION
_from_partner	This parameter identifies whether to instantiate the configurator with client integration enabled. The ABO BML will return a true value if the sourceIdentfier parameter passed to the asset service, if a value is not included the operation will default to "_external_order" which indicates that the order is stored outside Oracle CPQ. When not supplied, the default value is false.

2. Invoke the model_config.jsp directly with the assetKey URL parameter. The configurator invokes the same underlying abo_getConfigInstance BML function as a modify asset action. The response returned by BML is directly acquired by the configurator launching logic. There are a few additional URL parameters equivalent to the action parameters for a modify asset action.

   The following URL parameters should not be included when using this method:

   `segment`, `product_line`, `model`, `configContextKey`, and bm`_sales_root_bom_item_id`

   The following URL parameters might need to be included when using this method:

   `actionVarName` and `_from_partner`

   The following additional URL parameters need to be included when using this approach unless they are marked as optional. These parameters are practically the same parameters used to invoke the modify asset service.

PARAMETER	DESRIPTION
assetKey(String)	An invariant key that uniquely identifies the asset component for the lifetime of the asset. This is a required parameter and this attribute value can be found by querying assets resource.
sourceIdentifier	The Oracle CPQ Commerce Process variable name (e.g. oraclecpqo). This parameter is only required for orders stored in Oracle CPQ. If a value is not included for the "sourceIdentifier", the operation will default to "_external_order" which indicates that the order is stored outside Oracle CPQ.
transactionDate	(Optional) The date (in ISO format) when the transaction is effective. If a transaction date is not included, this value will default to the current date during processing.
id	The transaction identifier. This parameter is only required for orders stored in Oracle CPQ. This parameter is used to synchronize the modified data to the referenced transaction during modify or follow-on order operations. This corresponds to the transactionId parameter in the modify asset service. Notice the name difference.

3. Invoke the external_reconfig.jsp or GetConfiguration SOAP API with the configId parameter. This invokes the same underlying abo_getConfigInstance BML function by passing the assetKey, transaction Id, sourceIdentifer, and transactionDate information stored on the corresponding configBomInstance. The result projected BOM/configuration is used to launch the configurator UI or generate a configuration response.

   Sample external reconfiguration URL

   The configId and access token are typically the only parameters needed for this method.

```
https://sitename/commerce/new_equipment/products/external_reconfig.jsp?_config_id=<configId
>&accessToken=<accessToken>&accessTokenData=<accessTokenData>&publicKeyVarName=<publicKeyVa
rName>
```

Note: For guest users, additional security access tokens are required when using methods two and three to directly launch the configurator.
The following example shows the security access tokens to authorize the usage of the related assetkeys.

```
accessToken=<accessToken>&accessTokenData=<accessTokenData>&publicKeyVarNa
me=<publicKeyVarName>
```

For backward compatibility on sites with client side integrations without ABO, the access tokens can be omitted when the configId is for a new configuration instead of a configuration modification.

APPENDIX J: DEFAULT JSON CONTEXT FILE

In the 18D ABO implementation package, some of the initialization parameters that were previously in the `abo_initializeContext` BML function were moved into the "defaultContextJson.txt" file. This file can be accessed under the ABO folder when you navigate to Admin > Utilities > File Manager.

Some parameters in this file can be changed to customize the behavior according to customer requirements. The BML function `abo_loadDefaultContext` is used to load the default context from this file. This BML function can be further customized to make any modifications to the loaded context.

> **IMPORTANT:** This file is not intended to store large amounts of data. You should not double the size of original JSON file. If the file size is too large, you will run into `usersessionset` file size limitations.

The following are the contents of the defaultContextJson file.

CONTEXT PROPERTY	PROPERTY/VALUE

`siteInfo`	This context property stores the property/value details for REST endpoints, for assets, and the configBomInstance.
	`"siteInfo": {`
	`"localAssetSvc": "$RESTURL/assets",`
	`"localBomSvc": "$RESTURL/configBomInstance",`
	`"siteUrl": "$SITEURL"`
	`},`

`IntProcessInfo`	This context property section stores property/value details for the applicable commerce process.
	If a customized commerce process is used the custom property/value pairs should be added to this array.
	`"IntProcessInfo": [{`
	`"commerceProcessName": "oraclecpqo",`
	`"isExternal": "false",`
	`"main_Doc_Url": "$RESTURL/commerceDocumentsOraclecpqoTransaction",`
	`"sub_Doc_Url":`
	`"$RESTURL/commerceDocumentsOraclecpqoTransactionTransactionLine",`
	`"postNewSaveAction": "cleanSave_t",`
	`"subDocAttributes": {`
	`"oRCL_ABO_ActionCode_l": {"type": "menu"},`
	`"fulfillmentStatus_l": {"type": "menu"}`
	`}`
	`}`
	`],`

	`"commerceProcessName": "oraclecpqo"` This is the variable name of the applicable commerce process.
	`"main_Doc_Url": "$RESTURL/commerceDocumentsOraclecpqoTransaction"` The main document path is "`commerceDocuments`" + commerce process variable name + main document variable name. **Note:** The first letter of the commerce process variable name and main document variable name should be capitalized regardless of how they are defined in Commerce.
	`"sub_Doc_Url": "$RESTURL/commerceDocumentsOraclecpqoTransactionTransactionLine"` The sub-document path is: "`commerceDocuments`" + commerce process variable name + main document variable name + sub-document variable name. **Note:** The first letter of the commerce process variable name, main document variable name, and sub-document variable name should be capitalized regardless of how they are defined in Commerce.

CONTEXT PROPERTY	PROPERTY/VALUE
extProcessInfo	The details in this context property section are used when orders are stored outside Oracle CPQ in an external application such as Oracle Commerce Cloud. **Note:** The following properties should not be modified. "commerceProcessName": "_external_order" "isExternal": "true"
extraLineFieldForAsset	This context property section provides a list of additional fields that are retrieved in a getbom call when the abo_updateAsset BML function is invoked. "netAmount_l": "c" "priceType_l": "s" "requestedUnitOfMeasure_l": "s" "discountAmount_l": "c" "contractStartDate_l": "d" "contractEndDate_l": "d" "itemInstanceId_l": "c" "listPrice_l": "c" "pricePeriod_l": "s"
assetInfo	This context property section provides a list of asset attributes that are retrieved when querying an asset. Additional asset fields should be added to this list if they are required from the assets resource when an asset is queried in the abo_loadAsset BML function. "fieldQueryList":"assetKey,parentAsset,quantity,partNumber,displayKey, attributes,discountAmount,currency,startDate,endDate,bomItemId, parentBomItemId,modelPath,status,suspendDate,resumeDate, descendantAssets.assetKey,descendantAssets.parentAsset, descendantAssets.quantity,descendantAssets.partNumber, descendantAssets.displayKey,descendantAssets.attributes, descendantAssets.discountAmount,descendantAssets.currency, descendantAssets.startDate,descendantAssets.endDate, descendantAssets.bomItemId,descendantAssets.parentBomItemId, descendantAssets.modelPath,descendantAssets.status, descendantAssets.suspendDate,descendantAssets.resumeDate"
ALL_ACTIONS:	This context property section provides details of all applicable actions on an asset order. "MODIFY": {"DISPLAY":"Modify", "CATEGORY":"MODIFY"}, "RECONFIG": {"DISPLAY":"Reconfig", "CATEGORY":"RECONFIG"}, "FOLLOWON": {"DISPLAY":"Followon Order", "CATEGORY":"MODIFY"}, "ADD": {"DISPLAY":"Add", "CATEGORY":"ADD"}, "UPDATE": {"DISPLAY":"Update", "CATEGORY":"UPDATE"}, "DELETE": {"DISPLAY":"Delete", "CATEGORY":"DELETE"}, "NO_UPDATE": {"DISPLAY":"No Update", "CATEGORY":"NO_UPDATE"}, "TERMINATE": {"DISPLAY":"Terminate", "CATEGORY":"TERMINATE"}, "SUSPEND": {"DISPLAY":"Suspend", "CATEGORY": SUSPEND"}, "RESUME": {"DISPLAY": Resume", "CATEGORY":"RESUME"}, "RENEW": {"DISPLAY":"Renew", "CATEGORY":"RENEW"}

CONTEXT PROPERTY	PROPERTY/VALUE

"ALL_STATES"	This context property section provides details of all applicable states of an asset order and the transition map for each action.
	Review this file in the 18D ABO Implementation Package for all defined state/action transition map. Only a subset is provided here for illustration.
	For example if state = "ACTIVE", the actions in the "VALID_ACTIONS" section are allowed. If a suspend action is invoked on an asset, the state will transition to a suspended state.
	Similarly, the actions in the "INVALID_ACTIONS" section are not allowed. When a user tries to invoke a resume action on an asset that is not suspended, an error message displays "You can only resume a suspended asset."
	`{`
	`  "ACTIVE": {`
	`  "DISPLAY": "Active",`
	`  "INVALID_ACTIONS": {`
	`  "RESUME": "You can only resume a suspended asset"`
	`  },`
	`  "VALID_ACTIONS": {`
	`  "MODIFY": {`
	`  "TARGET": "ACTIVE"`
	`  },`
	`  "SUSPEND": {`
	`  "TARGET": "SUSPENDED"`
	`  },`
	`  "TERMINATE": {`
	`  "TARGET": "INACTIVE"`
	`  },`
	`  "RENEW": {`
	`  "TARGET": "ACTIVE"`
	`  },`
	`  "UPDATE": {`
	`  "TARGET": "ACTIVE"`
	`  },`
	`  "NO_UPDATE": {`
	`  "TARGET": "ACTIVE"`
	`  },`
	`  "RECONFIG": {`
	`  "TARGET": "ACTIVE"`
	`  }`
	`  }`
	`  }`

CONTEXT PROPERTY	PROPERTY/VALUE
"deltaBomSvcSetting"	This context property section provides details of some of the properties that are applicable to the delta/apply functionality.
	`"generateAssetKey": true,` This property determines if the assetKey value is generated. The default setting is true and the assetKey is generated. If customers want to generate their own asset key, then they can set this property to false and override the abo_delta function to populate bomitem field itemInstance_l with the customized asset key.
	`"assetKeyPrefix":  "abo_",` This property identifies the prefix for the generated assetKey value, in the default delta process.

	The following properties determine the line document attributes used for storing asset keys, action codes, request dates, and attribute summary respectively.
	`"assetKeyField": "itemInstanceId_l",`
	`"actionField": "oRCL_ABO_ActionCode_l",`
	`"requestDateField": "requestDate_l",`
	`"attributeSummaryField": "oRCL_ABO_ComponentAttributes_l",`

	`"populateAttributeSummary": true,` This property determines if the attribute summary field is populated.
	`"StateAttributeName": oRCL_ABO_PriorAssetState",` This property determines the name of the attribute used to maintain projected state while implementing suspend or resume on child/grandchild models and parts. Refer to Enable Suspend and Resume for Child Operations for more details
	`"ingoreInvalidAction": false` This property determines if an error is thrown when an invalid action is encountered when the state/action transition map is validated
	`"commerceAttributeInDelta": "contractStartDate_l", "contractEndDate_l"]` This property determines additional commerce fields to use for comparison during the delta process.
"AboDiagnosticDisabled"	`true` When this context property set to false, it provides enhanced diagnostic information that can be used for troubleshooting. IMPORTANT: This property should be set to false only while troubleshooting critical issues; otherwise, it could lead to severe application performance issues.
"businessTimeZone"	`"GMT+0"` This optional context property sets a business-level time zone to be used in ABO flows to interpret dates and to override the server default time zone. The following requirements apply: • Use GMT+n or GMT-n for absolute time zone. • Use time zone name for DST Adjusted. For example: "America/Los Angeles" **Note**: Even though this property is available in abocontext in runtime, it is not part of defaultcontextjson.txt in the out of the box package. It is specified in "abo_loadDefaultContext" function directly with the default value of “GMT+0”. IMPORTANT: The business-level time zone feature was introduced in the 21A ABO Implementation package. Customization of older ABO packages can be implemented for this feature.

APPENDIX K: SAMPLE UPDATE ASSETS ACTION

Note: In Oracle CPQ 23B or later release, a new transaction action ***updateAsset*** is already included in the new commerce standard process definition.

You can use this sample Update Assets action to simulate the fulfillment process to convert a CPQ order to an asset. This action can be used for the following scenarios:

- Customers that do not need to track the fulfillment status within CPQ
- Training and fault isolation
- Customers that are not integrated with fulfillment systems such as Oracle CX Commerce (also known as Commerce Cloud)

Complete the following steps to implement the Update Assets action.

1. Log in to Oracle CPQ and open the Admin Home page.
2. Click **Process Definition**, in the Commerce and Documents section. The Processes page opens.
3. Select **Documents** from the Navigation drop-down menu for the applicable process (e.g. Oracle Quote to Order).
4. Click **List.** The Document List page opens.
5. Select **Actions** from the Navigation drop-down menu for the main document (e.g. Transaction).
6. Click **List**. The Action List page opens.
7. Click **Add** at the bottom of the Actions List page.
8. Enter the **Label**: Update Assets
9. Click inside the **Variable Name** field. The Variable Name field populates automatically.

   > **Note:** You can change the variable name before saving, but after saving, the value is read-only. Variable names can only contain alphanumeric characters and underscores.

10. Select **Modify** from the Action Type from the drop-down.
11. Click **Add** to save changes and open Admin Action editor.
12. Click the Define Advanced Modify – Before Formulas radio button.
13. Click **Define Function**. The Select Attributes page opens.
14. Select the "`_system_buyside_id`" variable from the System Variable Name tab.
15. Select the following variables from the Variable Name for (Transaction) tab:

    "`currency_t`", "`paymentTerms_t`", and "`_transaction_customer_id`"

16. Select the following variables from the Variable Name for (Transaction Line) tab:

    "`_document_number`", "`itemInstanceId_l`", "`fulfillmentStatus_l`", "`requestDate_l`", "`_line_bom_parent_id`", and "`oRCL_ABO_ActionCode_l`"

17. Select the following **ORCL_ABO Util Library Functions** from the Library Functions tab:

    "`abo_initializeContext`" and "`abo_updateAsset`"

18. Click **Next**. The BML Editor opens.
    19. Copy and paste the Sample Update Assets Script into the BML window.
20. Click Save and Close.
21. Click **Update**.

> **Note:** To make the Update Assets action available on the Transaction UI, refer to Add ABO Items to the Commerce Layout.

```
//Sample Update Assets Script
commerceProcess="oraclecpqo";
abocontext = util._ORCL_ABO.abo_initializeContext(commerceProcess);
bDiagnosisOff=jsonget(abocontext, "AboDiagnosticDisabled", "boolean", true);
callContext="updateAsset";

customer_id = _transaction_customer_id;
currency = currency_t;
paymentTerm = paymentTerms_t;

FULFILLED = "FULFILLED";
DOCUMENT_NUMBER  = "documentNumber";
CUSTOMER = "customer";
TRANSACTION_ID = "transactionId";
CURRENCY_CODE = "currency";
REQUEST_DATE =  "requestDate";
ACTION_CODE =  "actionCode";
ITEM_INSTANCE_ID = "itemInstanceId";
PAYMENT_TERM = "paymentTerms";

if(customer_id == "" OR isnull(customer_id)){
   throwError("Please select a customer.");
}

txn_json = json();
jsonput(txn_json, CUSTOMER, customer_id);
jsonput(txn_json, CURRENCY_CODE, currency);
jsonput(txn_json, PAYMENT_TERM, paymentTerm);
jsonput(txn_json, TRANSACTION_ID, _system_buyside_id);

successString ="";
lineJsonArray = jsonArray();
for line in transactionLine{
  if(line.fulfillmentStatus_l == "FULFILLED" OR line.fulfillmentStatus_l == "CANCELLED"){
    continue; //skip lines already fulfilled or cancelled.
  }
  if(line._line_bom_parent_id <>"") { //skip non-root line
    continue;
  }
  if(line.itemInstanceId_l=="") { //skip non abo & non model line
    continue;
  }

  lineJson =  json();

  jsonput(lineJson, DOCUMENT_NUMBER, line._document_number);

  transactionDate = line.requestDate_l;
  if(transactionDate == "" OR isnull(transactionDate)){
     transactionDate = datetostr(getDate(false), "yyyy-MM-dd HH:mm:ss");
  }else{
     tranDate =  strtojavadate(transactionDate, "MM/dd/yyyy HH:mm:ss");
     transactionDate =  datetostr(tranDate, "yyyy-MM-dd HH:mm:ss");
  }
  jsonput(lineJson, REQUEST_DATE, transactionDate);
  jsonput(lineJson, ACTION_CODE, line.oRCL_ABO_ActionCode_l);
  jsonput(lineJson, ITEM_INSTANCE_ID, line.itemInstanceId_l);
  jsonarrayappend(linejsonArray, lineJson);

  if (successString<> ""){
    successString =  successString + "|";
  }
  successString = successString +  line._document_number + "~fulfillmentStatus_l~"+FULFILLED;
}

response = util._ORCL_ABO.abo_updateAsset(txn_json, lineJsonArray);

return successString;
```

APPENDIX L: PRESERVE NEW TRANSACTION LINE ATTRIBUTE VALUES IN ASSETS

Complete the following steps to preserve new transaction line attribute values in assets.

1. Navigate to Admin > Products > Assets.
2. Add a new custom attribute for assets, and then deploy the changes.

   > For additional information, refer to Asset REST APIs in Oracle CPQ Administrator Online Help Commerce > Asset-Based Ordering > Custom Asset Fields.

   This new custom asset attribute can be used to store the value of the new transaction line attribute in an asset.
3. Navigate to: Admin > BML Library > ORCL_ABO folder.
4. Perform one of the following options for the `abo_getOpenOrderBoms` function:
   a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
   b. If the function has already been overridden, click on Edit.
5. Search for the "`lineField`" array and add a new array with the variable name of the new transaction line attribute.
6. Save the "`abo_getOpenOrderBoms`" library function.

If you are using 18D or later package, complete the following:

1. Navigate to: Admin > File Manager > ABO > defaultContextJson.txt
2. Add the variable name of the new custom asset attribute to the "`assetInfo.fieldQueryList`" property.
   a. Ensure that you add the custom asset attribute twice. For example – Refer to "*assetKey*" that is added twice in "`assetInfo.fieldQueryList`" property as follows:
      i. *"fieldQueryList":"assetKey …… descendantAssets.assetKey"*
3. Add the variable name of the new transaction line attribute to "`extraLineFieldForAsset`" property.
4. Navigate to: Admin > BML Library > ORCL_ABO.
5. Perform one of the following options for the `abo_convertDeltaBomtoAsset` or `abo_convertDeltaBomItemToAssetItem_ext` function:
   a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
   b. If the function has already been overridden, click on Edit.

   Include the logic to copy from transaction line to asset. Refer to how `displayKey` in `assetPayload` is populated from `itemInstanceName_l` in `fieldJson` and add similar changes to store the new transaction attribute value in the new asset custom attribute.

   For example - Assuming new transaction line item attribute is a text attribute, the following lines will copy the new transaction line item attribute value to the new custom asset attribute.

```
transAttrValue = jsonget(fieldJson, NEW_TRANSACTION_ATTR_VARIABLE_NAME , "string", "");
if (transAttrValue <> "") {
      jsonput(assetPayLoad, NEW_ASSET_CUSTOM_ATTR_VARIABLE_NAME, transAttrValue);
}
```

   `NEW_TRANSACTION_ATTR_VARIABLE_NAME` and `NEW_ASSET_CUSTOM_ATTR_VARIABLE_NAME` are referring to constants that should be declared in the constant declaration section for the new transaction attribute variable name and new asset custom attribute variable name respectively.
6. Save the "`abo_convertDeltaBomtoAsset`" or "`abo_convertDeltaBomItemToAssetItem_ext`" library function.
7. Perform one of the following options for the `abo_loadAsset` function:
   a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
   b. If the function has already been overridden, click on Edit.
8. Locate the loop to convert asset to BOM.

   Copy the field value from the new asset custom attribute to the corresponding transaction line attribute within the "fields" JSON. Refer to sample code where `displayKey` field from `oneAssetNode` is copied to `itemInstanceName_l` within "`fields`" JSON in existing code.

9. Save the "`abo_loadAsset`" library function. Navigate to: Admin > BML Library > ORCL_ABO, override the BML function, *"abo_reconfigure"*. Search for the *"linefield"* array and add a new array element with the variable name of the new transaction line attribute. Save the library function.
10. Deploy all the modified library functions.

If you are using 18C or earlier package, then do the following:

1. Navigate to: Admin > BML Library > ORCL_ABO and perform one of the following options for the `abo_initializeContext` function:
    a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
    b. If the function has already been overridden, click on Edit.
2. Locate the block to initialize "`extraLineFieldsforAsset`" JSON field (around line# 98 to 103).
3. Add the variable name of the new transaction line attribute in the "`fieldList`" along with the other attributes.
4. Save the "`abo_initializeContext`" library function.
5. Perform one of the following options for the `abo_convertDeltaBomtoAsset` function:
    a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
    b. If the function has already been overridden, click on Edit.
6. Add logic to copy transaction line attribute value from BOM fields (`fieldJson`) to `assetPayLoad` JSON. Refer to how `displayKey` in `assetPayload` is populated `from itemInstanceName_l` in `fieldJson` and do similar changes to store the new transaction attribute value in the new asset custom attribute.
7. Save the "`abo_convertDeltaBomtoAsset`" library function.
8. Perform one of the following options for the `abo_loadAsset` function:
    a. If the function hasn't been previously overridden, click on the Create link under the Override Column.
    b. If the function has already been overridden, click on Edit.
9. Locate the `fieldArr` declaration (around line# 18).
10. In the loop to convert asset to BOM, copy the field value from the new asset custom attribute to the corresponding transaction line attribute within the "`fields`" JSON.

    > **Note:** This will have to be done for both root and descendant assets.
    > Refer to how `assetKey` has been defined for both root and descendant assets.

11. Save the "`abo_loadAsset`" library function.

CONNECT WITH US

Call +1.800.ORACLE1 or visit oracle.com.
Outside North America, find your local office at oracle.com/contact.

 blogs.oracle.com  facebook.com/oracle  twitter.com/oracle