CPQ Cloud BOM Mapping

CPQ Cloud Release 18B

Implementation Guide

TABLE OF CONTENTS

REVISION HISTORY

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

Date	What's Changed	Notes
JUNE 2018		Initial Document Creation

INTRODUCTION

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

The BOM Mapping feature delivers the ability to:

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

IMPORTANT:

- BOM Mapping is used in Subscription Ordering and System Configuration, and is a prerequisite for using those features.
- Implementing BOM Mapping requires advanced knowledge of CPQ Configuration and fulfillment system configuration models.

PURPOSE

The CPQ Cloud BOM Mapping Implementation Guide describes BOM Mapping functionality and provides instructions for implementing BOM Mapping.

PREREQUISITES

BOM Mapping Implementation is available on CPQ Cloud sites on 2016 R1, or later. Some features described in this guide are only available in 18B, or later.

SCOPE

This implementation guide is divided into the following sections:

- BOM Mapping Overview - describes BOM Mapping functionality and BOM Mapping Use Cases.
- BOM Mapping Configuration Rules - explains BOM Mapping Configuration rules and provides information about rule execution.
- BOM Instance - describes the BOM Instance and saving a BOM Instance to Commerce transactions and quotes.
- Implementing BOM Mapping - provides instructions for implementing and validating BOM Mapping.
- Reference Information - describes BOM Mapping System Attributes, Table Schema, BML functions, and JSON objects.

BOM MAPPING OVERVIEW

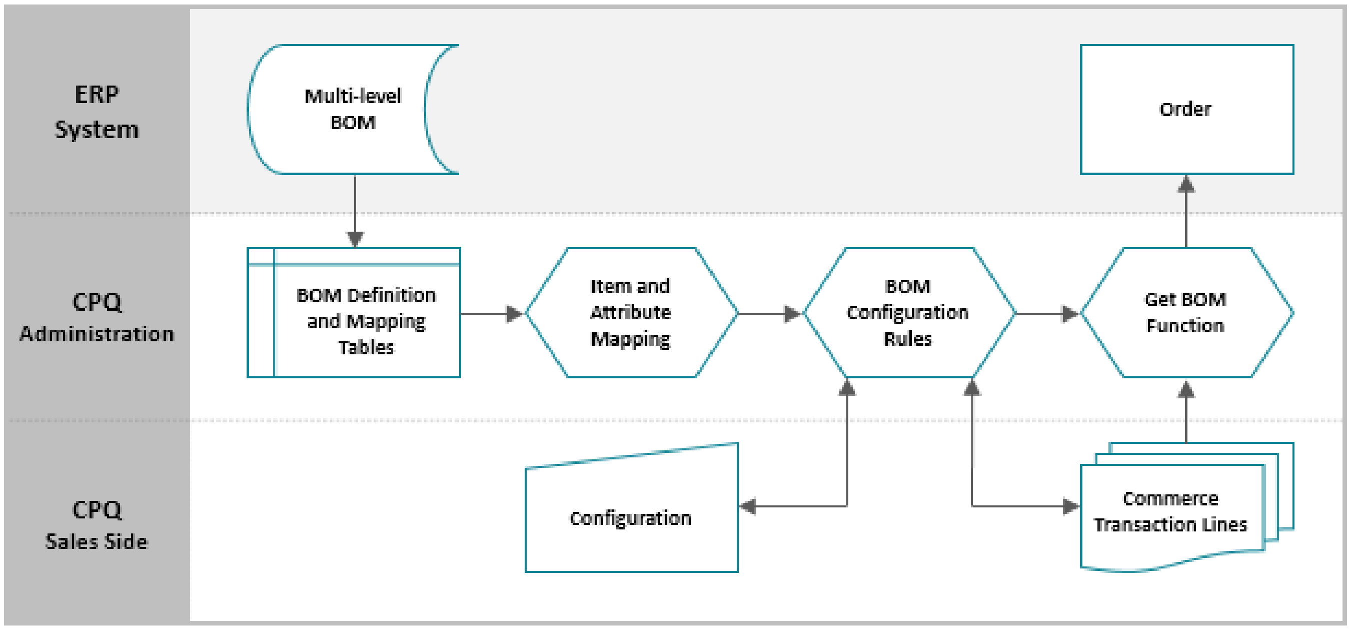
Fulfillment systems often maintain bills of material (BOMs) containing complex, multi-level part structures that differ from the Configuration attributes used in CPQ Cloud when sales users configure products. CPQ Cloud provides an external, sales-oriented, guided selling view of products that focuses on the customer's needs and perceived value for the product. In contrast, fulfillment and ERP systems provide an internal, manufacturing or production oriented view of products. An example of this difference can be seen in eCommerce sites selling computer laptops.

- In CPQ Cloud, a customer may initially be asked, "What do you want to do with your laptop?" and presented with different options (i.e. business, gaming, video, and media). The selected option will result in the user being asked different additional questions concluding with a laptop configuration bundle that will meet their needs.
- In a fulfillment system, the guide that is used to design and assemble a laptop typically contains hundreds of parts and options that are crucial to assembling and sourcing the product components, but are not meaningful to the customer.

The BOM Mapping feature provides a data-driven mechanism for mapping these differing product views.

As illustrated in the image below, BOM Mapping enables administrators to associate their fulfillment system BOMs to an Oracle CPQ Cloud Configuration. The product structures are stored in BOM definition tables. Administrators map tables and setup BOM Configuration rules to link the BOM definitions to the CPQ Cloud Configuration attribute selections.

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



BOM Mapping Overview

HIERARCHICAL RELATIONSHIPS IN BOM MAPPING

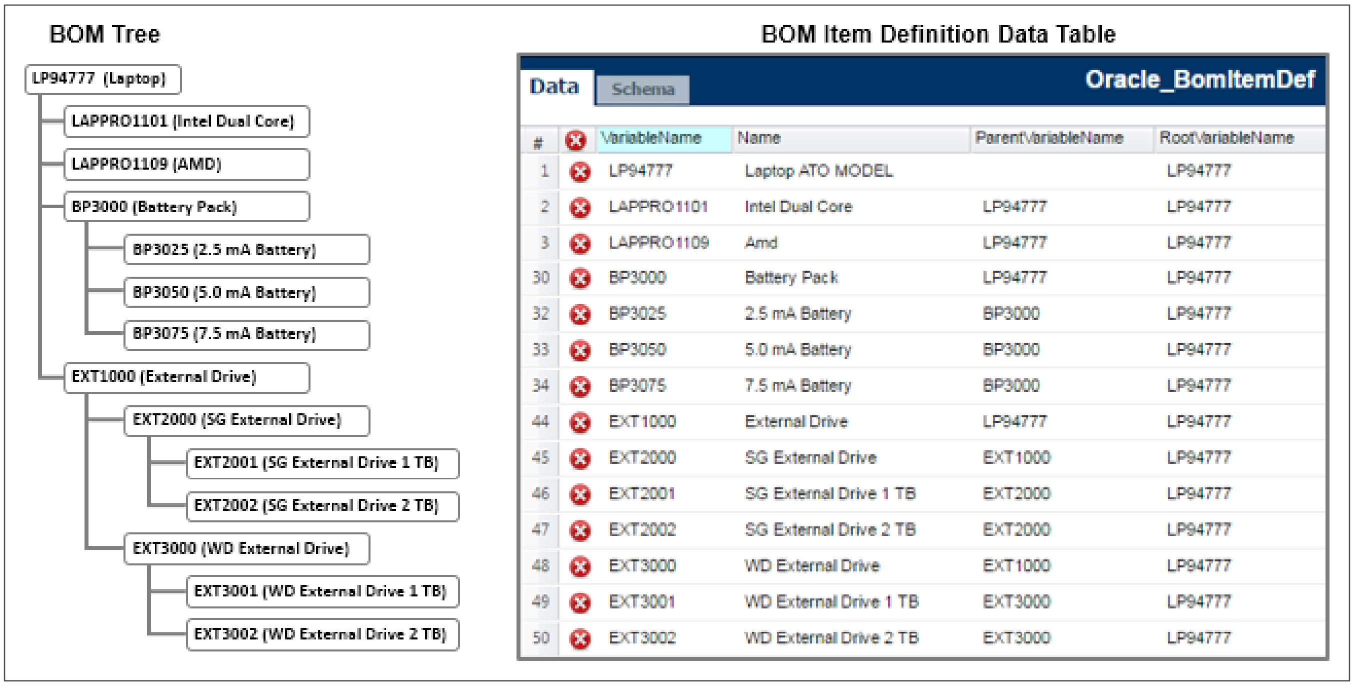
A central component of the BOM Mapping feature is the capture of BOM product structure hierarchies in CPQ Cloud for reference by the BOM Mapping rules. The following example illustrates how these hierarchical parent-child relationships are stored.

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

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

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

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


BOM Tree
LP94777 (Laptop)
LAPPRO1101 (Intel Dual Core)
LAPPRO1109 (AMD)
BP3000 (Battery Pack)
BP3025 (2.5 mA Battery)
BP3050 (5.0 mA Battery)
BP3075 (7.5 mA Battery)
EXT1000 (External Drive)
EXT2000 (SG External Drive)
EXT2001 (SG External Drive 1 TB)
EXT2002 (SG External Drive 2 TB)
EXT3000 (WD External Drive)
EXT3001 (WD External Drive 1 TB)
EXT3002 (WD External Drive 2 TB)
BOM Item Definition Data Table
Data
Schema
Oracle_BomItemDef

#	✖	VariableName	Name	ParentVariableName	RootVariableName
1	✖	LP94777	Laptop ATO MODEL		LP94777
2	✖	LAPPRO1101	Intel Dual Core	LP94777	LP94777
3	✖	LAPPRO1109	Amd	LP94777	LP94777
30	✖	BP3000	Battery Pack	LP94777	LP94777
32	✖	BP3025	2.5 mA Battery	BP3000	LP94777
33	✖	BP3050	5.0 mA Battery	BP3000	LP94777
34	✖	BP3075	7.5 mA Battery	BP3000	LP94777
44	✖	EXT1000	External Drive	LP94777	LP94777
45	✖	EXT2000	SG External Drive	EXT1000	LP94777
46	✖	EXT2001	SG External Drive 1 TB	EXT2000	LP94777
47	✖	EXT2002	SG External Drive 2 TB	EXT2000	LP94777
48	✖	EXT3000	WD External Drive	EXT1000	LP94777
49	✖	EXT3001	WD External Drive 1 TB	EXT3000	LP94777
50	✖	EXT3002	WD External Drive 2 TB	EXT3000	LP94777

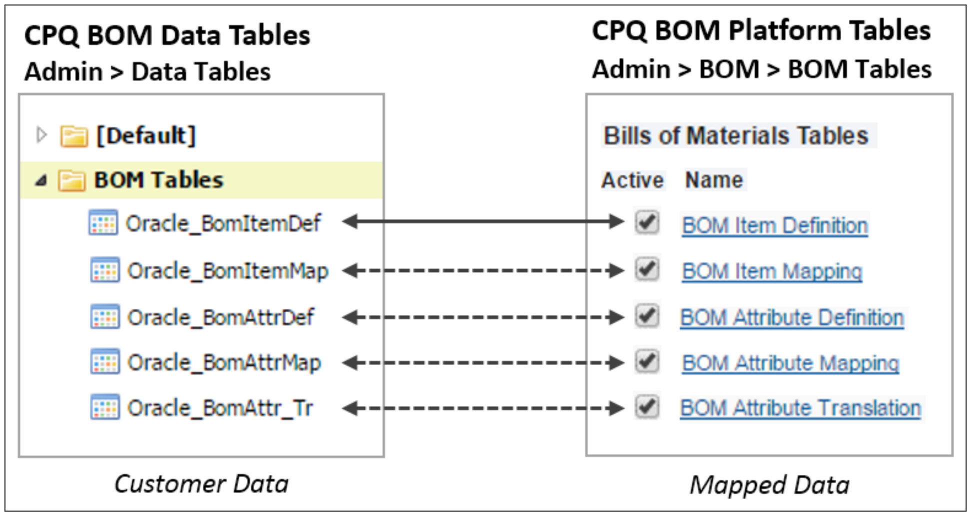
BOM Tree Example

BOM MAPPING TABLES

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

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

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



BOM Data Table to BOM Platform Table Mapping

CPQ Cloud provides standard, downloadable Data Table definitions that can be used to create the implementation-specific tables, which map automatically to the BOM Mapping platform tables. Alternatively, customers can reuse existing Data Tables containing BOM structure details by mapping the columns in their tables to the BOM Mapping platform tables. The new Edit BOM Table Definition page provides drop-down menus for column mapping and Data Table definitions for download.

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

BOM ITEM DEFINITION TABLE

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

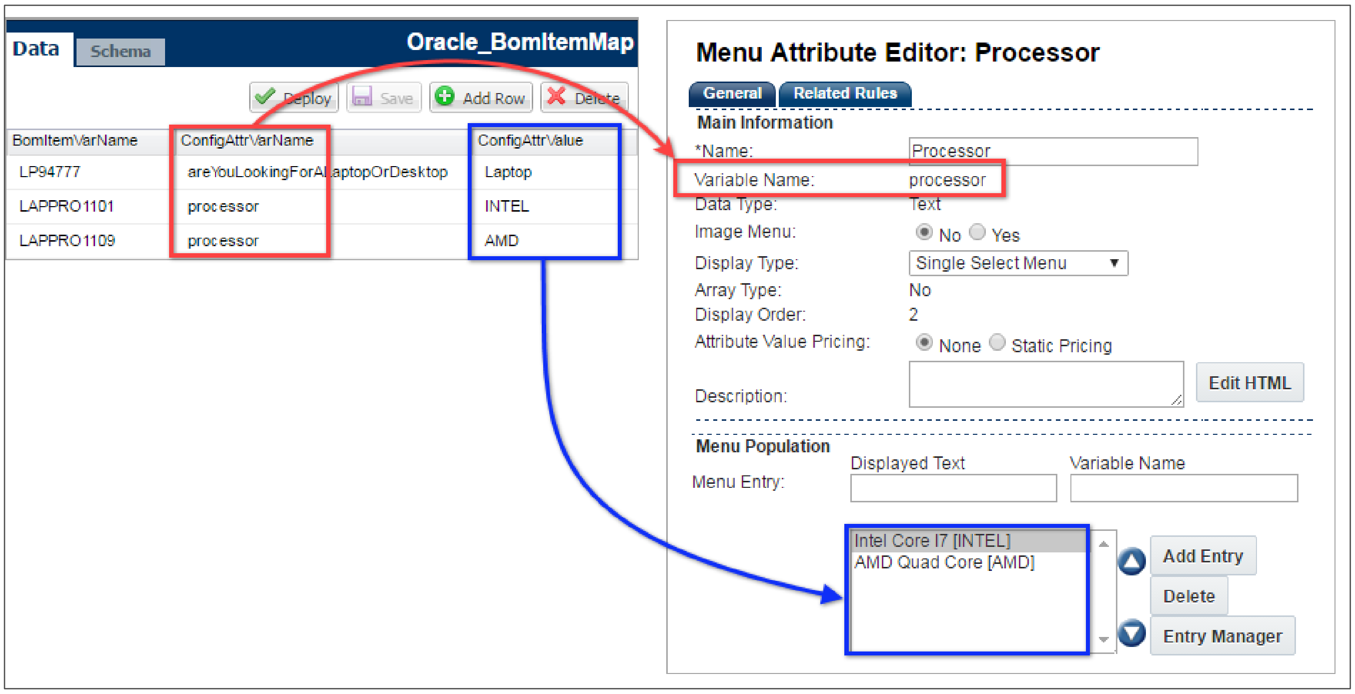
- Fulfillment system IDs
- Default quantity
- Whether an item is optional, a sales item, or a manufacturing item
- BOM item effective dates

Refer to the BOM Item Definition Table Schema for an explanation of all table fields.

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

BOM ITEM MAPPING TABLE

The BOM Item Mapping Table associates BOM items to Configuration attributes. Activating this table enables simple Table-Based BOM Mapping Configuration rules. If this table is not active, administrators would use advance BML-Based Rules to establish the association between BOM items and Configuration attributes. **Error! Reference source not found.** shows the how the BOM Item Mapping Data Table “ConfigAttrVarName” and “ConfigAttrValue” items relate to the Configuration attributes.



BomItemVarName	ConfigAttrVarName	ConfigAttrValue
LP94777	areYouLookingForALaptopOrDesktop	Laptop
LAPPRO1101	processor	INTEL
LAPPRO1109	processor	AMD

Menu Attribute Editor: Processor

*Name:	Processor
Variable Name:	processor
Data Type:	Text
Image Menu:	[✓ No] Yes
Display Type:	Single Select Menu
Array Type:	No
Display Order:	2
Attribute Value Pricing:	[✓ None] Static Pricing
Description:	

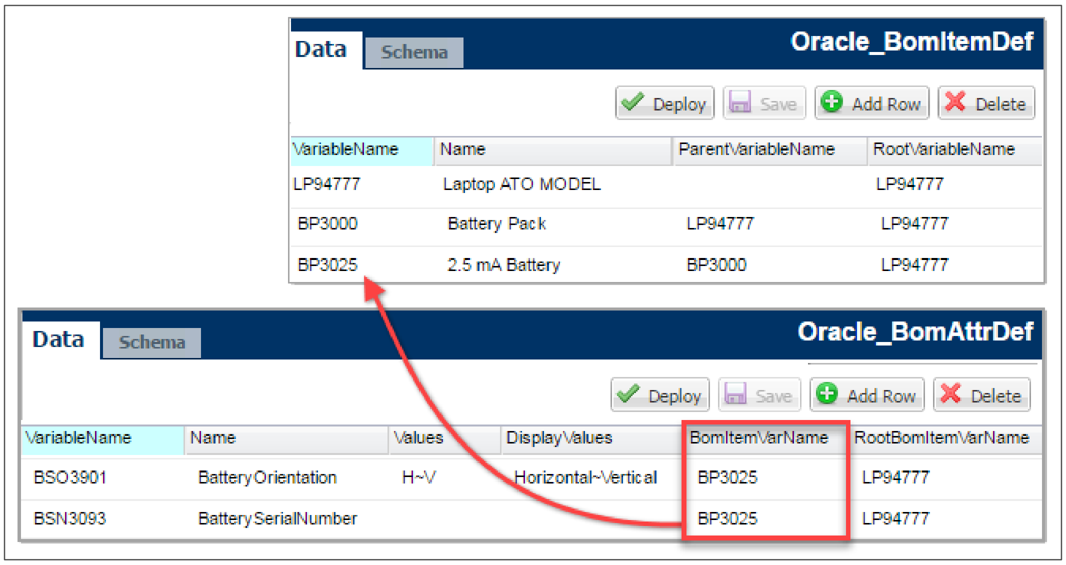
Intel Core I7 [INTEL]
AMD Quad Core [AMD]

BOM Item Mapping Example

Refer to the BOM item Mapping Table Schema for an explanation of all table fields.

BOM ATTRIBUTE DEFINITION TABLE

The BOM Attribute Definition Table stores attribute definitions and attribute effective dates from the fulfillment system. These attributes can define options, such as color or size. BOM item variable references associate the fulfillment system attributes with the applicable BOM items. For Example: In the following image, BOM attributes “BSO3901” and “BSN3903” reference “BP3025” in the “BomItemVarName” field. This field references the “VariableName” field in the BOM Item Definition table, which establishes the association of BOM attributes to BOM items.



Oracle_BomItemDef

VariableName	Name	ParentVariableName	RootVariableName
LP94777	Laptop ATO MODEL		LP94777
BP3000	Battery Pack	LP94777	LP94777
BP3025	2.5 mA Battery	BP3000	LP94777

Oracle_BomAttrDef

VariableName	Name	Values	DisplayValues	BomItemVarName	RootBomItemVarName
BSO3901	BatteryOrientation	H~V	Horizontal~Vertical	BP3025	LP94777
BSN3093	BatterySerialNumber			BP3025	LP94777

BOM Attribute Definition Example

Refer to the BOM Attribute Definition Table Schema for an explanation of all table fields.

BOM ATTRIBUTE MAPPING TABLE

The BOM Attribute Mapping Table stores associations between BOM attributes, Configuration attributes, Commerce transaction line attributes, and quantity values. Setting up the BOM Attribute Mapping table enables simple Table-based BOM Mapping Configuration rules to be used to associate BOM attributes to CPQ Configuration and Commerce items.

The BOM Attribute Mapping table identifies describes BOM Attribute Mapping Target Types.

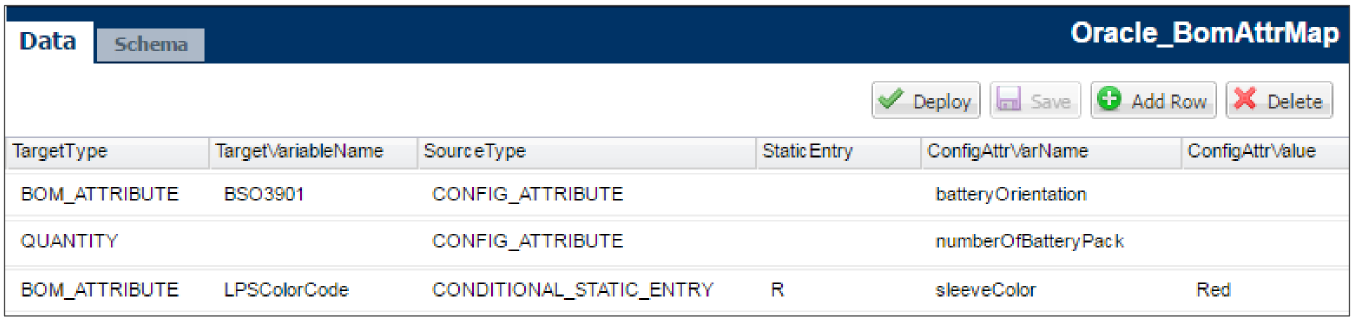
- **BOM_ATTRIBUTE** - Maps to a BOM attribute. Attributes that are not defined in the BOM Attribute Definition are ad hoc BOM attributes.
- **LINE_ATTRIBUTE** - Maps to a Commerce line attribute. The line attribute must exist in the target Commerce process.
- **QUANTITY** - Maps to the BOM line item quantity.

The Source Type field sets the source behavior, and there are three types:

- **STATIC_ENTRY** - The target is always set to the value in the source Static Entry.
- **CONDITIONAL_STATIC_ENTRY** - The target is set to the value in the Static Entry if the selected Configuration attribute matches the Configuration attribute value.
- **CONFIG_ATTRIBUTE** - The target is set to the value of the selected Configuration attribute.

Refer to the image below for the following examples:

- The first line sets the "BSO3901" BOM attribute value to the value selected for the Configuration "batteryOrientation" attribute.
- The second line entry sets the quantity to the value entered for the Configuration "numberofBatteryPack" attribute.
- The third line sets the "LPSColorCode" BOM attribute value to "R", if the selected Configuration "sleeveColor" attribute value is "Red".



TargetType	TargetVariableName	SourceType	Static Entry	ConfigAttrVarName	ConfigAttrValue
BOM_ATTRIBUTE	BSO3901	CONFIG_ATTRIBUTE		batteryOrientation	
QUANTITY		CONFIG_ATTRIBUTE		numberOfBatteryPack	
BOM_ATTRIBUTE	LPSColorCode	CONDITIONAL_STATIC_ENTRY	R	sleeveColor	Red

BOM Attribute Mapping Example

BEST PRACTICE RECOMMENDATION: Maintain a 1-to-1 mapping between a Target line or BOM Attribute value and a Source Configuration attribute value.

Refer to the BOM Attribute Mapping Table Schema for an explanation of all table fields.

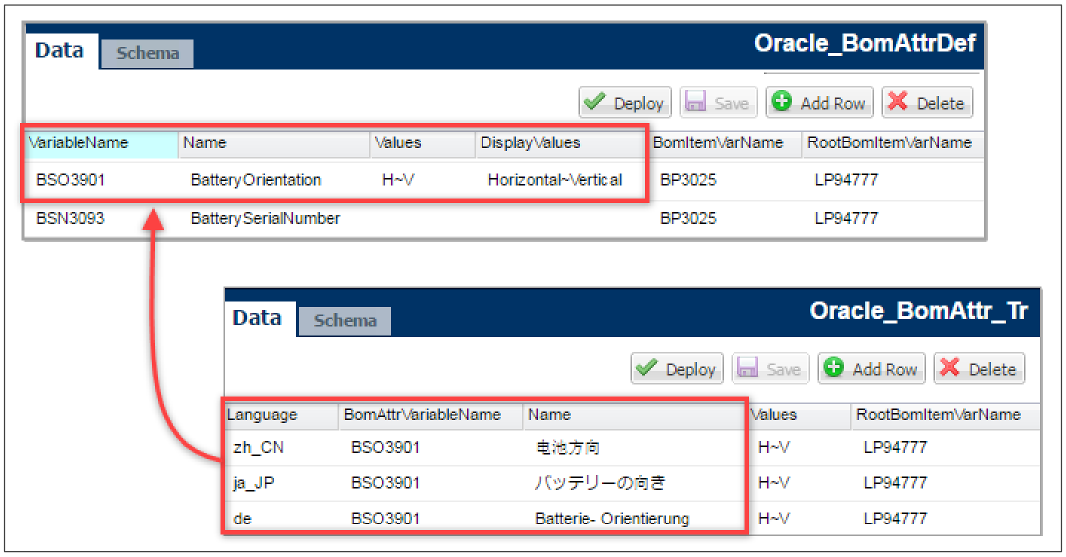
BOM ATTRIBUTE TRANSLATION TABLE

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

IMPORTANT:

- Only attribute mappings with BOM_ATTRIBUTE as the target type are translated.
- All translations are included in JSON files, not just the user's preferred language.

As shown in the following image, the BOM Attribute Translation table provides three translations for BOM Attribute variable “BSO3901”: zh_CN – Chinese (Simplified), ja_JP - Japanese, and de – German.



VariableName	Name	Values	DisplayValues	BomItemVarName	RootBomItemVarName
BSO3901	BatteryOrientation	H~V	Horizontal~Vertical	BP3025	LP94777
BSN3093	BatterySerialNumber			BP3025	LP94777

Language	BomAttrVariableName	Name	Values	RootBomItemVarName
zh_CN	BSO3901	电池方向	H~V	LP94777
ja_JP	BSO3901	バッテリーの向き	H~V	LP94777
de	BSO3901	Batterie- Orientierung	H~V	LP94777

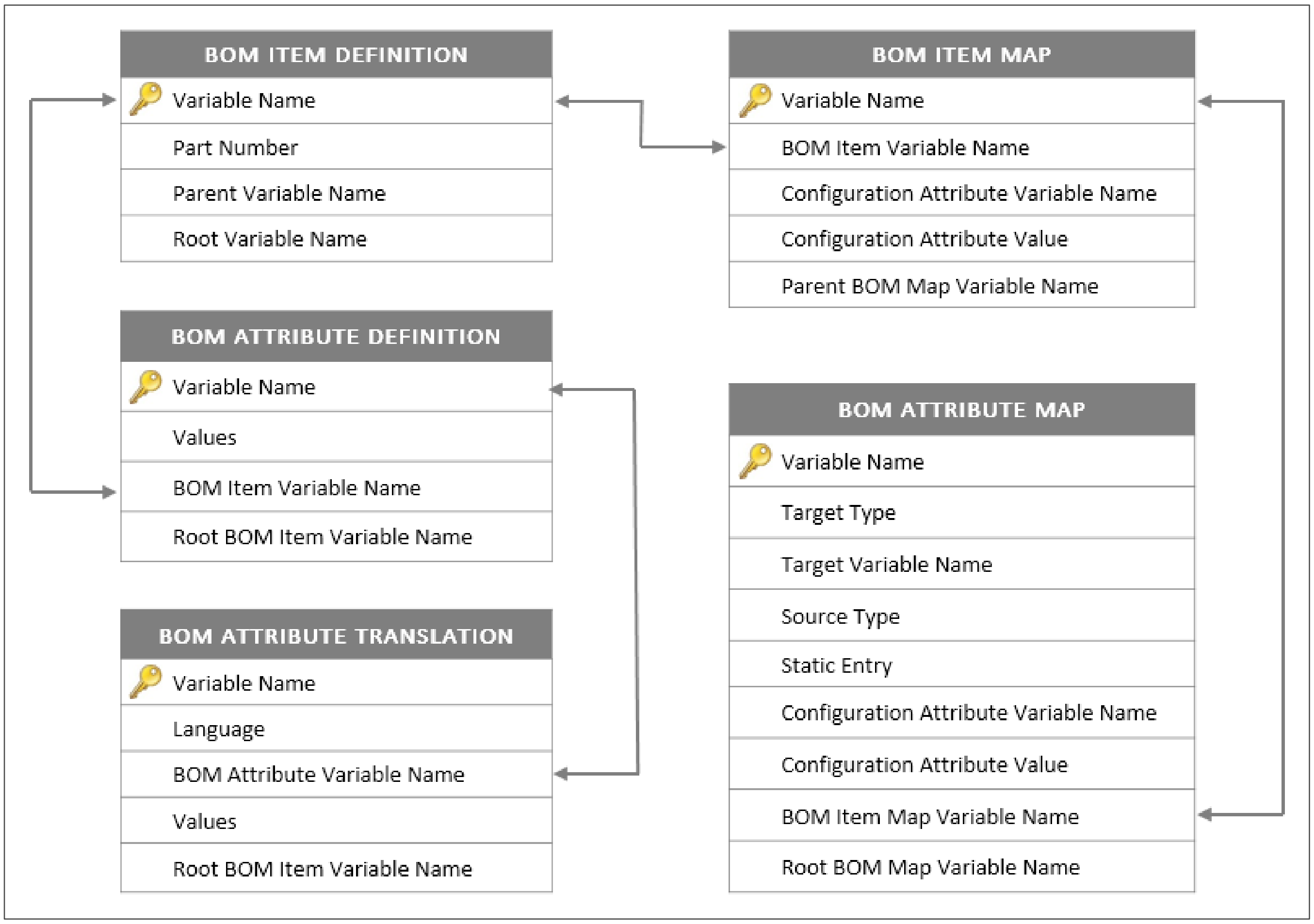
BOM Attribute Translation Example

BEST PRACTICE RECOMMENDATION: When providing attribute translations, Single Select Menu and Multi-Select Menu attributes should be used. Use of Text field attributes should be avoided.

Refer to the BOM Attribute Translation Table Schema for an explanation of all table fields.

BOM TABLE RELATIONSHIPS

BOM Mapping uses variable names as references to capture hierarchical relationships. BOM Mapping also uses variable names to identify relationships between BOM tables. The following image illustrates these relationships.


BOM ITEM DEFINITION
Variable Name
Part Number
Parent Variable Name
Root Variable Name
BOM ITEM MAP
Variable Name
BOM Item Variable Name
Configuration Attribute Variable Name
Configuration Attribute Value
Parent BOM Map Variable Name
BOM ATTRIBUTE DEFINITION
Variable Name
Values
BOM Item Variable Name
Root BOM Item Variable Name
BOM ATTRIBUTE MAP
Variable Name
Target Type
Target Variable Name
Source Type
Static Entry
Configuration Attribute Variable Name
Configuration Attribute Value
BOM Item Map Variable Name
Root BOM Map Variable Name
BOM ATTRIBUTE TRANSLATION
Variable Name
Language
BOM Attribute Variable Name
Values
Root BOM Item Variable Name

BOM Table Relationships

BOM MAPPING USE CASES

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

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

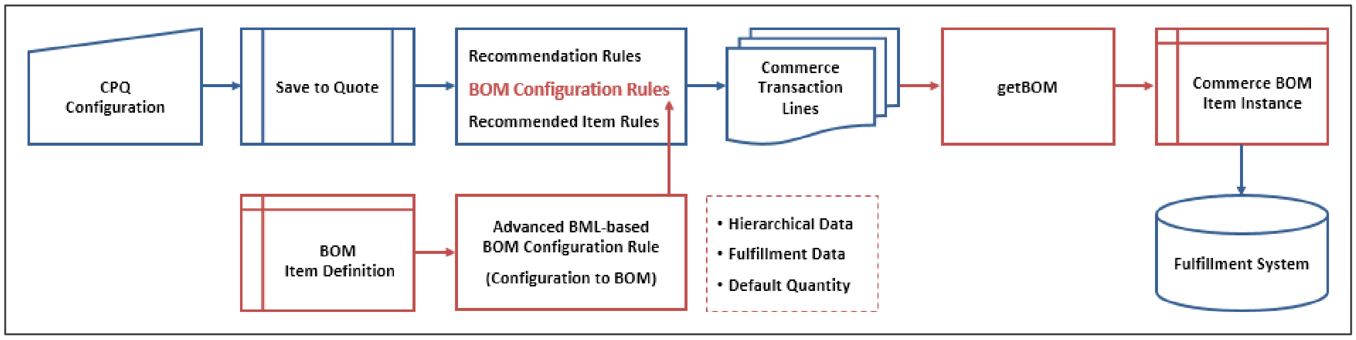
BOM MAPPING FUNCTIONAL OVERVIEW

BOM Mapping Configuration rules associate BOM items to Configuration attributes and Commerce lines. This section explains sales configuration and reconfiguration process flows with BOM Mapping. Examples for basic and full service implementations are addressed.

BASIC BOM MAPPING PROCESS FLOWS

The following examples cover the process flows for a basic BOM Mapping implementation, which only uses the BOM Definition Table. The following description explains the configuration process flow.

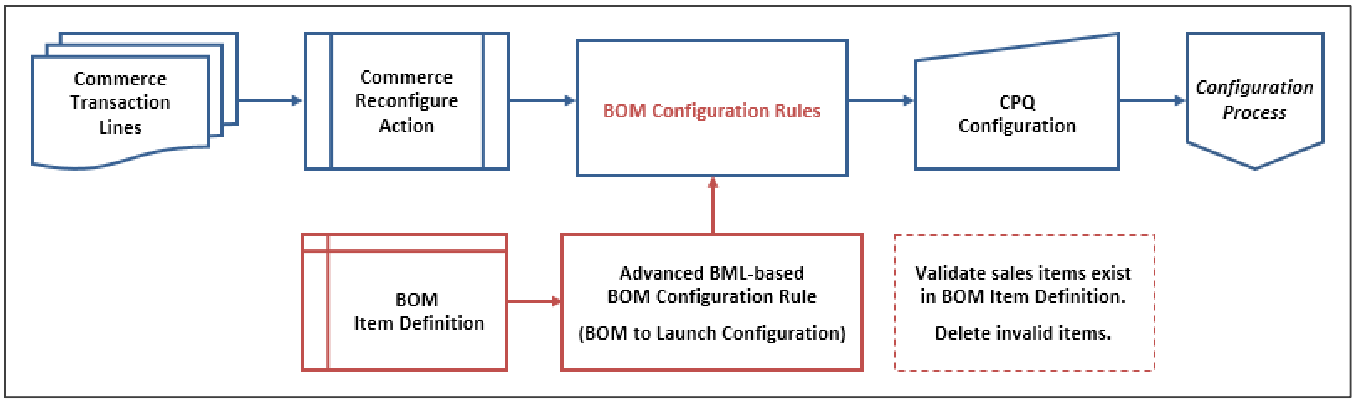
1. Configuration rules actuate when a user saves a sales configuration to a quote.
2. The advanced BML-based BOM Configuration Rule references the BOM Item Definition table to accomplish the following tasks:
    - Record BOM hierarchical data (parent-child relationships)
    - Identify BOM item quantity
3. The configuration, BOM part numbers, quantities, and parent-child relationships are saved to Commerce Transaction Lines.
4. Administrators use the new getBOM BML function to extract the Sales or Manufacturing BOM. These BOM instances include BOM item variable names and BOM fulfillment data (BOM item ID, type, etc.).
5. The BOM, including order information, is sent to the Fulfillment System, to complete the order.



Basic BOM Mapping Configuration Process Flow

The following example covers the reconfiguration process flow for a basic BOM mapping implementation.

1. Configuration rules actuate when a user reconfigures a transaction line.
2. The CPQ Configuration attribute values are set to reflect the transaction line. Advanced BML-based Configuration Rules validate sales items still exist in the BOM Item Definition table. Invalid BOM items are deleted.
3. After reconfiguration, the order follows the Configuration Process.


Commerce Transaction Lines
Commerce Reconfigure Action
BOM Configuration Rules
CPQ Configuration
Configuration Process
BOM Item Definition
Advanced BML-based BOM Configuration Rule
(BOM to Launch Configuration)
Validate sales items exist in BOM Item Definition.
Delete invalid items.

Basic BOM Mapping Reconfiguration Process Flow

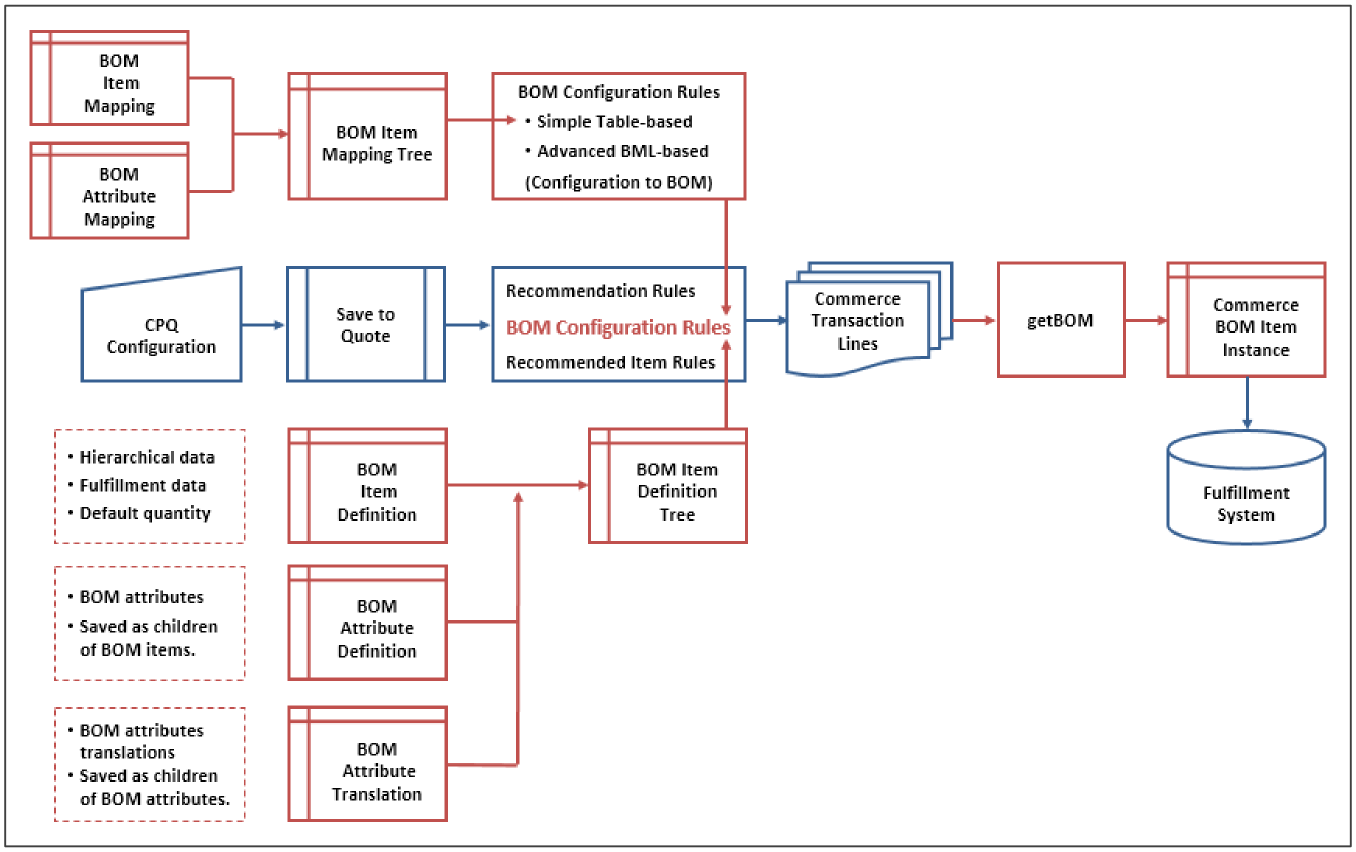
FULL-SERVICE BOM MAPPING PROCESS FLOWS

The next two examples cover full-service BOM Mapping implementations. Full-service implementations use all BOM mapping tables. The first example covers the configuration process flow for full-service BOM mapping.

1. Configuration rules actuate when a user reconfigures a sales quote.

IMPORTANT: Using BOM mapping tables enables the simple table-based configuration rule.

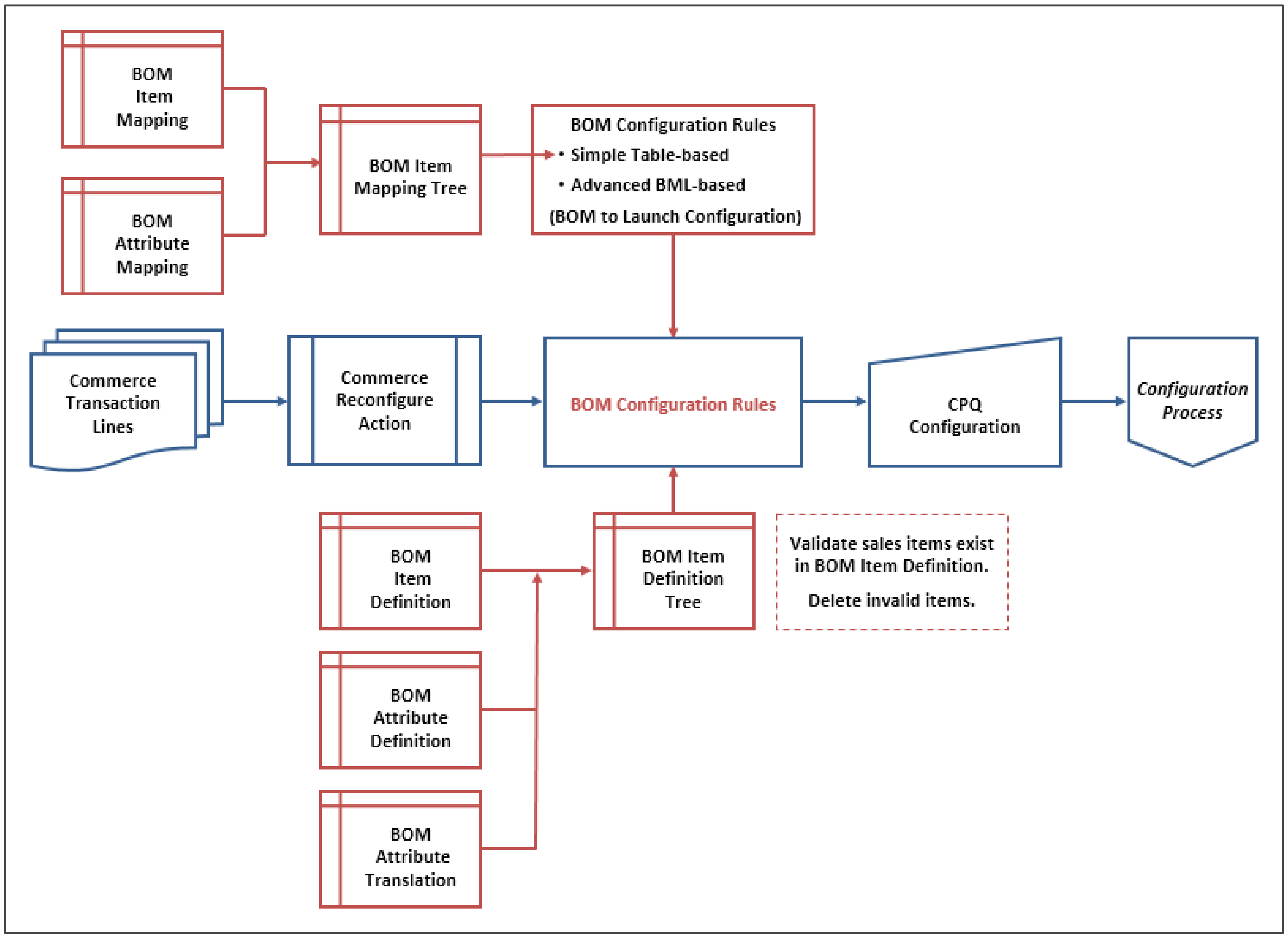
2. The simple table-based BOM Configuration Rule references the BOM item definition. Administrators can also use advanced BML-based to further refine the input to configuration. BOM Mapping configuration rules accomplish the following tasks:
    - Record BOM hierarchical data (parent-child relationships)
    - Identify BOM item quantity
3. The configuration, BOM part numbers, quantities, and parent-child relationships are saved to Commerce Transaction Lines.
4. Administrators use the new getBOM BML function to extract the Sales or Manufacturing BOM. These BOM instances include BOM item variable names and BOM fulfillment data (BOM item ID, type, etc.).
5. The BOM, including order information, is sent to the Fulfillment System, to complete the order.



Full-Service BOM Mapping Configuration Process Flow

The same reversed sequence described for the basic implementation, will occur for the full-service implementation. This example covers the reconfiguration process flow for a full-service BOM mapping implementation.

1. Configuration rules actuate when a user reconfigures a sales quote.
2. The CPQ Configuration attribute values are set to reflect the transaction line. Simple table-based Configuration Rules reference the BOM Item Tree table to validate sales items still exist in the BOM Item Definition table. Invalid BOM items are deleted.
3. After reconfiguration, the order follows the Configuration Process.



Full-Service Reconfiguration Process Flow

BOM MAPPING CONFIGURATION RULES

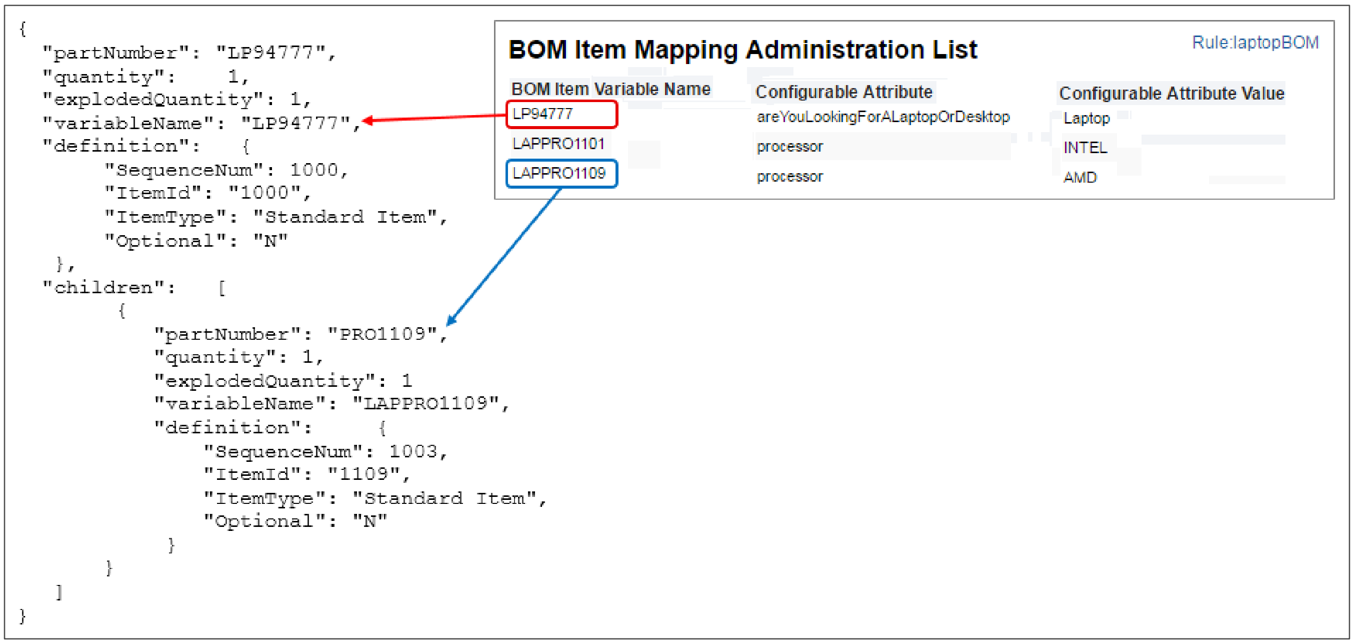
BOM Mapping Configuration rules associate BOM items and attributes to CPQ Configuration attributes and Commerce line items. Simple Table-based Configuration rules are defined using the BOM Item Mapping Data Table and optionally the BOM Attribute Mapping Data Table. Advanced BML-based rules can then be used in addition to the Table-based rules to refine the mapping results. In rare cases, BOM Item Mapping tables are not used at all, and administrators use advanced BML-based Rules to establish Configuration and Commerce association with BOM items. If BOM Mapping tables are not used, administrators can use advance BML-based Rules to establish Configuration and Commerce association. Advanced BML-based rules can be used in addition to the Table-based rules to refine the mapping results.

BOM ITEM CONFIGURATION RULES

CONFIGURATION BOM ITEM MAPPING

BOM item mapping associates BOM items to Configuration attributes. During Configuration, whenever the configuration state is changed, the BOM Mapping Configuration rules are invoked to map Configuration attributes to BOM items. The following image illustrates BOM item and Configuration attribute associations. For Example: When the "areYouLookingForALaptopOrDesktop" Configuration attribute is "Laptop", the root BOM item with the Variable Name "LP94777" is created.

At the next child level, if the "processor" configuration attribute is "INTEL", the child BOM item "LAPPRO1101" is created. If it is "AMD", the child BOM item "LAPPRO1109" is created. The child BOM item is created under the root BOM item. Their hierarchical relationship is defined in the BOM Item Definition table. A child BOM item is created only if its parent BOM item is created.



```
{
  "partNumber": "LP94777",
  "quantity":    1,
  "explodedQuantity": 1,
  "variableName": "LP94777",
  "definition":    {
       "SequenceNum": 1000,
       "ItemId": "1000",
       "ItemType": "Standard Item",
       "Optional": "N"
   },
  "children":    [
        {
           "partNumber": "PRO1109",
           "quantity": 1,
           "explodedQuantity": 1
           "variableName": "LAPPRO1109",
           "definition":      {
               "SequenceNum": 1003,
               "ItemId": "1109",
               "ItemType": "Standard Item",
               "Optional": "N"
            }
       }
   ]
}
```

BOM Item Mapping Administration List

Rule:laptopBOM

BOM Item Variable Name	Configurable Attribute	Configurable Attribute Value
LP94777	areYouLookingForALaptopOrDesktop	Laptop
LAPPRO1101	processor	INTEL
LAPPRO1109	processor	AMD

JSON Representation of a BOM Item

For more information about BOM item instances, refer to the BOM Instances section. The BOM Item Definition provides the default quantity in the BOM instance. The definition node also comes from BOM Item Definition.

Items in the BOM instance, created by BOM item mapping, must exist in the BOM Item Definition. The hierarchy of BOM item mapping is entirely based on the BOM Item Definition. As a result, a child BOM item is generated only if its parent BOM item is included.

RECONFIGURATION BOM ITEM MAPPING

During reconfiguration, the Launch Configuration attributes are mapped from the BOM item instance. Refer to the *JSON Representation of a BOM Item* image for the following example. If "LP94777" is the BOM instance root BOM item, when the Configuration session is launched, the "areYouLookingForALaptopOrDesktop" Configuration attribute is set to "Laptop". Similarly, if the child BOM item "LAPPRO1109" exists, the "processor" Configuration attribute is set to "AMD".

If the BOM Mapping rule values conflict with sales side data, Configuration rules attempt to set the same Configuration attribute with different values. If the attribute is not a Multi Select Menu, a runtime error occurs.

It is considered a match when the sales side Configuration attribute value is "equal", or "the same as" the mapped value. However, if the Configuration attribute is a Multi Select Menu, the comparison operator is inclusively "contains". In other words, as long as the Multi Select Menu attribute contains the value required by the rule, it is considered a match.

BEST PRACTICE RECOMMENDATION: Maintain a 1-to-1 mapping between a BOM item and a Configuration attribute value.

The table-based reverse mapping sets the Configuration attributes based on which BOM items and BOM attributes exist in the BOM instance. It does not attempt to modify the Configuration attributes based on which BOM items or BOM attributes are absent in the BOM instance. Refer the "LAPPRO1109" BOM item example above for the following description. If the BOM instance contains "LAPPRO1109", the "processor" Configuration attribute is set to "AMD" accordingly. If the BOM instance does not contain "LAPPRO1109", the table-based reverse mapping does not attempt to set the "processor" attribute, even if it happens to be "AMD". If the "processor" attribute must be set to a specific value when "LAPPRO1109" is not present, advanced BML can be used to set the value, refer to From BOM Launch Configurations for more information.

When a Commerce Model line that contains a captured Sales BOM is reconfigured, mapping from BOM to Launch Configuration invokes the following actions:

1. The Configuration is restored to the values saved on the Model line
2. The Sales BOM items are validated against the current BOM Item Definition. Invalid items are removed.
3. If Table-based BOM Mapping rules exist, they are invoked first, to map the BOM instance values to the Configuration attributes.
4. If "From BOM to Configuration" advanced BML-based BOM mapping rules that are defined, they are invoked next, using the defined order.

Special attention should be paid to selector configuration attributes. If the selector (search input) value needs to be overridden via BOM Mapping, the Configuration attribute must be set to auto-locked. This is consistent with the current Configuration behavior: the selector recommended value is preferred, unless the attribute is auto-locked during reconfiguration. The auto-lock must be defined in the attribute, instead of being dynamically set by a Recommendation rule.

BOM ATTRIBUTE CONFIGURATION RULES

Similar to BOM item mapping, the BOM attribute mapping is also bidirectional. From Configuration to BOM, the target values are set based on the source types and values. From BOM to Launch Configuration, the initial Configuration attribute values are set from source to target.

BOM attribute mapping is used to set a BOM attribute, the quantity of the BOM item, or a Commerce line attribute. Under each BOM item, administrators can define many BOM attributes using attribute mapping.

Conflicting end values occur if multiple BOM attribute mappings attempt to set the same attribute to different values. Similar results can occur From BOM to Launch Configuration, if Configuration rules attempt to set the same Configuration attribute with different values. These events will generate runtime errors.

The BOM Attribute Mapping table identifies the attribute Target and Source. In the following JSON sample, the “attributes” node stores BOM attribute Target Types, and the “fields” node stores Line attribute Target Types.

```
{
   "partNumber": "BP3025",
   "quantity": 1,
   "variableName": "BP3025",
   "definition":          {
      "SequenceNum": 2020,
      "ItemId": "3025",
      "ItemType": "Standard Item",
      "Optional": "N"
   },
   "attributes":          {
      "BSO3901": {"value": "H~V"},
      "BSN3095": {"value": ""}
   },
   "fields" : {
       "lineActionCode", "Add"
   },
   "explodedQuantity": 1,
   "children": []
 }
```

BOM MAPPING RULE EXECUTION

During Configuration, BOM Mapping rules are invoked when changes to Configuration attribute values are saved. BOM Mapping rules run after Recommendation rules, but before Recommended Item rules. If the Configuration contains multiple nodes, BOM Mapping rules are only run in the last node, similar to Recommended Item rules.

BOM Mapping rules are invoked as follows:

- **Simple Table-based BOM Mapping rules** – During set up, a Configuration Model can be associated with different BOMs using multiple BOM Mapping rules. However, on the sales side, only one Table-based BOM Mapping rule should apply. In other words, a given Configuration state on the sales side can only be associated with one Sales BOM.
- **Advanced BML-based BOM Mapping rules** – BML-based rules are actuated in order, if they are defined. This allows administrators to revise the BOM instance using BML scripts. The BML revised result is an updated JSON BOM instance, which must conform the BOM Item Definition.

During Configuration, administrators can reference the current BOM instance by using the system Configuration attribute `_bm_bom_instance`. If the BOM instance is not null, it can be viewed in the Pipeline Viewer. The Pipeline Viewer displays which Configuration attributes and BOM Mapping rules were used to create the BOM instance.

If the BOM instance is not null, pricing of the BOM parts occurs. If the price is not zero, it is displayed in the Configuration Pricing section.

IMPORTANT: The root BOM item should not have an assigned price. If the root Part Number has a defined price, BOM pricing ignores the defined price. The BOM price displayed during Configuration is the unit price, while the pricing displayed in quotes is the total using an exploded quantity.

EFFECTIVE DATES

By default, BOM Mapping is executed at the current time. When the BOM is stored into a quote, the effective date of the BOM Mapping can be set on the line attribute "Line BOM Effective Date" field of the root BOM item line, (i.e. the Model line). This effective date is honored when:

- A Sales BOM that is extracted from a quote is validated
- A BOM instance is reconfigured

In the latter case, the effective date only applies to BOM Mapping Configuration rules. Other Configuration rules do not honor the BOM Mapping effective date.

MAPPING TO SINGLE SELECT PICK LIST ATTRIBUTES

Beginning in Release 18B, CPQ Cloud provides BOM Mapping support for Single Select Pick Lists, which are also known as Dynamic Menus. When a Single Select Pick List option is selected, the applicable BOM Mapping rules are invoked to add parts to a Transaction based on the Single Select Pick List selection. The Single Select Pick List attribute type can be created in Configuration for Text, Integer, and Float data types.

IMPORTANT: To ensure proper operation of the BOM Mapping rules, the variable names referenced for the Single Select Pick List options must match the Configurable Attribute variable names and values in the BOM Item Mapping table.

BOM INSTANCE

A Configuration BOM instance is represented by a JSON object. 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.

SALES AND MANUFACTURING BOMS

BOM items can be classified as sales items, manufacturing items or both. These items are defined in the BOM Item Definition table.

- CPQ uses sales items to create a BOM instance during Configuration, sales items are used for quotes, and the BOM instance used during reconfiguration contains sales items. In other words, only sales items are used inside CPQ.

  IMPORTANT: Sales Items must have corresponding Part Numbers defined in the CPQ Parts Database.

- Manufacturing items are only used when CPQ submits a Manufacturing BOM to a back-end Fulfillment system. Administrators use the BML getbom function to create a Manufacturing BOM from the saved Configuration in a quote.

CAPTURE A BOM INSTANCE IN COMMERCE TRANSACTIONS

The BOM instance is stored in a quote when the end user saves the Configuration to a quote.

Structure

The root BOM item is equivalent to a Model line, and it will be processed as Model line within CPQ. Model lines do not use part fields; therefore, the root BOM item Part Number is stored in the "Line BOM Part Number" field. The other part fields of the root BOM item are not saved into the Model line. As a result, part-based pricing is not executed on the root BOM item. Customers should not define part pricing for the root BOM item.

All child BOM items are stored as child Part lines of the Model line. All child and grandchild BOM items are stored as the direct children of the Model line. Hierarchical relationships are stored the "Line BOM Parent ID" field.

When a BOM tree is saved into a quote, the item Variable Names are not stored. During reconfiguration, the getBOM BML function performs the following actions:

- Extracts the Sales BOM from the saved BOM instance
- Validates items against the current BOM Item Definition
- Resolves the BOM item Variable Names

CPQ uses the following system fields to capture a BOM instance into a quote:

Name	Variable Name	Type	Description / Notes
Line Item BOM ID	`_line_bom_id`	Text	The BOM item instance id.
			When ABO is enabled, this field is used to store the asset key.
Line BOM Parent ID	`_line_bom_parent_id`	Text	The parent BOM item instance id.
Part Number	`_part_number`	Text	The part number of the BOM item
			Used for child BOM items, not applicable to root BOM items.
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 Item BOM Attributes	`_line_bom_attributes`	Text Area	BOM attributes, stored as a JSON string.
			Refer to BOM Attributes for the format of the JSON object.
Quantity	`_price_quantity`	Integer	Quantity of the line item.
			The exploded line quantity.
Line BOM Item Quantity	`_line_bom_item_quantity`	Integer	The BOM item line quantity. This is the unexploded line quantity, whereas _price_quantity stores the exploded quantity.
			BOM Item line quantity. This is the unexploded line quantity. The _price_quantity stores the exploded, effective quantity of the line.
Line BOM Level	`_line_bom_level`	Integer	The BOM item depth (level) in the quote.
			Add a new system attribute _line_bom_level, of integer type. • 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`	Date	BOM Effective Date. If null, the current time is used.
			Stores the system attribute in the Model line Effective Date to enforce date effectivity in all BOM mappings. Enforces date effectivity data in the BOM Item Definition and BOM Item Mapping Data Tables. Enforced at runtime for Model to BOM, Get BOM from quote, and BOM to Configuration.
Is Line Item Mandatory?	`_is_line_item_mandatory`	Boolean	Used to identify if the line item is a mandatory part.
			True if the BOM item is not optional. The default logic can be overridden by BOM attribute mapping.

Quantity

BOM Mapping makes it possible to set the Model line “Price Quantity" to numbers greater than one, by using “Price Quantity” and “Line BOM Item Quantity” fields. The child “Line BOM Quantity” value is multiplied by the parent “Price Quantity” value, and the sum is stored in the child “Price Quantity” field.

- **Price Quantity** - the effective or exploded quantity, i.e. multiplied by the parent Price Quantity.
  This field is used to calculate pricing.
- **Line BOM Item Quantity** - the BOM item quantity before explosion.

For example: If the BOM Model line “Line BOM Quantity” is 2, and the child line “Line BOM Quantity” is 3; the BOM Model line “Price Quantity” will be 2, and the child line “Price Quantity” will be 6 *(3*2).*

Model Quantity and Pricing

Since the Model line quantity can be more than one, the total price of the quote uses this quantity when calculating the Model line subtotal. This is different from the standard Configuration total price, where the unit price assumes the Model quantity is one.

SAVE A BOM INSTANCE TO A QUOTE

When the Configuration is saved to a quote, the BOM instance is saved as item lines in the quote. Before the BOM instance is saved to a quote, any defined BOM attributes and BOM attribute translations are added to the BOM instance.

If a BOM item instance contains a "fields" node, it is used to populate the Commerce line attribute. BOM attribute mapping creates a "fields" node when an attribute Target is Line Attribute. Administrators can also use advanced BML-based rules to populate Commerce lines.

When quote line items are initially created, defined line attribute default values take precedence over BOM Mapping attempts to set the line attribute. This is consistent with the current Commerce behavior.

The Configuration date attribute does not have a “time” component. If a Configuration date attribute is mapped to a Commerce date attribute that includes time, the time portion of the Configuration date attribute is set to 00:00 AM on the date in the default time zone, set up on the CPQ server.

IMPLEMENTING BOM MAPPING

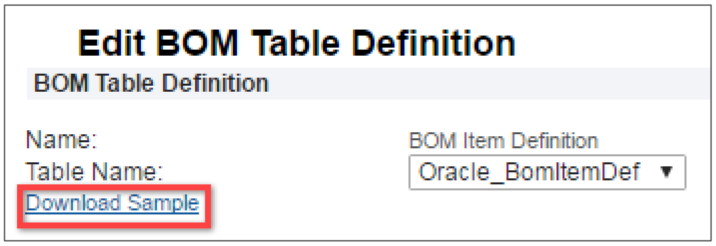
The following items outline setup activities:

- Capture BOM Definitions in CPQ Data Tables
- Activate BOM Mapping Tables
- Map BOM Data Tables to CPQ BOM Platform Tables
- Validate the BOM Item Definition Tree
- Define BOM Mapping Configuration Rules
- Validate Table-based Configuration Rules
- Referencing BOM Items
- BML Functions to Extract Sales and Manufacturing BOMs
- Advanced Settings (Disable Rule Updates and Customize Page Size)

CAPTURE BOM DEFINITIONS IN CPQ BOM DATA TABLES

Administrators perform the following steps to set up BOM Mapping Data Tables.

1. Identify BOM Mapping tables needed to support customer requirements, for more information refer to the BOM Mapping Use Cases section.
2. Populate required Data Tables. CPQ Cloud provides sample table definitions for each BOM table, which define the required schema. Perform the following steps to acquire sample:
   a. Click **Admin** to navigate to the Admin Home page.
   b. Click **BOM** in the **Products** section.
   c. Click BOM Tables in the BOM Declaration section.
   d. Click the **Name** link for the appropriate BOM Table.
   e. Click the **Download Sample** link.



Download BOM Table Schema

IMPORTANT: Use the CPQ database date format for Data Table "Effective From" and "Effective To" fields:

- **Format:** yyyy-mm-dd hh:mm:ss
- **Example:** 2015-09-02 23:58:15

Alternatively, the following ISO date formats can be used:

- YYYY-MM-DD
- YYYY-MM-DDThh:mm:ss
- YYYY-MM-DDThh:mm:ssTZD

⚠ If you open a Data Table CSV file in Excel, Excel may change the date to the "m/d/yyyy hh:mm" format. For example: "9/1/2015 23:58". Verify the date is formatted, as specified above or parsing errors will occur.

For more information on the next two steps, refer to the CPQ Cloud Online Help "Data Table" and "Bulk Upload" topics.

1. Upload the required Data Tables.
2. Deploy the Data Tables.

IMPORTANT: Setting up BOM tables is not part of migration. Administrators should initialize BOM tables in the target environment, before migrating BOM data from the source to the target environment.

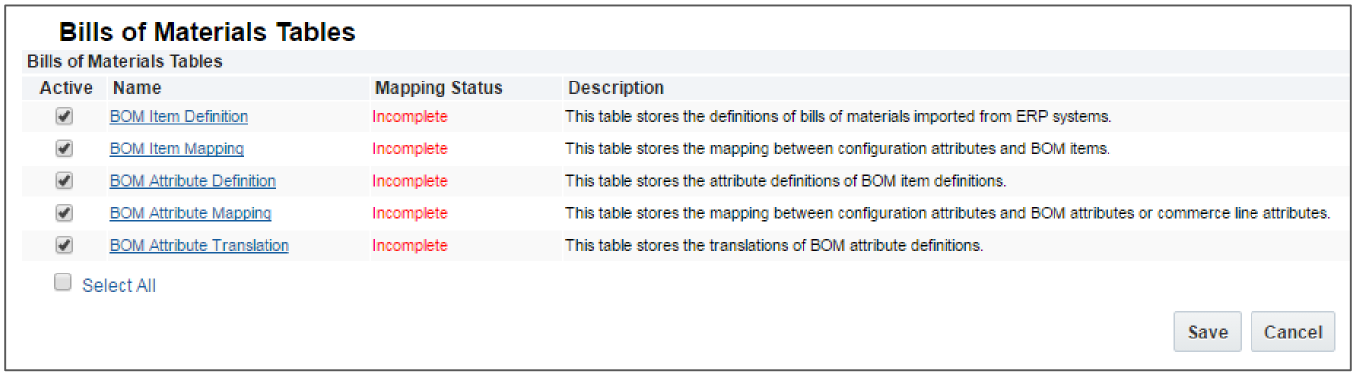
Administrators must set up the required BOM tables. This process is a one-time event. Administrators can modify the setup, if business needs are changed. Setting up BOM tables consists of activation and mapping.

ACTIVATE BOM TABLES

Activation enables BOM tables used for BOM Mapping.

1. Navigate to **Admin > BOM > BOM Tables**.

   The **Bills of Materials Tables** page appears.



Bills of Materials Tables

Bills of Materials Tables

Active	Name	Mapping Status	Description
✓	BOM Item Definition	Incomplete	This table stores the definitions of bills of materials imported from ERP systems.
✓	BOM Item Mapping	Incomplete	This table stores the mapping between configuration attributes and BOM items.
✓	BOM Attribute Definition	Incomplete	This table stores the attribute definitions of BOM item definitions.
✓	BOM Attribute Mapping	Incomplete	This table stores the mapping between configuration attributes and BOM attributes or commerce line attributes.
✓	BOM Attribute Translation	Incomplete	This table stores the translations of BOM attribute definitions.

Bills of Materials Tables – Incomplete Mapping

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

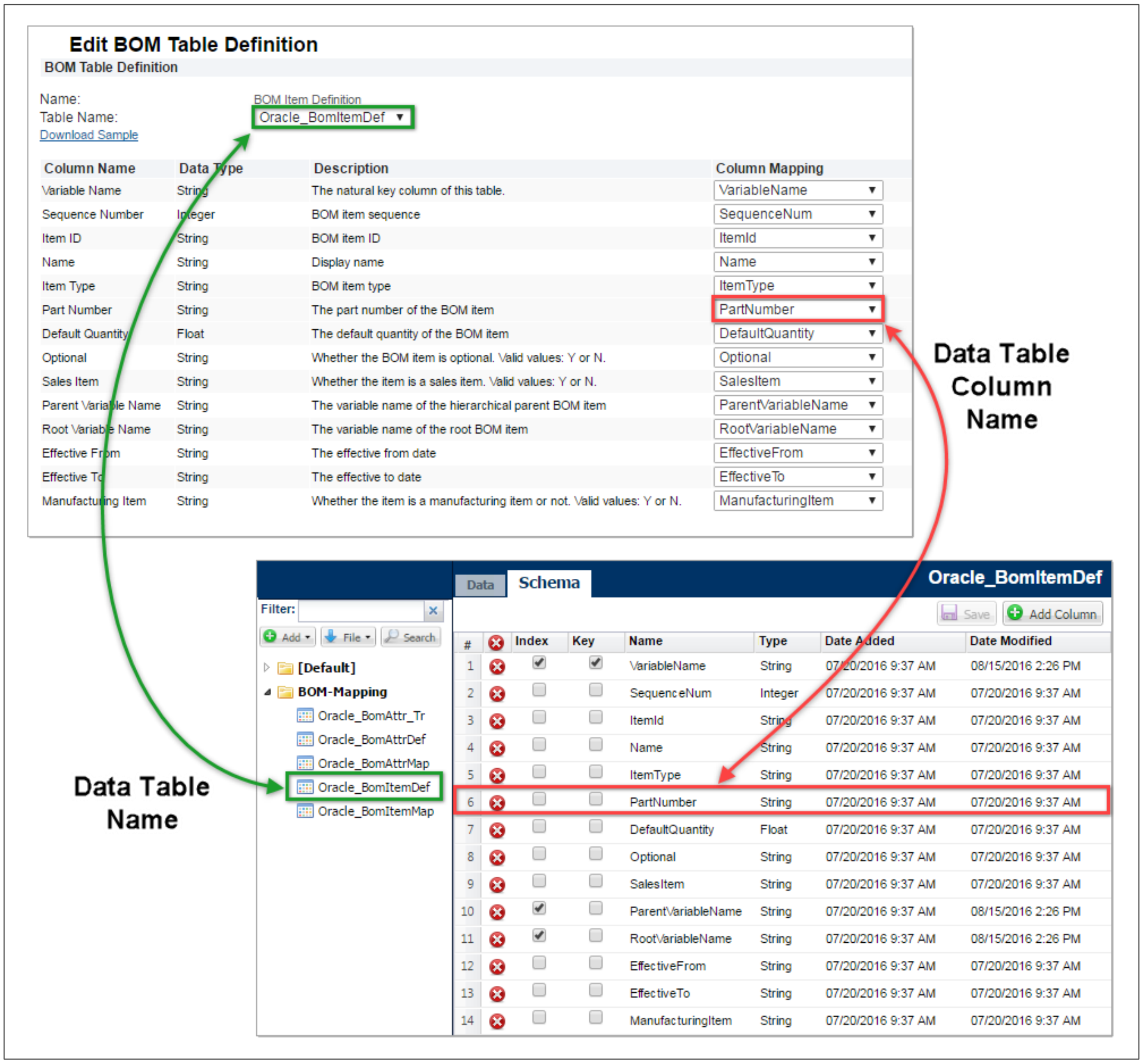
2. Select the appropriate **Active** checkbox to activate the required tables.
3. Click **Save**.

MAP BOM DATA TABLES TO CPQ BOM PLATFORM TABLES

The mapping status of the BOM table displays "Incomplete" for the first activation. After mapping, the status changes to "Complete".

1. Navigate to **Admin > BOM > BOM Tables**.
2. Select the Name link of the appropriate BOM table.
3. Select the appropriate table in the **Table Name** drop-down menu.

   If the selected Data Table column names and data types match the default, the column mapping is automatic. If the column names and data types do not match, map the columns manually. Select the appropriate columns from the **Column Mapping** drop-down menus.
4. Click **Save**, when column mapping is complete.



Edit BOM Table Definition

BOM Table Definition

Name:	BOM Item Definition
Table Name:	Oracle_BomItemDef

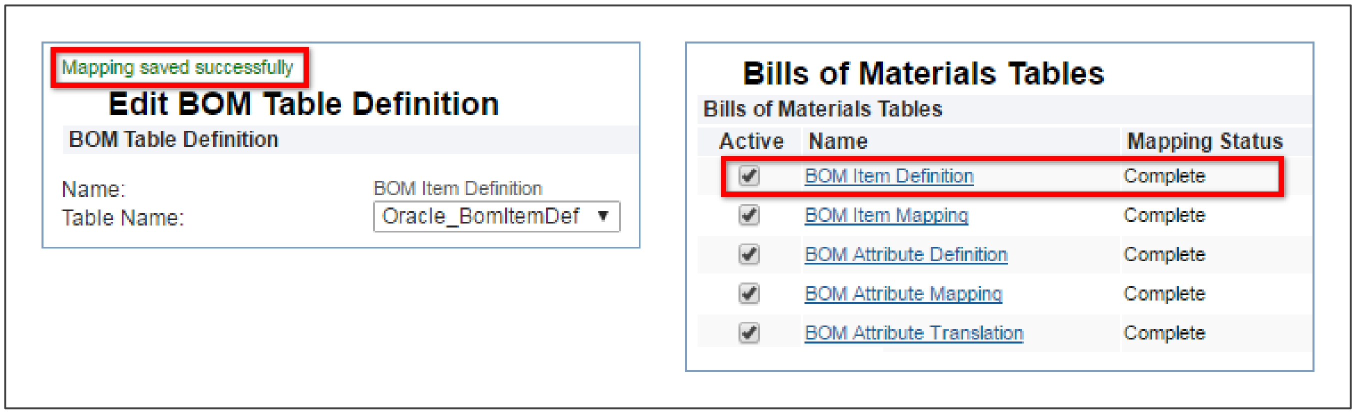
Download Sample

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

#		Index	Key	Name	Type	Date Added	Date Modified
1	✖	✓	✓	VariableName	String	07/20/2016 9:37 AM	08/15/2016 2:26 PM
2	✖			SequenceNum	Integer	07/20/2016 9:37 AM	07/20/2016 9:37 AM
3	✖			ItemId	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
4	✖			Name	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
5	✖			ItemType	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
6	✖			PartNumber	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
7	✖			DefaultQuantity	Float	07/20/2016 9:37 AM	07/20/2016 9:37 AM
8	✖			Optional	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
9	✖			SalesItem	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
10	✖	✓		ParentVariableName	String	07/20/2016 9:37 AM	08/15/2016 2:26 PM
11	✖	✓		RootVariableName	String	07/20/2016 9:37 AM	08/15/2016 2:26 PM
12	✖			EffectiveFrom	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
13	✖			EffectiveTo	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM
14	✖			ManufacturingItem	String	07/20/2016 9:37 AM	07/20/2016 9:37 AM

BOM Item Definition Data Table Mapping

After successful completion of a BOM table mapping, the **Edit BOM Table Definition** page displays "Mapping saved successfully", and the Mapping Status updates to "Complete" on the **Bill of Materials** page.



Mapping saved successfully

Edit BOM Table Definition

BOM Table Definition

Name:	BOM Item Definition
Table Name:	Oracle_BomItemDef

Bills of Materials Tables

Bills of Materials Tables

Active	Name	Mapping Status
✓	BOM Item Definition	Complete
✓	BOM Item Mapping	Complete
✓	BOM Attribute Definition	Complete
✓	BOM Attribute Mapping	Complete
✓	BOM Attribute Translation	Complete

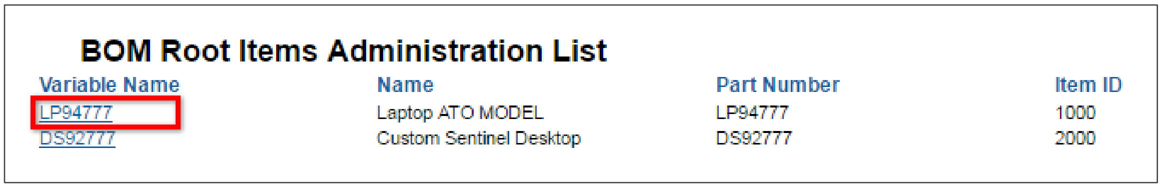
Successful BOM Table Set Up

VALIDATE BOM DEFINITIONS

After mapping BOM Data Tables to CPQ platform BOM Tables, administrators should validate the BOM Item Tree, which contains BOM items, BOM item attributes, and BOM attribute translations.

Perform the following steps to validate the BOM Item Tree.

1. Navigate to **Admin > BOM > BOM Root Items List**.



BOM Root Items Administration List

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

BOM Root Items Administration List Page

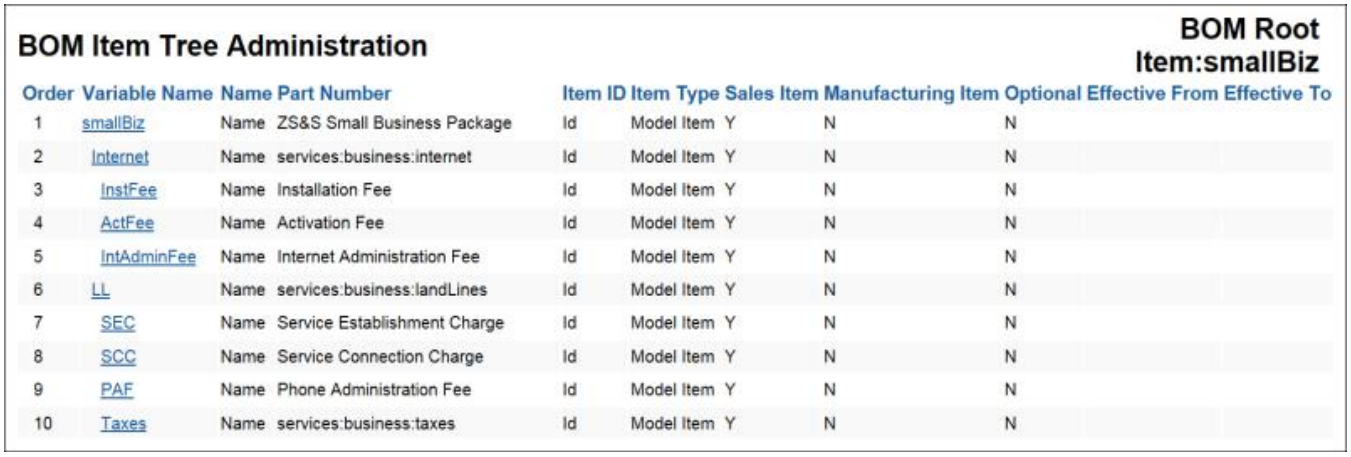
2. Select the appropriate **Variable Name** link.

   The **BOM Item Tree Administration** page appears.

BOM with Models as Children

Beginning in 2017 R1, a BOM can have models as children of other models. Companies can use this feature to offer packaged bundles containing models from separate product families. As in prior releases, the BOM Item Tree Administration page displays the expanded hierarchy and BOM definition information for a root BOM item, child items, and grandchild items. The BOM Item Tree Administration page also displays root models and other models as child items. Each model can have its own individual parts and models.

The following image shows the BOM Item Tree Administration page with models as children of other models. For example, models "Installation Fee", "Activation Fee", and "Internet Administration Fee" are child items to "Internet".



BOM Item Tree Administration

BOM Root Item:smallBiz

Order	Variable Name	Name	Part Number	Item ID	Item Type	Sales Item	Manufacturing Item	Optional	Effective From	Effective To
1	smallBiz	Name	ZS&S Small Business Package	Id	Model Item	Y	N	N		
2	Internet	Name	services:business:internet	Id	Model Item	Y	N	N		
3	InstFee	Name	Installation Fee	Id	Model Item	Y	N	N		
4	ActFee	Name	Activation Fee	Id	Model Item	Y	N	N		
5	IntAdminFee	Name	Internet Administration Fee	Id	Model Item	Y	N	N		
6	LL	Name	services:business:landLines	Id	Model Item	Y	N	N		
7	SEC	Name	Service Establishment Charge	Id	Model Item	Y	N	N		
8	SCC	Name	Service Connection Charge	Id	Model Item	Y	N	N		
9	PAF	Name	Phone Administration Fee	Id	Model Item	Y	N	N		
10	Taxes	Name	services:business:taxes	Id	Model Item	Y	N	N		

IMPORTANT: Child models are not reconfigurable. They behave as mandatory models added by Recommended Item Rules.

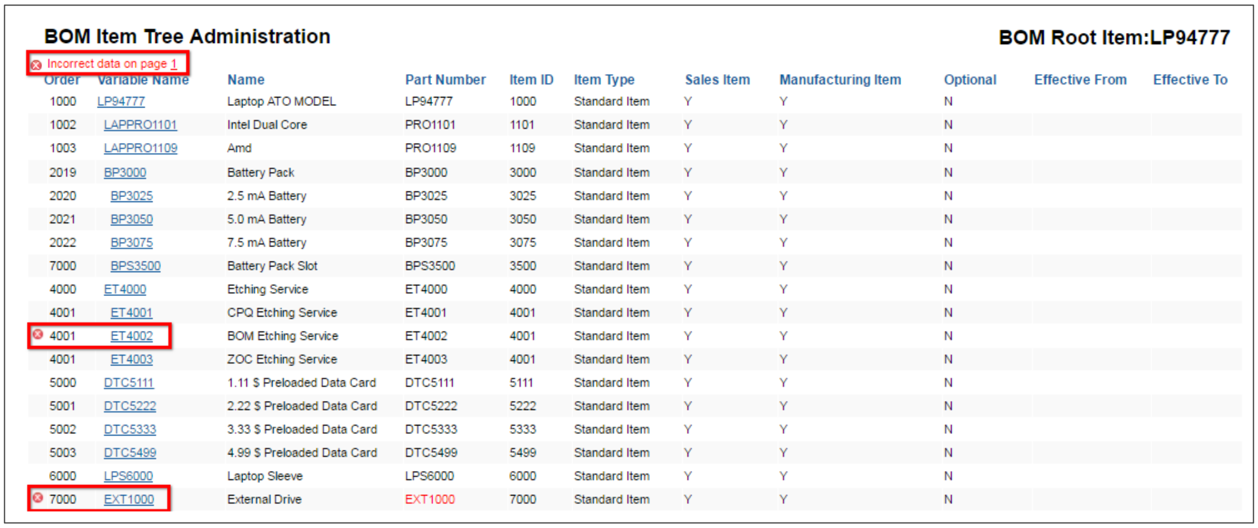
Validation Errors

If there are any validation errors in the BOM item definition, the BOM Item Tree Administration page displays an error message, at the top of the page. The item with an error is proceeded by error indicator ⊗. A BOM item definition is marked as incorrect, if errors exist in the item definitions, attribute definitions, or attribute translations.

BOM MAPPING WITH TWO VALIDATION ERRORS

The following example illustrates two validation errors.

As shown in the following image**Error! Reference source not found.**, a message at the top of the page indicates an error on page 1. The sample page also indicates that errors exist for "ET4002" and "ET1000".



BOM Item Tree Administration

BOM Root Item:LP94777

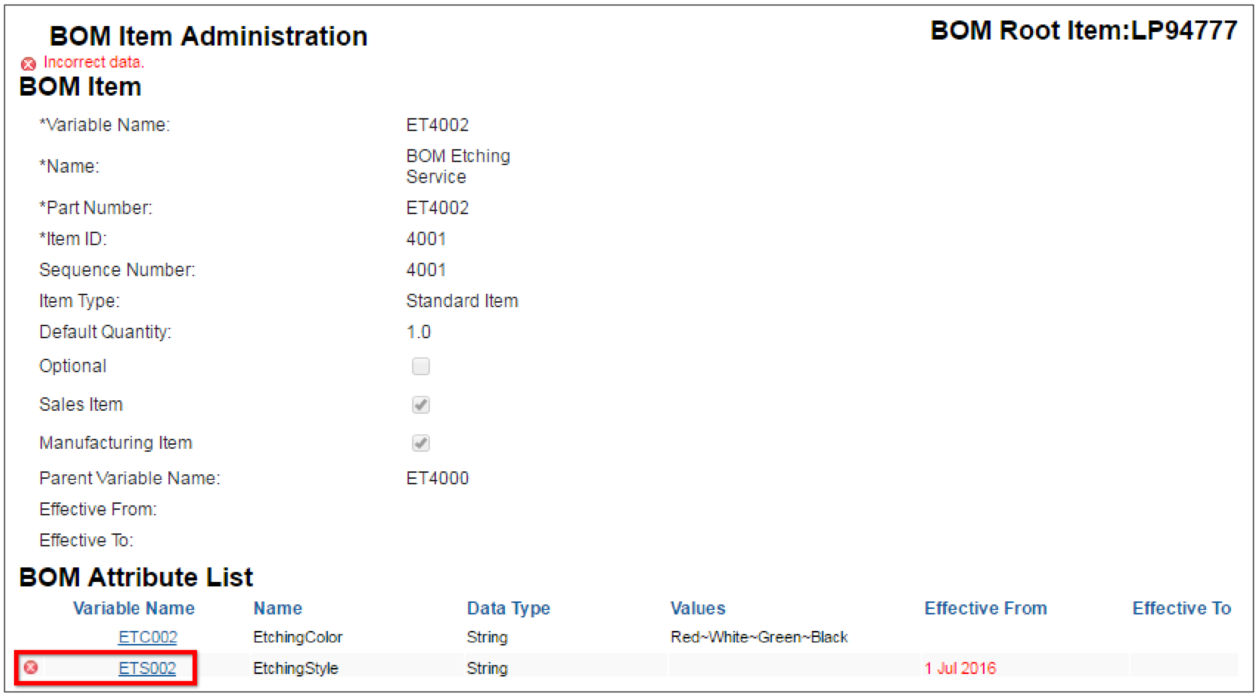
⊗ Incorrect data on page 1

Order	Variable Name	Name	Part Number	Item ID	Item Type	Sales Item	Manufacturing Item	Optional	Effective From	Effective To
1000	LP94777	Laptop ATO MODEL	LP94777	1000	Standard Item	Y	Y	N		
1002	LAPPRO1101	Intel Dual Core	PRO1101	1101	Standard Item	Y	Y	N		
1003	LAPPRO1109	Amd	PRO1109	1109	Standard Item	Y	Y	N		
2019	BP3000	Battery Pack	BP3000	3000	Standard Item	Y	Y	N		
2020	BP3025	2.5 mA Battery	BP3025	3025	Standard Item	Y	Y	N		
2021	BP3050	5.0 mA Battery	BP3050	3050	Standard Item	Y	Y	N		
2022	BP3075	7.5 mA Battery	BP3075	3075	Standard Item	Y	Y	N		
7000	BPS3500	Battery Pack Slot	BPS3500	3500	Standard Item	Y	Y	N		
4000	ET4000	Etching Service	ET4000	4000	Standard Item	Y	Y	N		
4001	ET4001	CPQ Etching Service	ET4001	4001	Standard Item	Y	Y	N		
⊗ 4001	ET4002	BOM Etching Service	ET4002	4001	Standard Item	Y	Y	N		
4001	ET4003	ZOC Etching Service	ET4003	4001	Standard Item	Y	Y	N		
5000	DTC5111	1.11 $ Preloaded Data Card	DTC5111	5111	Standard Item	Y	Y	N		
5001	DTC5222	2.22 $ Preloaded Data Card	DTC5222	5222	Standard Item	Y	Y	N		
5002	DTC5333	3.33 $ Preloaded Data Card	DTC5333	5333	Standard Item	Y	Y	N		
5003	DTC5499	4.99 $ Preloaded Data Card	DTC5499	5499	Standard Item	Y	Y	N		
6000	LPS6000	Laptop Sleeve	LPS6000	6000	Standard Item	Y	Y	N		
⊗ 7000	EXT1000	External Drive	EXT1000	7000	Standard Item	Y	Y	N		

BOM Item Tree Administration Page with Validation Errors

Click on the Variable Name link, for details about the error.

- Refer to the *Part Number Does Not Exist in CPQ Parts Database* image to view "EXT1000" error details.
- As shown in the following image, the error for "ET4002" is caused by "ETS002", one of the associated attributes.



BOM Item Administration

BOM Root Item:LP94777

Incorrect data.

BOM Item

*Variable Name:	ET4002
*Name:	BOM Etching Service
*Part Number:	ET4002
*Item ID:	4001
Sequence Number:	4001
Item Type:	Standard Item
Default Quantity:	1.0
Optional	☐
Sales Item	✓
Manufacturing Item	✓
Parent Variable Name:	ET4000
Effective From:	
Effective To:	

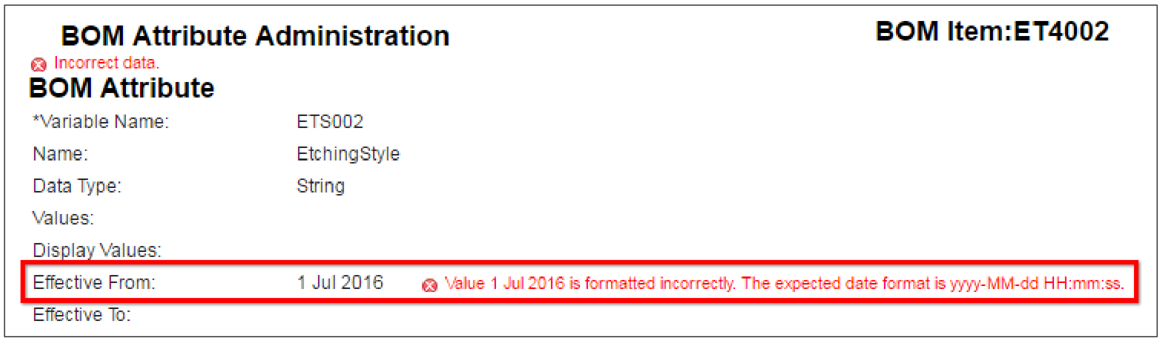
BOM Attribute List

Variable Name	Name	Data Type	Values	Effective From	Effective To
ETC002	EtchingColor	String	Red~White~Green~Black		
ETS002	EtchingStyle	String		1 Jul 2016	

ET4002 Error Example

Click on the attribute Variable Name link, for details about the error.

The following image displays an error message denoting that the “Effective From” value is formatted incorrectly.



BOM Attribute Administration

BOM Item:ET4002

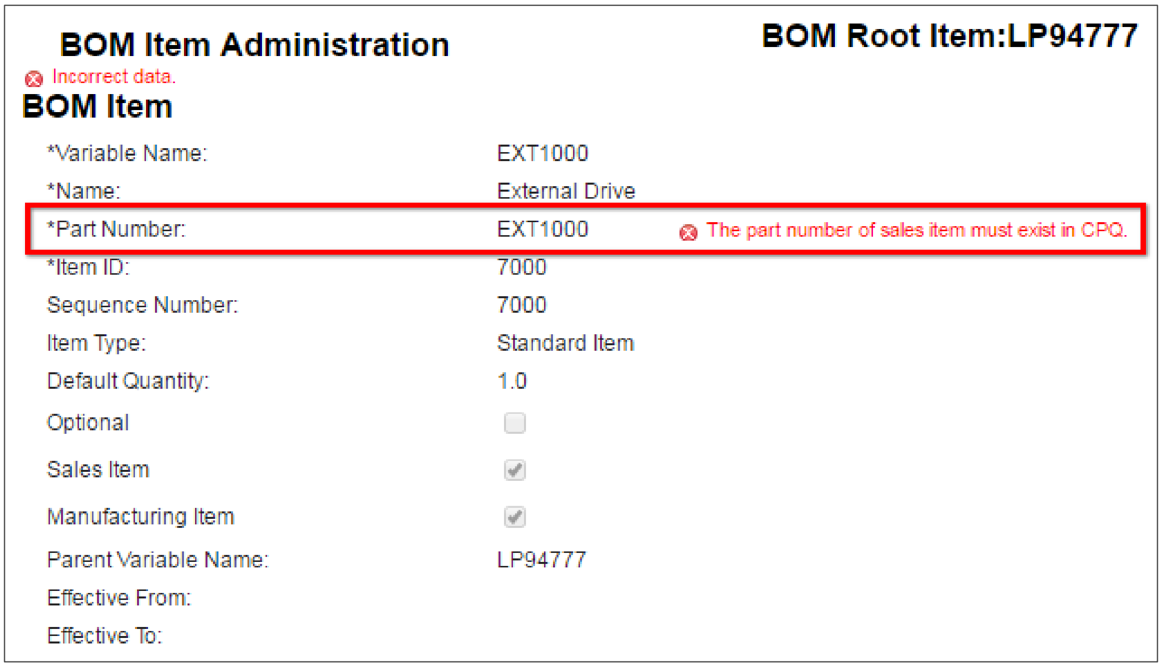
Incorrect data.

BOM Attribute

*Variable Name:	ETS002	
Name:	EtchingStyle	
Data Type:	String	
Values:		
Display Values:		
Effective From:	1 Jul 2016	Value 1 Jul 2016 is formatted incorrectly. The expected date format is yyyy-MM-dd HH:mm:ss.
Effective To:		

Date Formatted Incorrectly

As shown in the following image, “EXT1000” is displaying an error because the Part Number does not exist in the CPQ Parts Database.



BOM Item Administration

BOM Root Item:LP94777

⊗ Incorrect data.

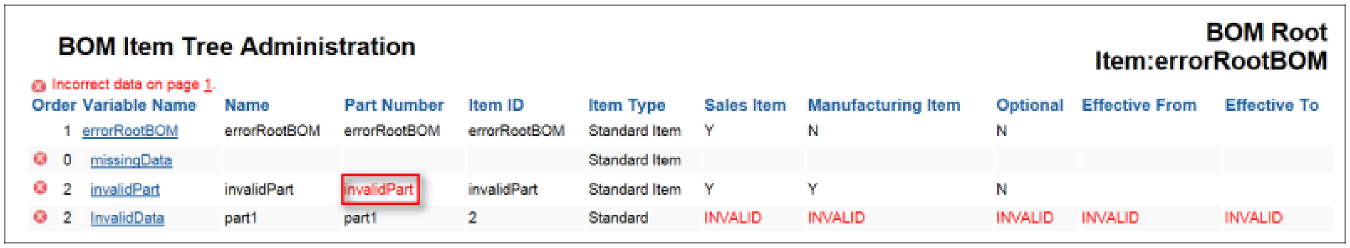
BOM Item

*Variable Name:	EXT1000	
*Name:	External Drive	
*Part Number:	EXT1000	⊗ The part number of sales item must exist in CPQ.
*Item ID:	7000	
Sequence Number:	7000	
Item Type:	Standard Item	
Default Quantity:	1.0	
Optional	☐	
Sales Item	✓	
Manufacturing Item	✓	
Parent Variable Name:	LP94777	
Effective From:		
Effective To:		

Non-existent Part Number Error

VALIDATION ERROR IN BOM HIERARCHY WITH MODELS AS CHILDREN

As shown below, when administrators define child items in the BOM hierarchy as models, CPQ Cloud validates the models in the BOM item definition in the same way as other BOM errors. The BOM Item Tree Administration page displays an error message at the top of the page. The items with an error are proceeded by the following error indicator.



BOM Item Tree Administration

BOM Root Item:errorRootBOM

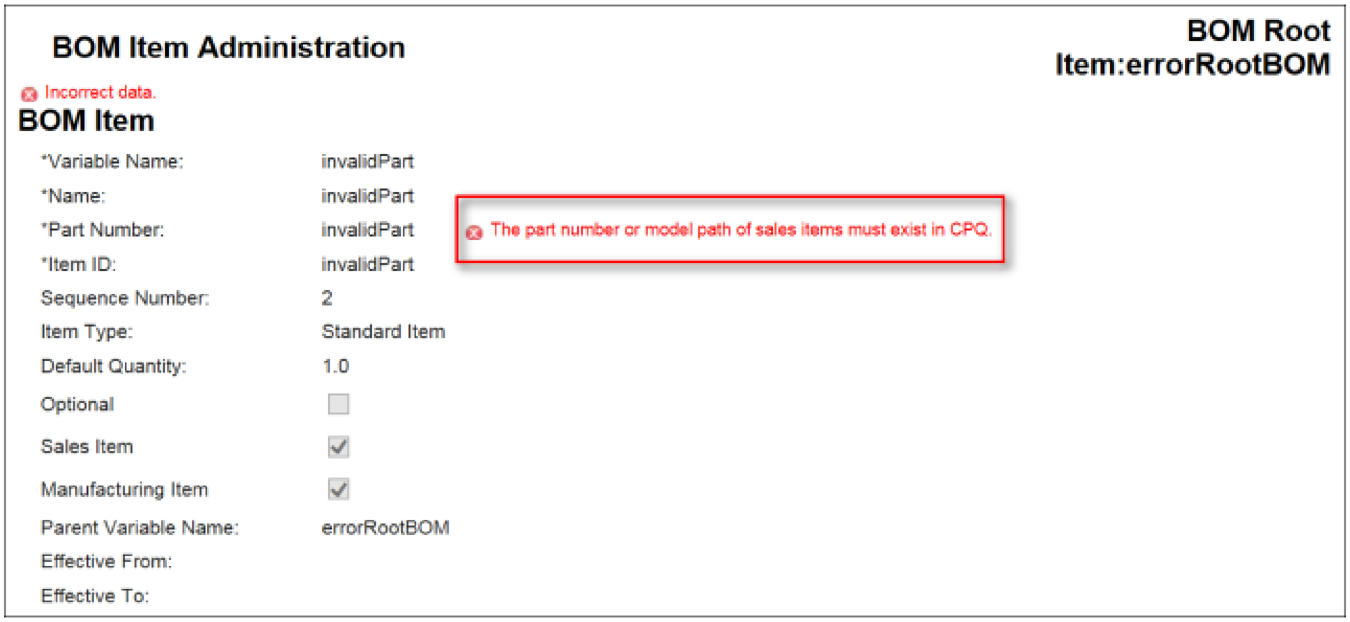
⊗ Incorrect data on page 1.

	Order	Variable Name	Name	Part Number	Item ID	Item Type	Sales Item	Manufacturing Item	Optional	Effective From	Effective To
	1	errorRootBOM	errorRootBOM	errorRootBOM	errorRootBOM	Standard Item	Y	N	N		
⊗	0	missingData				Standard Item					
⊗	2	invalidPart	invalidPart	invalidPart	invalidPart	Standard Item	Y	Y	N		
⊗	2	InvalidData	part1	part1	2	Standard	INVALID	INVALID	INVALID	INVALID	INVALID

BOM Item Tree with Errors

Click on the attribute **Variable Name** link, for details about the error.

When administrators click the **Variable Name** associated with the error, the **BOM Item Administration** page opens. As shown in the following image, an error message displays when the part number associated with a BOM item or the path to a model in the BOM hierarchy does not exist in CPQ Cloud.



BOM Item Administration

BOM Root Item:errorRootBOM

Incorrect data.

BOM Item

*Variable Name:	invalidPart
*Name:	invalidPart
*Part Number:	invalidPart
*Item ID:	invalidPart
Sequence Number:	2
Item Type:	Standard Item
Default Quantity:	1.0
Optional	☐
Sales Item	✓
Manufacturing Item	✓
Parent Variable Name:	errorRootBOM
Effective From:	
Effective To:	

The part number or model path of sales items must exist in CPQ.

Non-existent Part Number of Model Path

IMPORTANT:

- Errors must be resolved before setting up BOM Mapping Configuration rules.
- If the BOM Item Administration page shows that parts or models do not exist when an administrator has created them, deploy the BOM Item Definition table.
- If the "Buy Direct" field of a part number referenced by a BOM item definition is modified, redeploy the BOM Item Definition data table.

CREATE BOM MAPPING CONFIGURATION RULES

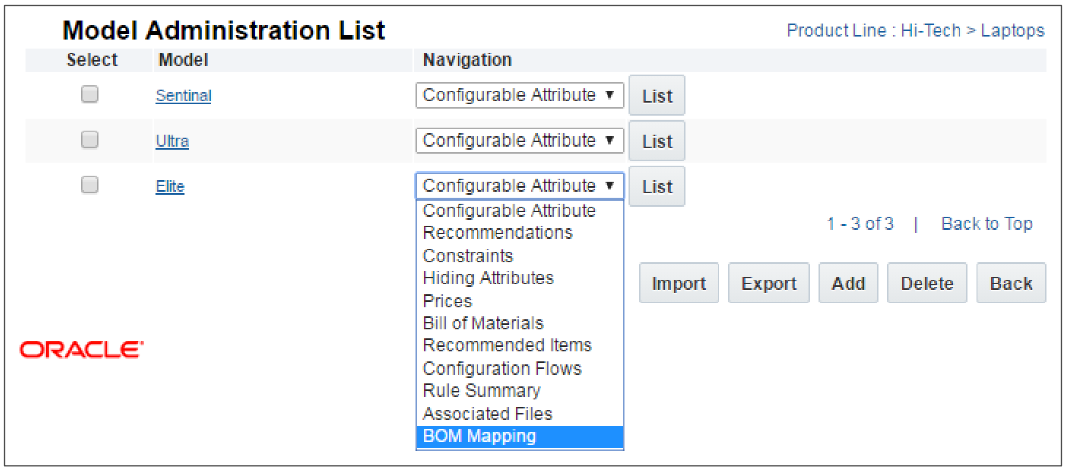
Administrators create BOM Mapping Configuration rules at the Model level. BOM Mapping rules associate Configuration attributes to the BOM items. Use simple Table-Based Rules, advanced BML-Based Rules, or both for BOM Mapping.

Perform the following steps to create a BOM Mapping Configuration rule.

1. Navigate to the **Model Administration List** page.

   Admin > **Catalog Definition** > **Product Families** > **Product Lines** > **Models**

2. Select **BOM Mapping** from the **Navigation** drop-down menu, and click **List**.



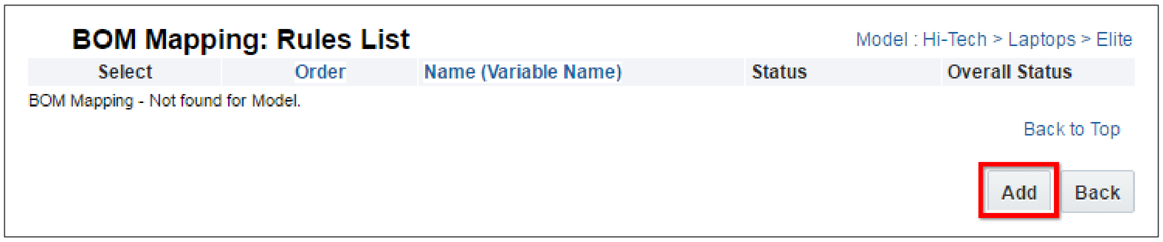
Model Administration List

Product Line : Hi-Tech > Laptops

Select	Model	Navigation
☐	Sentinal	Configurable Attribute ▼ List
☐	Ultra	Configurable Attribute ▼ List
☐	Elite	Configurable Attribute ▼ List

Access BOM Mapping Rules for a Model

3. Click **Add** to set up a new BOM Mapping Configuration rule.



BOM Mapping: Rules List

Model : Hi-Tech > Laptops > Elite

Select	Order	Name (Variable Name)	Status	Overall Status

BOM Mapping - Not found for Model.

BOM Mapping: Rules List Page

4. Continue to one of the following procedures:

   - Define Simple Table-based Configuration Rules
   - Define Advanced BML-based Configuration Rules

DEFINE SIMPLE TABLE-BASED CONFIGURATION RULES

A simple table-based BOM mapping rule uses the BOM item mapping data table to declaratively map the configuration attributes to BOM items.

IMPORTANT: The BOM Item Definition and BOM Item Mapping tables must be activated to use simple Table-based Configuration rules. If the BOM Item Mapping table is not activated, advanced BML-based must be used.

1. Enter a **Name**, **Variable Name**, and **Description** for the new BOM Mapping Configuration rule.
2. Define the **Condition Type** for BOM mapping rules.

   Beginning in Release 18B, administrators can define a logic condition that specifies when a rule should run. There are three options: **Always True**, **Simple Condition**, and **Advanced Condition**.

   a. **Always True**: The rule will fire automatically, every time, because a specific condition does not need to be met.

   b. **Simple Condition**: The rule will fire based on defined condition attributes, operators, and values that are selected from drop-down menus.

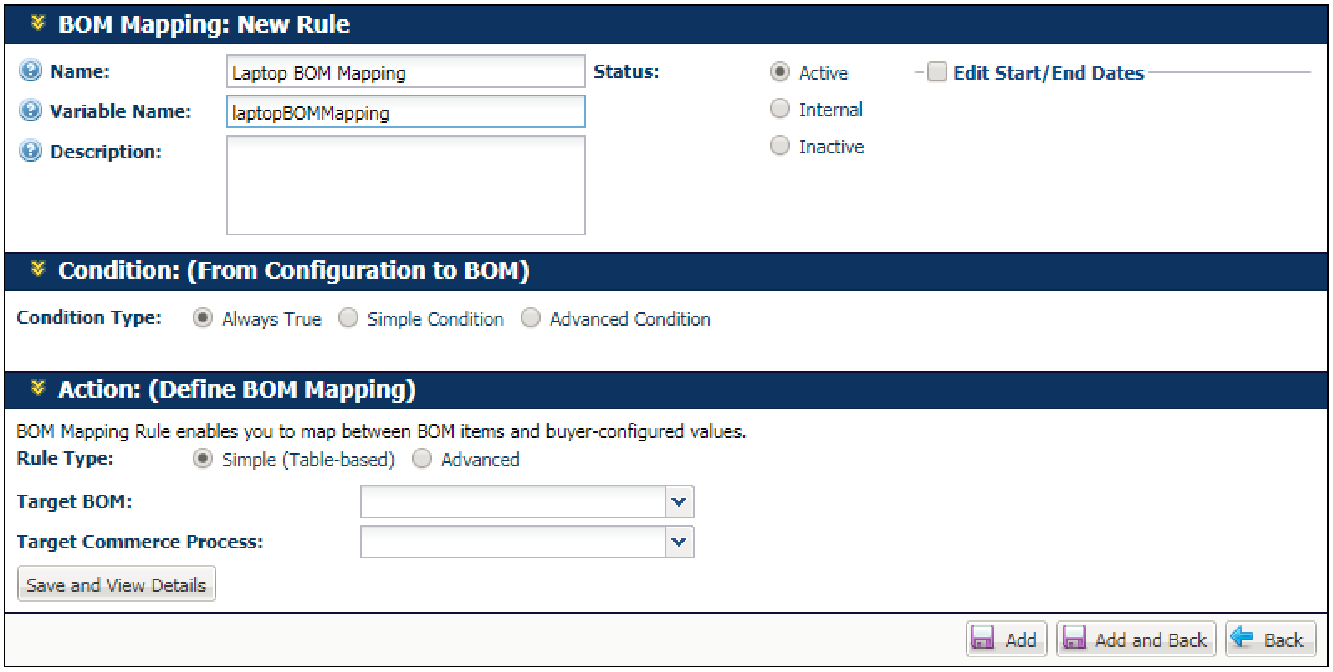
      When **Simple Condition** is enabled, administrators can define Attributes, Operators, Values, Range Operators, and Range Values for the desired condition. They can also Add Rows and set the Order of Operations to provide multiple conditions.

   c. **Advanced Condition**: The rule will fire based on an advanced function written with BML.

      The advanced function is meant for complex condition logic. Administrators use BML to define conditions in the Function Editor. After selecting the **Advanced Condition** type, administrators select the **View/Edit the BML Function** to access the Function Editor.

IMPORTANT: Simple and Advanced Conditions are only evaluated during Configuration; these conditions are NOT evaluated during Reconfiguration.

3. Select a **Target BOM** from the drop-down menu. The target BOM is an available BOM in the BOM root items list.
4. Select a **Target Commerce Process**, if BOM attributes are mapped to line attributes.
5. Click Save and View Details.



BOM Mapping: New Rule

Name:	Laptop BOM Mapping	Status:	[● Active] ○ Internal ○ Inactive
Variable Name:	laptopBOMMapping		☐ Edit Start/End Dates
Description:			

Condition: (From Configuration to BOM)

Condition Type: [● Always True] ○ Simple Condition ○ Advanced Condition

Action: (Define BOM Mapping)

BOM Mapping Rule enables you to map between BOM items and buyer-configured values.

Rule Type: [● Simple (Table-based)] ○ Advanced

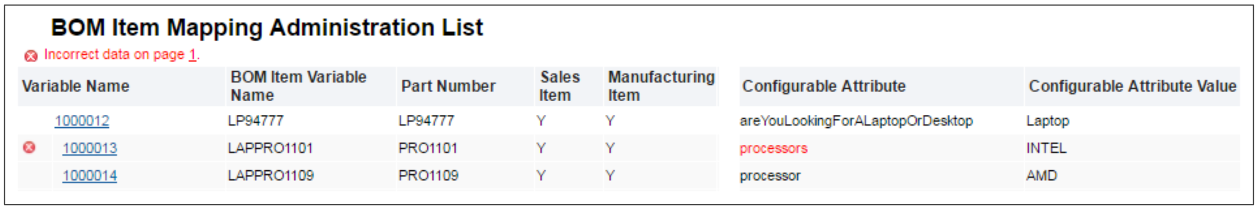
Target BOM:

Target Commerce Process:

Create BOM Mapping Rule

Validate Table-based BOM Mapping Rules

If there are validation errors for the BOM item mapping or its children BOM attribute mappings, an error icon is shown up on the BOM item.



BOM Item Mapping Administration List

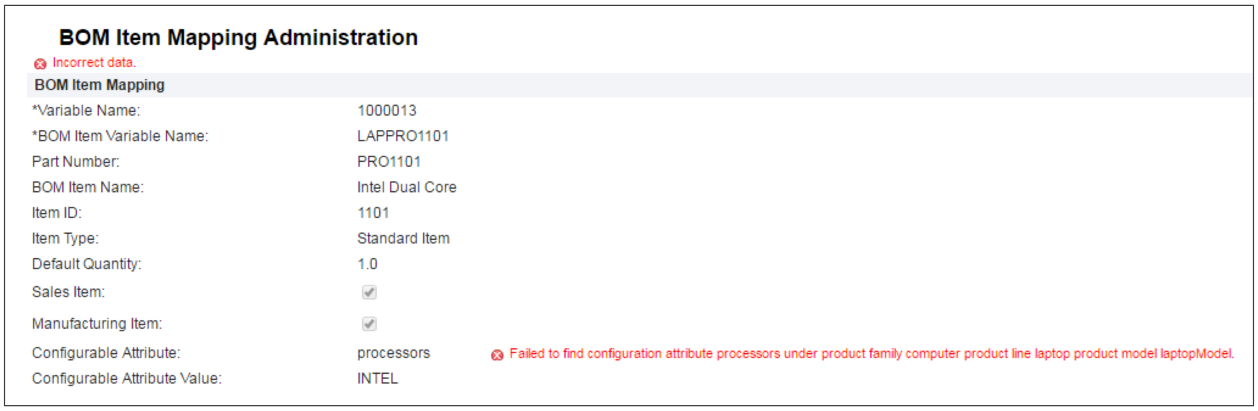
Incorrect data on page 1.

	Variable Name	BOM Item Variable Name	Part Number	Sales Item	Manufacturing Item	Configurable Attribute	Configurable Attribute Value
	1000012	LP94777	LP94777	Y	Y	areYouLookingForALaptopOrDesktop	Laptop
⊗	1000013	LAPPRO1101	PRO1101	Y	Y	processors	INTEL
	1000014	LAPPRO1109	PRO1109	Y	Y	processor	AMD

BOM Item Mapping Validation Error

Click on the item Variable Name link, for details about the error.

As shown in the following image, there is a typo of the configurable attribute name. It should be "processor" instead of "processors".



BOM Item Mapping Administration

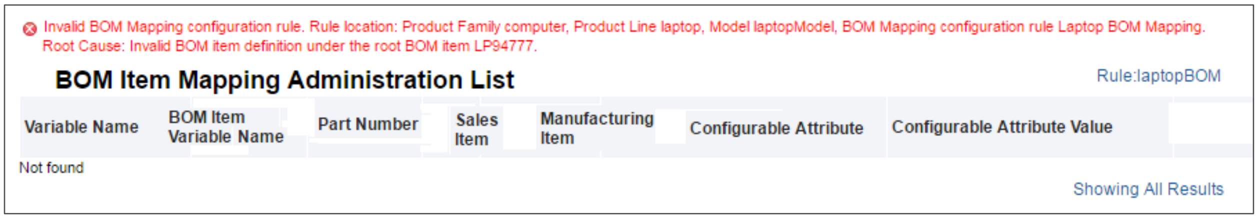
Incorrect data.

BOM Item Mapping

*Variable Name:	1000013	
*BOM Item Variable Name:	LAPPRO1101	
Part Number:	PRO1101	
BOM Item Name:	Intel Dual Core	
Item ID:	1101	
Item Type:	Standard Item	
Default Quantity:	1.0	
Sales Item:	✓	
Manufacturing Item:	✓	
Configurable Attribute:	processors	Failed to find configuration attribute processors under product family computer product line laptop product model laptopModel.
Configurable Attribute Value:	INTEL	

BOM Item Mapping with Configurable Attribute Error

In the following example, the following error is displayed after the administrator selects **Save and View Details**. The BOM Item Definition table had more than one root item under LP94777. Each Table-based BOM Mapping rule can only map to a single root BOM item tree. If the data table contains more than one root BOM item under a single BOM mapping rule, it is a validation error.



Invalid BOM Mapping configuration rule. Rule location: Product Family computer, Product Line laptop, Model laptopModel, BOM Mapping configuration rule Laptop BOM Mapping.
Root Cause: Invalid BOM item definition under the root BOM item LP94777.

BOM Item Mapping Administration List

Rule:laptopBOM

Variable Name	BOM Item Variable Name	Part Number	Sales Item	Manufacturing Item	Configurable Attribute	Configurable Attribute Value

Not found

Showing All Results

BOM Mapping Rule with Invalid BOM Item Definition Error

When creating BOM Item Configuration rules, administrators can define multiple BOM mapping rules, each being associated with different root BOM items. However, on the sales side, only one of the table-based BOM mapping rules should apply, based on the Configuration attribute and value defined in the root BOM item mapping. In other words, a given Configuration state on the sales side can only be associated with a single Sales BOM.

Mapping to Configurable Array Attributes

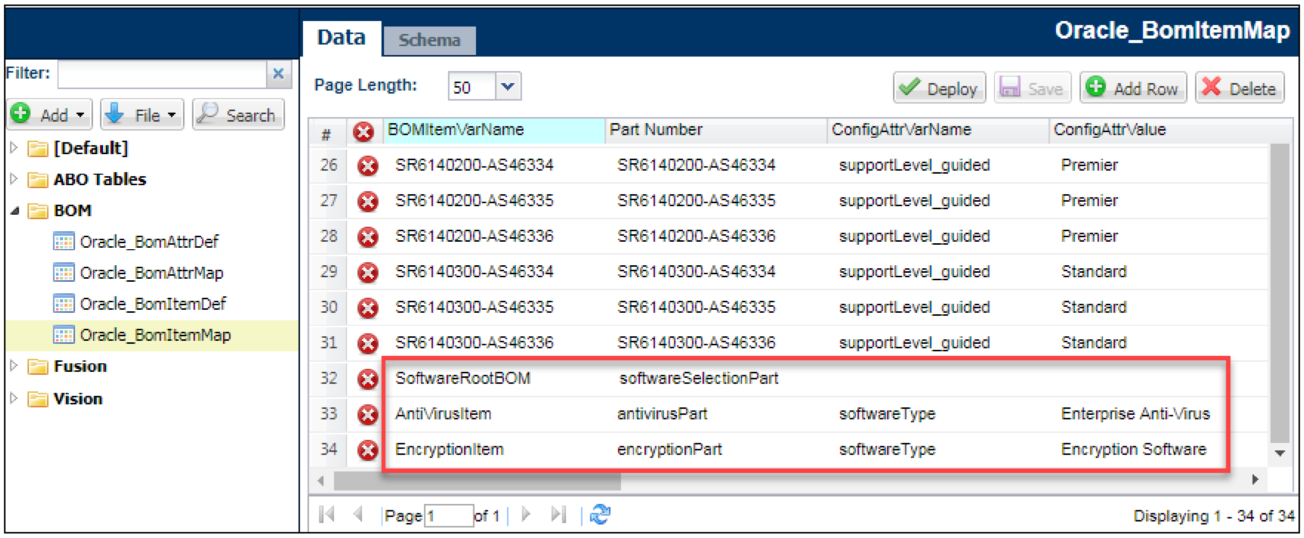
Table-based BOM Mapping supports mapping to configurable array attributes in both BOM item mapping and BOM attribute mapping.

BOM ITEM MAPPING FOR CONFIGURABLE ARRAY ATTRIBUTES

When Configurable array attributes are used with BOM Item Mapping, the Array Set row items are mapped to a matching BOM item. The mapping should be one-to-one between the Array Set row and a BOM item.

For example: In the following BOM Item Mapping Data Table, there are two BOM items displayed under the "SoftwareRootBOM" item:

- AntiVirusItem - Part Number "antiVirusPart", ConfigAttrVarName "softwareType", and ConfigAttrValue "Enterprise Anti-Virus"
- EncryptionItem - Part Number "encryptionPart", ConfigAttrVarName "softwareType", and ConfigAttrValue "EncryptionSoftware"

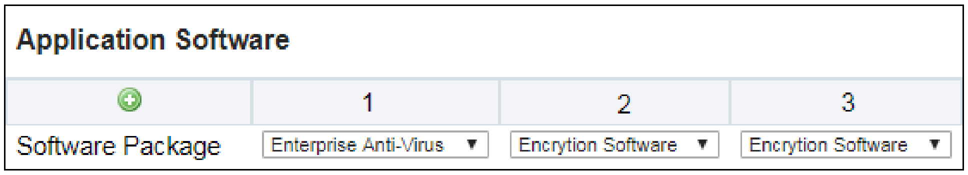


#	BOMItemVarName	Part Number	ConfigAttrVarName	ConfigAttrValue
26	SR6140200-AS46334	SR6140200-AS46334	supportLevel_guided	Premier
27	SR6140200-AS46335	SR6140200-AS46335	supportLevel_guided	Premier
28	SR6140200-AS46336	SR6140200-AS46336	supportLevel_guided	Premier
29	SR6140300-AS46334	SR6140300-AS46334	supportLevel_guided	Standard
30	SR6140300-AS46335	SR6140300-AS46335	supportLevel_guided	Standard
31	SR6140300-AS46336	SR6140300-AS46336	supportLevel_guided	Standard
32	SoftwareRootBOM	softwareSelectionPart		
33	AntiVirusItem	antivirusPart	softwareType	Enterprise Anti-Virus
34	EncryptionItem	encryptionPart	softwareType	Encryption Software

BOM Item Mapping Data Table

Customers can use Configurable Array Sets to define BOM items to be selected during Configuration.

For example: The following image displays an Application Software Configuration item with three items: "Enterprise Anti-Virus", "Encryption Software", and "Encryption Software".



Application Software Configuration with Software Packages

When these BOM items are selected during Configuration, the following BOM Instance is generated:

```
{
  "variableName": "SoftwareRootBOM",
  "partNumber": "softwareSelectionPart",
  "quantity": 1,
  "category": "sales",
  "children": [{
      "variableName": "AntiVirusItem",
      "partNumber": "antiVirusPart",
      "quantity": 1
    }, {
      "variableName": "EncryptionItem",
      "partNumber": " encryptionPart",
      "quantity": 1
    }, {
      "variableName": "EncryptionItem",
      "partNumber": "encryptionPart",
      "quantity": 1
    }]
 }
```

IMPORTANT:

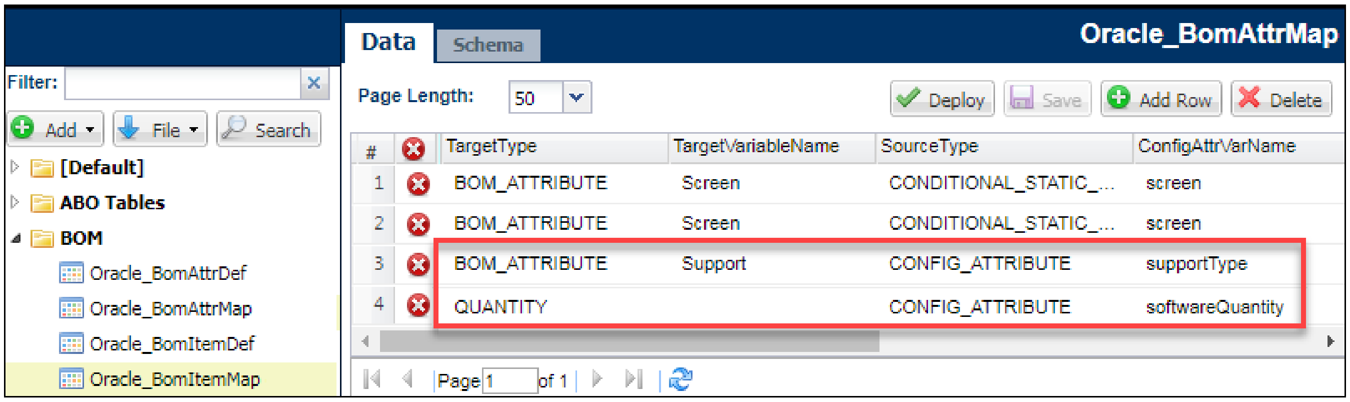
- When configurable array attributes from an array set are used in more than one BOM item mappings, the same array attributes form the set must be used in all BOM item mappings.
- Refer to Appendix G: BOM Mapping Configurable Array Attribute Restrictions for more restrictions on BOM item mappings and explanations.

BOM ATTRIBUTE MAPPING FOR CONFIGURABLE ARRAY ATTRIBUTES

When Configurable Array Attributes are used with BOM Item Mapping, related BOM Attribute Mappings can use additional Configurable Array Attributes from the same array set. The additional Configurable Array Attributes for a BOM item are mapped to Attribute Mapping items.

For example: In the following BOM Attribute Mapping Data Table, two Configurable Attributes are displayed:

- supportType - TargetType "BOM_ATTRIBUTE", TargetVariableName "Support", and SourceType "CONFIG_ATTRIBUTE".
- softwareQuantity - TargetType "QUANTITY" and SourceType "CONFIG_ATTRIBUTE".



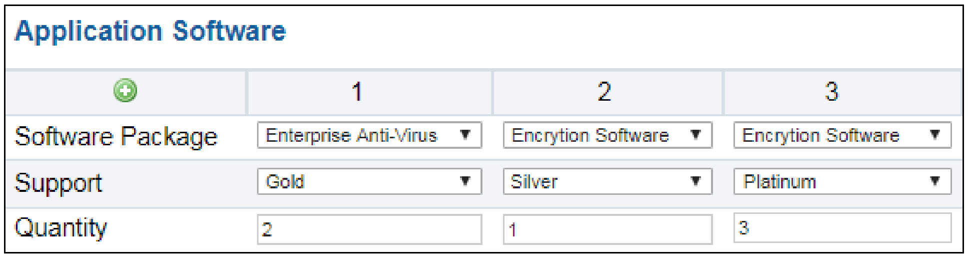
#	TargetType	TargetVariableName	SourceType	ConfigAttrVarName
1	BOM_ATTRIBUTE	Screen	CONDITIONAL_STATIC_...	screen
2	BOM_ATTRIBUTE	Screen	CONDITIONAL_STATIC_...	screen
3	BOM_ATTRIBUTE	Support	CONFIG_ATTRIBUTE	supportType
4	QUANTITY		CONFIG_ATTRIBUTE	softwareQuantity

BOM Attribute Mapping Data Table

Customers can also use Configurable Array Sets to define BOM item attributes to be selected during Configuration.

For example: The following image displays an Application Software Configuration with three items:

- "Enterprise Anti-Virus" with "Gold" support and quantity "2"
- "Encryption Software" with "Silver" support and quantity "1"
- "Encryption Software" with "Platinum" support and quantity "3"



	1	2	3
Software Package	Enterprise Anti-Virus	Encrytion Software	Encrytion Software
Support	Gold	Silver	Platinum
Quantity	2	1	3

Application Software Configuration with Software Packages, Support, and Quantity

When these BOM items and attributes are selected during Configuration, the following BOM Instance is generated:

```
{
  "variableName": "SoftwareRootBOM",
  "partNumber": "softwareSelectionPart",
  "quantity": 1,
  "category": "sales",
  "children": [{
      " variableName": "AntiVirusItem",
      "partNumber": "antiVirusPart",
      "quantity": 2,
      "attributes": {"Support": {value: "Gold"}}
    }, {
      " variableName": "EncryptionItem",
      "partNumber": " encryptionPart",
      "quantity": 1,
      "attributes": {"Support": {value: "Silver"}}
    }, {
      " variableName": "EncryptionItem",
      "partNumber": " encryptionPart",
      "quantity": 3,
      "attributes": {"Support": {value: "Platinum"}}
    }]
 }
```

IMPORTANT:

- BOM Attributes can be mapped to both Configurable Array Attributes and non-array type Configurable Attributes.
- Refer to Appendix G: BOM Mapping Configurable Array Attribute Restrictions for restrictions on BOM attribute mappings

RECONFIGURATION OF CONFIGURABLE ARRAY ATTRIBUTES

During reconfiguration, the array set is initialized to be consistent with the BOM items in the quote. For example: If a quote contains a single BOM item that maps to the array set, the size of the array set is set to one.

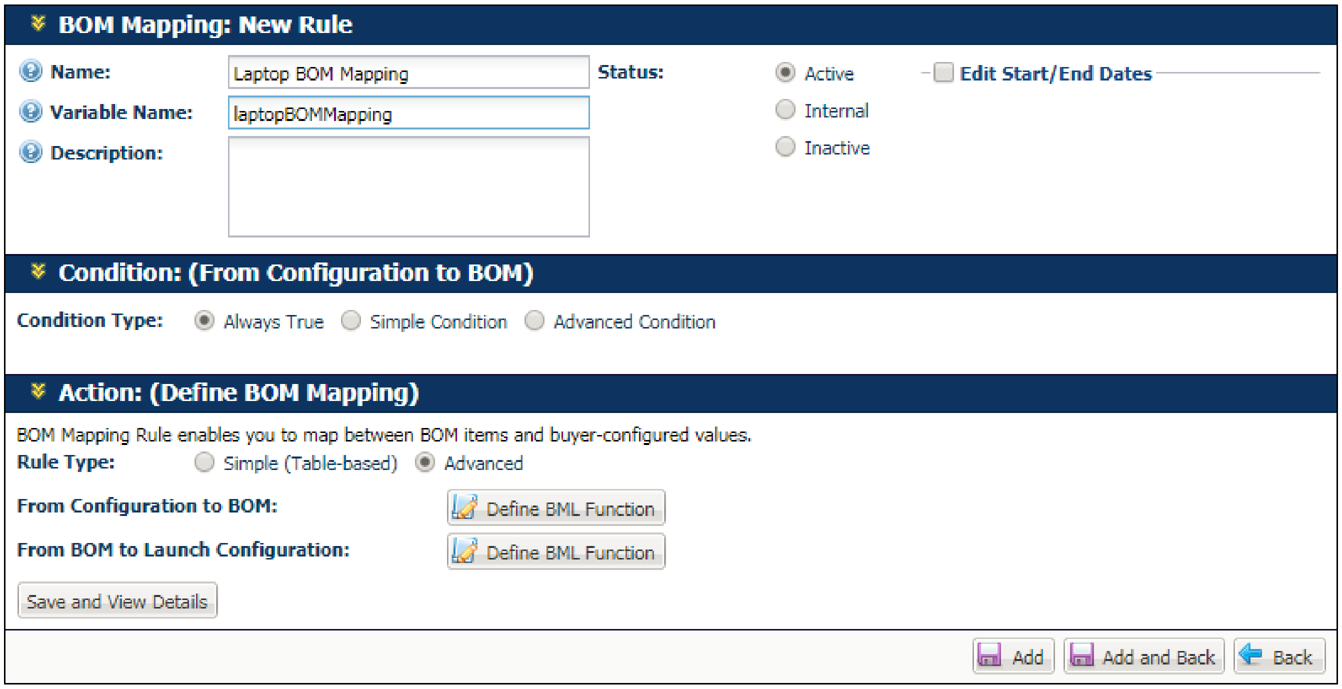
A more complicated use case occurs when some of the array-set attribute domain values are not mapped to BOM items. In the case, during reconfiguration the unmapped records are left unmodified. For example:

- A Configuration array set "Dessert" is defined, with an attribute called "DessertType" with three options: Coffee, Ice cream and Cake. "Ice-cream" is mapped to BOM item "BOMIcecream", "Cake" is mapped to BOM item "BOMCake", but "Coffee" is not mapped.
- On the sales side, there are only two rows in the "Dessert" menu, one is "Coffee", and the other is "Cake".
- When a user selects “Cake” during Configuration, a Sales BOM is created, containing a BOM item "BOMCake".
- When the BOM instance is saved to a quote, there is a part line corresponding to "BOMCake".
- When the quote is reconfigured, the "BOMCake" part line is replaced by "BOMIcecream".
- During reconfiguration, the BOM mapping from BOM to Launch Configuration replaces the "Cake" in the "Dessert" array set with "Ice cream", but leaves "Coffee" is unchanged, as Coffee is never involved in the BOM mapping.

IMPORTANT: Refer to Appendix G: BOM Mapping Configurable Array Attribute Restrictions, to view restrictions that are enforced, primarily to maintain one-to-one mapping between a row in the array set and a BOM item.

DEFINE ADVANCED BML-BASED CONFIGURATION RULES

Advanced BML-based Configuration rules use BML to map between Configuration attributes and BOM items.



BOM Mapping: New Rule

Name: Laptop BOM Mapping

Variable Name: laptopBOMMapping

Description:

Status: [✓ Active] Internal Inactive

Edit Start/End Dates

Condition: (From Configuration to BOM)

Condition Type: [✓ Always True] Simple Condition Advanced Condition

Action: (Define BOM Mapping)

BOM Mapping Rule enables you to map between BOM items and buyer-configured values.

Rule Type: Simple (Table-based) [✓ Advanced]

From Configuration to BOM: Define BML Function

From BOM to Launch Configuration: Define BML Function

BOM Mapping Rule - Advanced Function

Two BML expressions can be defined for the two-way mapping:

- From configuration to BOM items
- From BOM to launch configuration

From Configuration to BOM

BML from configuration to BOM is generally used to revise the BOM instance created by the table-based mapping, though it is possible to use BML-based mapping without the simple, table-based mapping. Use the `_bm_bom_instance` Configuration system attribute to acquire the BOM instance as a JSON object. The BML script should return a JSON object, which is used to set the BOM instance held by the configuration session.

The BOM instance created by the BML-based mapping must conform to the BOM Item Definition. The instance is validated against the current BOM Item Definition to verify:

1. BOM items are valid;
2. The following line item fields exist: `variableName`, `partNumber`, and `quantity`.

For BML-based Configuration rule examples, refer to the following samples.

READ AND UPDATE BOM ITEMS SAMPLE

The following example uses BML JSON functions to double the child BOM item quantities.

```
//---------------------------------------------------------------
// Sample BML from Configuration to BOM
//
// Double the quantities of the direct child BOM items of the root BOM
/
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON
//
// Output:
//    The BOM item instance in JSON
//--------------------------------------------------------------

// Get the first level child BOM items
size = 0;
children = jsonget(_bm_bom_instance, "children", "jsonarray");
if (isnull(children) == false) {
    size = jsonarraysize(children);
}

// Loop through child BOM items.
// No-op if the children is empty or does not exist.
indices = range(size);
for index in indices {
    bomItem = jsonarrayget(children, index, "json");
    qty = jsonget(bomItem, "quantity", "integer");
    jsonput(bomItem, qty * 2);
}

return _bm_bom_instance;
```

IMPORTANT: `jsonget` fetches `children` by reference, and `jsonarrayget` returns `bomItem` by reference. As a result, updating `bomItem` directly updates `_bom_item_instance`.

ADD A BOM ITEM SAMPLE

The following example creates the BOM item, and appends it to the "children" node of the parent BOM item.

```
// Sample code snippet to add a child BOM item to the root
tb1 = json();
jsonput(tb1, "variableName", "TB1");
jsonput(tb1, "partNumber", "40mb_100gb");
jsonput(tb1, "quantity", 3);

// Get the "children" node by reference.
// Subsequent changes to "children" directly update _bm_bom_instance
children = jsonget(_bm_bom_instance, "children", "jsonarray");
if (isnull(children)) {
    jsonput(_bm_bom_instance, "children", jsonarray());
    children = jsonget(_bm_bom_instance, "children", "jsonarray");
}

// Append the child BOM item.
jsonarrayappend(children, tb1);
```

The following sample BOM instance is created:

```
{
  "partNumber": "telecom_package",
  "quantity": 2,
  "isModel": false,
  "explodedQuantity": 2,
  "category": "sales",
  "variableName": "TBRoot",
  "children": [  {
    "partNumber": "40mb_100gb",
    "quantity": 3,
    "isModel": false,
    "explodedQuantity": 6,
    "variableName": "TB1"
  }]
}
```

IMPORTANT:

- `explodedQuantity` is automatically updated once all BML-based BOM mapping rules are executed and does not need to be set in BML.
- `isModel` will default to **False** and does not need to be set in BML. This option is only set to **True** for System Configuration.

CREATE A ROOT BOM FROM SCRATCH SAMPLE

If table-based BOM mapping rules do not exist, a root BOM can be created from scratch. In this special case, the `_bm_bom_instance` system attribute starts as an empty skeleton without the part number or variable name: `{"category": "sales", "isModel": false}`

```
//--------------------------------------------------------------
// Sample BML from Configuration to BOM
// Create the root BOM from scratch when there are no table-based BOM mapping rules
//
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON. It starts as a skeleton as
//        {"category": "sales", "isModel": false},
//        without the part number or variable name.
// Output:
//    The BOM item instance in JSON
//-------------------------------------------------------------

// The following three fields are mandatory fields to create a BOM item
jsonput(_bm_bom_instance, "variableName", "TBRoot");
jsonput(_bm_bom_instance, "partNumber", "telecom_package");
jsonput(_bm_bom_instance, "quantity", 1);
return _bm_bom_instance;
```

This results in the following BOM:

```
{
   "partNumber": "telecom_package",
   "quantity": 2,
   "isModel": false,
   "explodedQuantity": 2,
   "category": "sales",
}
```

`explodedQuantity` is automatically updated once all BML-based BOM mapping rules are executed and does not need to be set in BML.

In BML from Configuration to BOM, the input attribute `_bm_bom_instance` is never null. If the BOM item instance does not exist for the current Configuration session, an empty skeleton without the part number or variable name is sent.

There are subtle differences between table-based mapping and BML-based mapping. If table-based mapping is defined, it is always fired first. It creates a brand new BOM instance each time it is invoked and BML-based mapping can then make additional adjustments to that BOM. In this case, the BML does not have to be idempotent. However, if table-based mapping is not defined, the BML mapping receives the BOM instance created from the previous update as the starting point and the BML then must be idempotent.

For example: a BML-based BOM mapping creates a warranty BOM item based on configurable attribute "Select Warranty". If table-based mapping is defined, the table-based mapping creates a new BOM instance before the BML mapping is fired each time. However, if table-based mapping is not defined, the BML can work on a BOM instance from the previous update and must account for the possibility that the warranty BOM item may have been created in the previous update. Without that logic, a duplicate warranty BOM item may be created each time the end user clicks "Update".

DELETE A BOM ITEM SAMPLE

The following example removes the last child BOM item from the root BOM.

```
//-------------------------------------------------------------
// Sample BML from Configuration to BOM
// Remove the last child BOM item.
//
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON
//
// Output:
//    The BOM item instance in JSON
//------------------------------------------------------------

size = 0;
children = jsonget(_bm_bom_instance, "children", "jsonarray");
if (isnull(children) == false) {
    size = jsonarraysize(children);
}
if (size > 0) {
    jsonarrayremove(children, size - 1);

}

return _bm_bom_instance;
```

For more complicated cases, JSON path can be used. For example: in Oracle EBS, optional class BOM items are defined to serve as containers of optional items. In this example, all empty optional class items that do not have child items are removed.

```
//-------------------------------------------------------------
// Sample BML from Configuration to BOM
// Remove empty optional class items.
// An empty optional class item is as such:
//     - Its "definition" contains Itemtype == 'CLASS'.
//     - Its "children" is empty
//
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON
//

// Output:
//    The BOM item instance in JSON
//------------------------------------------------------------

jsonpathremove(_bm_bom_instance, "$..children[?(@.definition.ItemType=='CLASS' &&
@.children.length()==0)]");

return _bm_bom_instance;
```

IMPORTANT: Release 18B provides support for removal of empty "Option Class" BOM items in table-based mapping. Refer to "Option Class" BOM Item Type for more details.

ADD A BOM ATTRIBUTE SAMPLE

In the following example, BML is used to insert the "membership card type" BOM attribute, based on the customer rating. It also illustrates how to use BML to implement comparison operators other than equal.

```
//----------------------------------------------------------------
// Sample BML from Configuration to BOM
// Set the membership card type based on the customer rating
//   If 70 <= customerRating, set cardType = Platinum
//   If 30 <= customerRating < 70, set cardType = Gold
//   else set cardType = Silver
//
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON
//    customerRating: an integer from 0 to 100.
//
// Output:
//    The BOM item instance in JSON
//---------------------------------------------------------------

// Locate the membership card BOM item

membershipCard =jsonpathgetsingle(_bm_bom_instance, "$..children[?(@.variableName
== 'MembershipCard')]", "json");

if (isnull(membershipCard) == false) {

// Create cardType BOM attribute

    cardTypeValue = "Silver";
    if (70 <= customerRating) {
       cardTypeValue = "Platinum";
    } elif ((30 <= customerRating) AND (customerRating < 70)) {
        cardTypeValue = "Gold";
    }
    cardType = json();
    jsonput(cardType, "label", "Membership Card Type");
    jsonput(cardType, "value", cardTypeValue);

    // Attach the cardType BOM attribute

    attributes = jsonget(membershipCard, "attributes", "json");
    if (isnull(attributes)) {
        jsonput(membershipCard, "attributes", json());
        attributes = jsonget(membershipCard, "attributes", "json");
    }
    jsonput(attributes, "cardType", cardType);
}

return _bm_bom_instance;
```

CONFIGURABLE ARRAY ATTRIBUTES SAMPLE

This example demonstrates how to use configurable array attributes to create BOM items. The BML code loops through the software array set and creates the corresponding BOM item based on the software type selected.

```
//---------------------------------------------------------------
// Sample BML from Configuration to BOM
// Create BOM items based-on configurable array attributes
//
// Input Attributes:
//    _bm_bom_instance:    the BOM item instance in JSON
//    softwareArraySize:   int
//    softwareType:        string[]
//    softwareQuantity:    integer[]
//    softwareSupport:     string[]
//
// Output:
//    The BOM item instance in JSON
//-------------------------------------------------------------

// Set up the root BOM item
jsonput(_bm_bom_instance, "variableName", "SoftwareRootBOM");
jsonput(_bm_bom_instance, "partNumber", "softwareSelectionPart");
jsonput(_bm_bom_instance, "quantity", 1);

// Set up children
children = jsonarray();
indices = range(softwareArraySize);
for index in indices {
   software = softwareType[index];
   variableName = "";
   partNumber = "";
   if (software == "Enterprise Anti-Virus") {
      variableName = "AntiVirusItem";
      partNumber = "antiVirusPart";
   } elif (software == "Enterprise Anti-Spam") {
      variableName = "AntiSpamItem";
      partNumber = "antiSpamPart";
   } else {
     continue;
   }
   // Create the child BOM item
   item = json();
   jsonput(item, "variableName", variableName);
   jsonput(item, "partNumber", partNumber);
   jsonput(item, "quantity", softwareQuantity[index]);

   //Add BOM attributes
   attributes = json();
   attr = json();
   jsonput(attr, "value", supportType[index]);
   jsonput(attributes, "Support", attr);
   jsonput(item, "attributes", attributes);
   jsonarrayappend(children, item);
}
jsonput(_bm_bom_instance, "children", children);
return _bm_bom_instance;
```

MANUFACTURING BOM VS SALES BOM

BOM mapping can occur in two scenarios. During a normal Configuration flow, a Sales BOM is created via BOM mapping. In addition, the `getbom` BML function can be used to create a manufacturing BOM for fulfillment integration. Both scenarios trigger all defined BOM mapping rules. Inside the BML-based BOM mapping, the `category` field of `_bm_bom_instance` is used to indicate whether the BML is triggered for sales or manufacturing BOM mapping.

```
// No operations are performed if the BML is invoked for a manufacturing BOM.
// This check is only required when getbom is used to extract the manufacturing BOM

category = jsonget(_bm_bom_instance, "category"); //"sales" or "manufacturing"
if (category == "manufacturing") {
    return _bm_bom_instance;
}
```

JSON PATH FUNCTIONS

The JSON path functions can be used to provide versatile modification of JSON objects.

IMPORTANT: Refer to the JSON Function topic in the CPQ Cloud Administration Online Help for detailed information for the following JSON path functions:

- jsonpathgetsingle
- jsonpathgetmultiple
- jsonpathcheck
- jsonpathremove

From BOM to Launch Configurations

At the beginning of reconfiguration, BML can be used to set initial configuration attribute values to be consistent with the input BOM instance when the configurator is launched. The `_bm_bom_instance` configuration system attribute is used to acquire the BOM instance as a JSON object. The BML execution returns name-value pairs in JSON to set the configuration attributes:

- The key is the configuration attribute variable name.
- The values are typically strings, but any data that can be properly converted to string is acceptable.
- If the configuration attribute is MSM or an array attribute, the value should be a JSON array of strings.

The following table provides Configurable Attribute examples.

Configurable Attributes	Sample Value	Comment
Integer Attribute	"1"	
	1	Interpreted as "1"
Boolean Attribute	"true"	
	"false"	
	true	Interpreted as "true"
	false	Interpreted as "false"
Integer Array (or MSM) Attribute	["1", "2", "3"]	For array attributes, three rows exist in the array set. For MSM, menu item "1", "2", "3" are checked.
	[1, 2, 3]	Interpreted as ["1", "2", "3"]
Boolean Array Attribute	["true", "false", "true"]	
	[true, false, true]	Interpreted as ["true", "false", "true"]

OVERRIDE SIMPLE CONFIGURABLE ATTRIBUTES SAMPLE

In the following example, we set the customer rating to be consistent with the membership card type, if the customer rating is not in the range that is consistent with the card type.

```
//-----------------------------------------------------------------
// Sample BML from BOM to Launch Configuration
//
// Set the membership card type based on the customer rating if the current customer
// rating is inconsistent with the card type
//  If cardType == Platinum and customerRating < 70, set customerRating = 85
//  If cardType == Gold and customerRating not in [30,70), set customerRating = 50
//  If cardType == Silver and customerRating not in [0, 30), set customerRating = 15
//
// Input Attributes:
//   _bm_bom_instance: the BOM item instance in json
//   customerRating: an integer from 0 to 100.
//
// Output:
// attributeOverrides, json as a name-value pair to set configuration attribute values
//-----------------------------------------------------------------
attributeOverrides = json();
if(isnull(_bm_bom_instance) == false){
   cardTypeValue =
jsonpathgetsingle(_bm_bom_instance, "$..children[?(@.variableName ==
'MembershipCard')].attributes.cardType.value");
   if ((cardTypeValue == "Platinum") AND (customerRating < 70)) {
      jsonput(attributeOverrides, "customerRating", 85);
   } elif (cardTypeValue == "Gold" AND (customerRating < 30 OR customerRating
>= 70)) {
      jsonput(attributeOverrides, "customerRating", 50);
   } elif (cardTypeValue == "Silver" AND customerRating > 30) {
      jsonput(attributeOverrides, "customerRating", 15);
   }
}
return attributeOverrides;
```

Note the null check for _bm_bom_instance. Unlike from configuration to BOM, _bm_bom_instance can be null for BML to handle from BOM to configuration. Furthermore, from BOM to configuration is always invoked in the context of sales BOM.

OVERRIDE CONFIGURABLE ARRAY ATTRIBUTES SAMPLE

The following is a more complicated example that sets the values of configurable array attributes. It is the reverse mapping of the previous Configurable Array Attributes Sample.

```
//----------------------------------------------------------------
// Sample BML from BOM to Launch Configuration
// This example sets software array sets based on the enterprise software BOM items.
//
// Input Attributes:
//    _bm_bom_instance: the BOM item instance in JSON
//
// Output:
// attrOverrides, json as a name-value pair to set configuration attribute values
//----------------------------------------------------------------
attrOverrides = json();
// _bm_bom_instance might be null. It is prudent to check.
if(isnull(_bm_bom_instance)) {
   return attrOverrides;
}
size = 0;
children = jsonget(_bm_bom_instance, "children", "jsonarray");
if (isnull(children) == false) {
    size = jsonarraysize(children);
}
softwareTypes = jsonarray();
softwareQuantities = jsonarray();
softwareSupports = jsonarray();
// Loop through "children". No operations are performed if children do not exist
// or are empty.
indices = range(size);
for index in indices {
   item = jsonarrayget(children, index, "json");
   software = "";
   variableName = jsonget (item, "variableName");
   if (variableName == "AntiVirusItem") {
       software = "Enterprise Anti-Virus";
   } elif (variableName == "AntiSpamItem") {
       software = "Enterprise Anti-Spam";
   } else {
       continue;
   }
   quantity = jsonget(item, "quantity", "integer");
   support  = jsonpathgetsingle(item, "$.attributes.Support.value");
   jsonarrayappend(softwareTypes, software);
   jsonarrayappend(softwareQuantities, quantity);
   jsonarrayappend(softwareSupports, support);
}
jsonput(attrOverrides, "softwareType", softwareTypes);
jsonput(attrOverrides, "softwareQuantity", softwareQuantities);
jsonput(attrOverrides, "softwareSupport", softwareSupports);
// set the correct size for the array set
jsonput(attrOverrides, "softwareSize", jsonarraysize(softwareTypes));
return attrOverrides;
```

REFERENCING BOM ITEMS

BOM ITEM DEFINITION TREE REFERENCES

The BOM Item Definition is hierarchical. The “BomItemDef.ParentVariableName” recursively points to its parent “BomItemDef's VariableName”. The BOM Item Definition hierarchy, along with its BOM Attribute Definition children, and BOM Attribute Translation grandchildren forms a BOM item definition tree.

To ease queries and migration, all BOM definition entities must include the root BOM Item Definition Variable Name:

- BomItemDef.RootVariableName
- BomAttributeDef.RootBomItemVarName
- BomAttributeTranslation.RootBomItemVarName

BOM ITEM MAPPING TREE REFERENCES

The BOM Item Map and its children BOM Attribute Maps form a BOM Item Mapping Tree, based on the BOM Item Tree. A BOM Mapping Configuration rule should only reference a single BOM Item Mapping Tree.

To ease queries and migration, all BOM Mapping entities must include the BOM Mapping rule location “productFamilyVarName:productLineVarName:modelVarName:configRuleVarName”, in the following Data Table columns:

- BomItemMap.ParenBomMapVarName
- BomAttrMap.RootBomMapVarName

In the rare case where two Configuration Models use the same BOM item mappings, as defined in the Data Tables, the wildcard rule location can be used, in the format of `*:configRuleVarName`. Exact BOM Mapping rule locations have a higher precedent over wildcard rule locations. A BOM Mapping Configuration rule attempts to load its BOM Item Mapping Tree that uses an exact rule location first. If there are no BOM Item Mapping Data Table records matching the exact rule location, will the BOM Mapping Configuration rule attempt to load a BOM Item Mapping Tree that uses a wildcard rule location. This means that a BOM Mapping Configuration rule loads its BOM Item Mapping Tree using either the exact rule location or the wildcard rule location, but it never loads both locations.

BML FUNCTIONS TO EXTRACT SALES AND MANUFACTURING BOMS

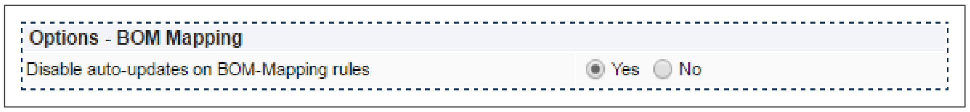
When a Sales BOM is extracted from a quote, the return result is based on the root BOM line (i.e. the root Model line) as its children BOM part lines. If the "validateBomModel" flag is **true**, the last saved BOM instance is validated, and normalized, against the latest BOM item definition.

When a Manufacturing BOM is extracted from a quote, it starts with the saved configuration state on the model line, and executes a BOM mapping from configuration to BOM. The execution fires both the table-based BOM mapping, as well as advance BML mapping. Table-based mapping automatically generates a Manufacturing BOM, based on the BOM item mappings and the BOM item definitions. The BML-based BOM mapping, if defined, is more complicated. The same BML-based BOM mapping rules are invoked to create both a Sales BOM and Manufacturing BOM. Inside the BML script, administrators indicate whether the BML-based rule creates a Sales BOM or a Manufacturing BOM using the `category`" field of the input `_bm_bom_instance`. If the category is "manufacturing", the BML script is invoked to create a Manufacturing BOM.

ADVANCED SETTINGS

Disable Auto Updates

1. Administrators can set Disable auto-updates on BOM Mapping rules.
2. Navigate to Admin > Products > Configuration Settings
3. Set the Disable auto-updates on BOM-Mapping rules option.
    - **No**, is the default setting.
    - **Yes**, disables BOM mapping rules during the Configuration session until the Configuration is saved to a quote.


Options - BOM Mapping

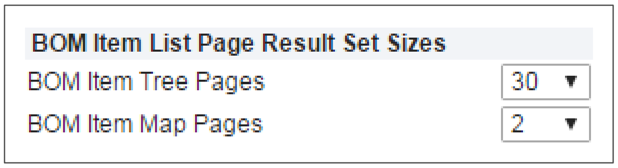
Disable auto-updates on BOM-Mapping rules [✓ Yes] No

Disable BOM Mapping Rule Auto-Updates

Set Row Count per Page

Administrators can customize the BOM Item Tree and BOM Item Mapping Tree page sizes.

1. Navigate to **Admin > General > General Site Options**
2. Click F**eature Settings > Row Count Per Page**.
3. Use the Drop-down menus to set **BOM Item List Page Result Set Sizes**.


BOM Item List Page Result Set Sizes

BOM Item Tree Pages	30
BOM Item Map Pages	2

Customize BOM Item Page Sizes

APPENDIX A: BOM MAPPING SYSTEM ATTRIBUTES

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

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

APPENDIX B: BOM ITEM DEFINITION TABLE SCHEMA

Default table name: Oracle_BomItemDef

Index	Key	Name	Type	Required	Display Name	Description / Comments
Y	Y	`VariableName`	String	Y	Variable Name	The primary key column of this table. This field cannot be empty.
		`SequenceNum`	Integer		Sequence Number	BOM item sequence.
		`ItemId`	String	Y	Item ID	BOM item ID, e.g. Item Id as stored in EBS.
		`Name`	String	Y	Name	Display name.
		`ItemType`	String		Item Type	BOM item type. To enable inherited parent hierarchies for child BOM items, refer to Option Class BOM Item Type.
Y		`PartNumber`	String	Y	Part Number	Contains either the part number of a BOM item or a Model path (i.e. the path to a model in the BOM hierarchy) • This field contains the part number when both a part number and a model path are required for System Configuration inter-model references. • For model path syntax, refer to Model Path Format.
		`DefaultQuantity`	Float		Default Quantity	The default quantity of the BOM item.
		`Optional`	String		Optional	Indicates whether the BOM item is optional. Valid values: Y or N.
		`EffectiveFrom`	String		Effective From	The effective from date. Formatted as `YYYY-MM-DD HH:mm:ss`
		`EffectiveTo`	String		Effective To	The effective to date. Formatted as `YYYY-MM-DD HH:mm:ss`
		`SalesItem`	String		Sales Item	Indicates whether the item is a sales item. Valid values: Y or N.
		`ParentVariableName`	String		Parent Variable Name	The variable name of the hierarchical parent BOM item. This is a recursive key to the parent BOM Item Definition `BomItemDef.VariableName`. • This field is Null for the root Item. • This field is required for all child items otherwise, a validation error will occur.

Index	Key	Name	Type	Required	Display Name	Description / Comments
Y		`RootVariableName`	String	Y	Root Variable Name	The variable name of the root BOM item. This is a recursive key to the root BOM Item Definition `BomItemDef.VariableName`. • For a root BOM Item, the variable name and the root variable name are the same. • Orphan records of dirty root BOM item variable names are ignored.
		`ManufacturingItem`	String		Manufacturing Item	Indicates whether the item is a manufacturing item. Valid values: Y or N.
		`IncludedInBasePrice`	String		Included In Base Price	This is an optional column that indicates whether the item is included in the base price. Valid values: Y or N.

DEFINE PARTS AND MODELS AS CHILD ITEM IN THE BOM HIERARCHY

Any part or model can be defined as a child item in the BOM hierarchy. Administrators accomplish this using either the BOM Item Definition table or the Save BOM BML function.

IMPORTANT: To add a child model to a BOM using a BOM Mapping Rule, administrators must add the child model to the BOM Item Mapping table. If adding a child model to a BOM using the Save BOM BML function, this step is not necessary.

The `PartNumber` column in the BOM Item Definition table can contain either the part number of a BOM item or the path to a model in the BOM hierarchy.

Model Path Format

The format for defining the path to a model is as follows:
`productFamilyVariableName:productLineVariableName:modelVariableName`

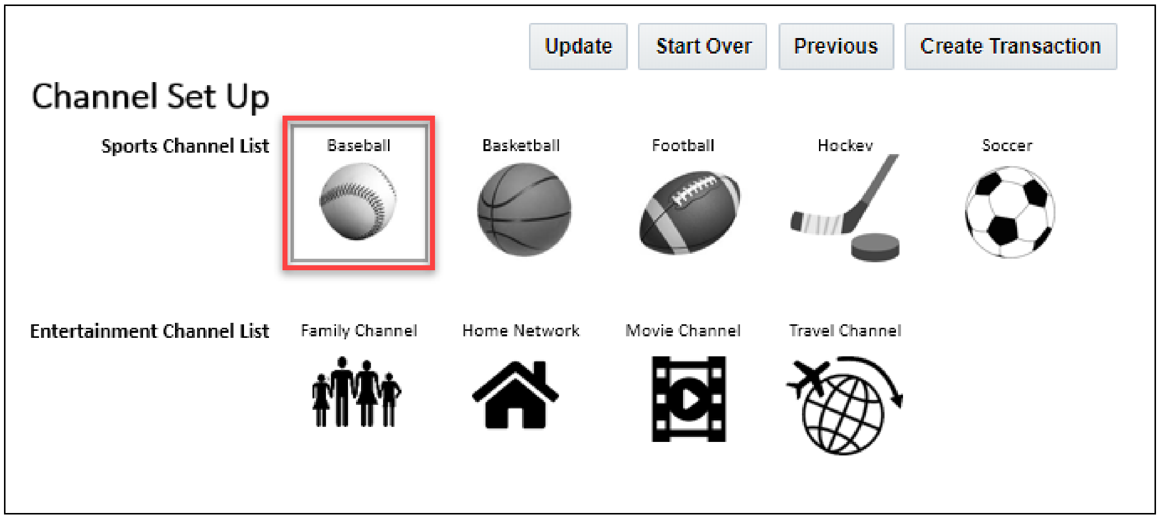
IMPORTANT: Entries in the PartNumber column that contain colons (:) represent a path to a model.

"Option Class" BOM Item Type

Beginning in Release 18B, the "Option Class" BOM item type can be used to enable inherited parent hierarchies for child BOM items. Option Class items are only added if they contain a descendant BOM item (child, grandchild, etc.) that is not an Option Class item.

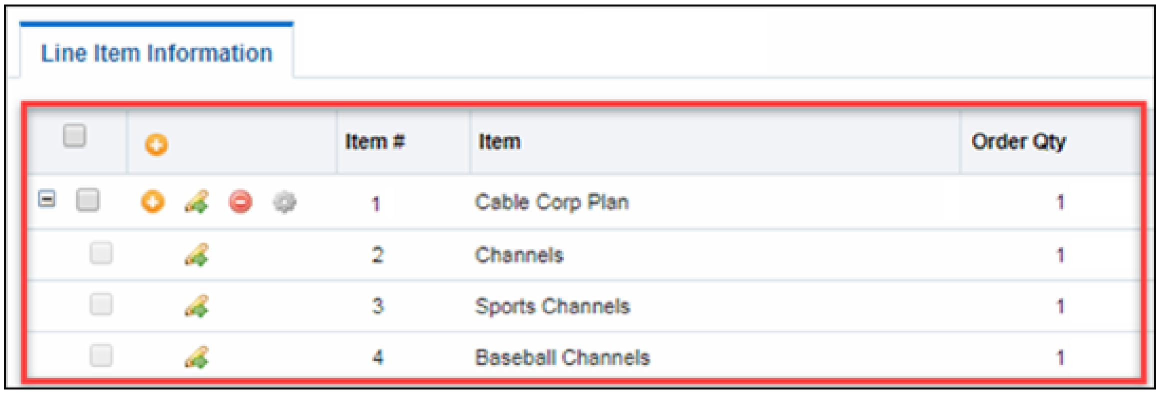
Option Class Example

The following image displays a Configuration page for a cable channel provider that offers different channel options as part of their Sports Channel and Entertainment Channel plans.



Sample Configuration Page for Cable Channel Provider

When the user selects the Baseball channel and creates a transaction, the parent items (I.e. "Channels" and "Sports Channels") are automatically added as Line Items to the Transaction. The parent items are also added to the BOM Instance, which is sent to the Fulfillment System, to complete the order.



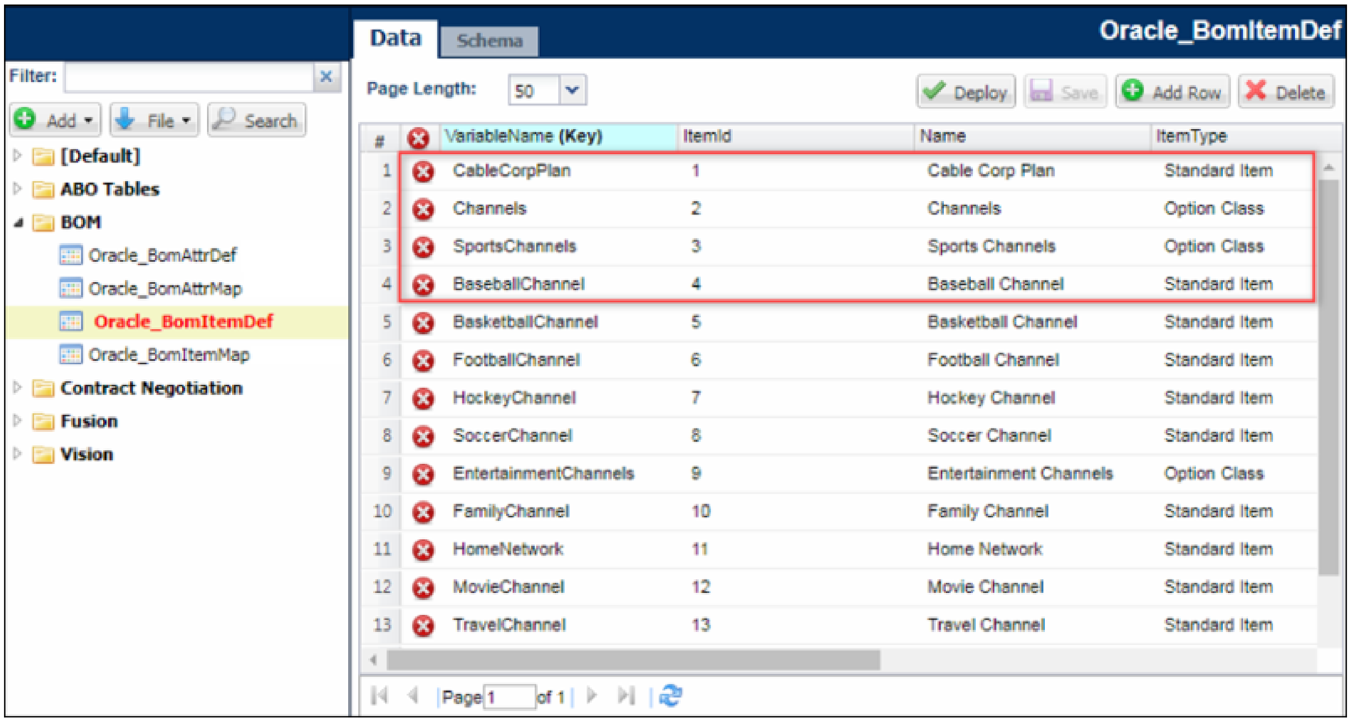
Line Item Information

Item #	Item	Order Qty
1	Cable Corp Plan	1
2	Channels	1
3	Sports Channels	1
4	Baseball Channels	1

Commerce Transaction for Sports Channel Page

For parent items the Item Type is set to "Option Class", the Item Type for child items can be set to anything other than "Option Class". The BOM parent items will only be added to the BOM Instance when a nested child item is selected. During table-based BOM mapping, "Option Class" items are not added to the BOM instance unless they have a descendant that is not an "Option Class" item.

Advanced BML BOM Mapping rules can be used to add "Option Class" items that do not have a descendant that is not an "Option Class" item.



#	VariableName (Key)	ItemId	Name	ItemType
1	CableCorpPlan	1	Cable Corp Plan	Standard Item
2	Channels	2	Channels	Option Class
3	SportsChannels	3	Sports Channels	Option Class
4	BaseballChannel	4	Baseball Channel	Standard Item
5	BasketballChannel	5	Basketball Channel	Standard Item
6	FootballChannel	6	Football Channel	Standard Item
7	HockeyChannel	7	Hockey Channel	Standard Item
8	SoccerChannel	8	Soccer Channel	Standard Item
9	EntertainmentChannels	9	Entertainment Channels	Option Class
10	FamilyChannel	10	Family Channel	Standard Item
11	HomeNetwork	11	Home Network	Standard Item
12	MovieChannel	12	Movie Channel	Standard Item
13	TravelChannel	13	Travel Channel	Standard Item

BOM Item Definition Data Table with Option Class Item Types

APPENDIX C: BOM ITEM MAPPING TABLE SCHEMA

Default table name: Oracle_BomItemMap

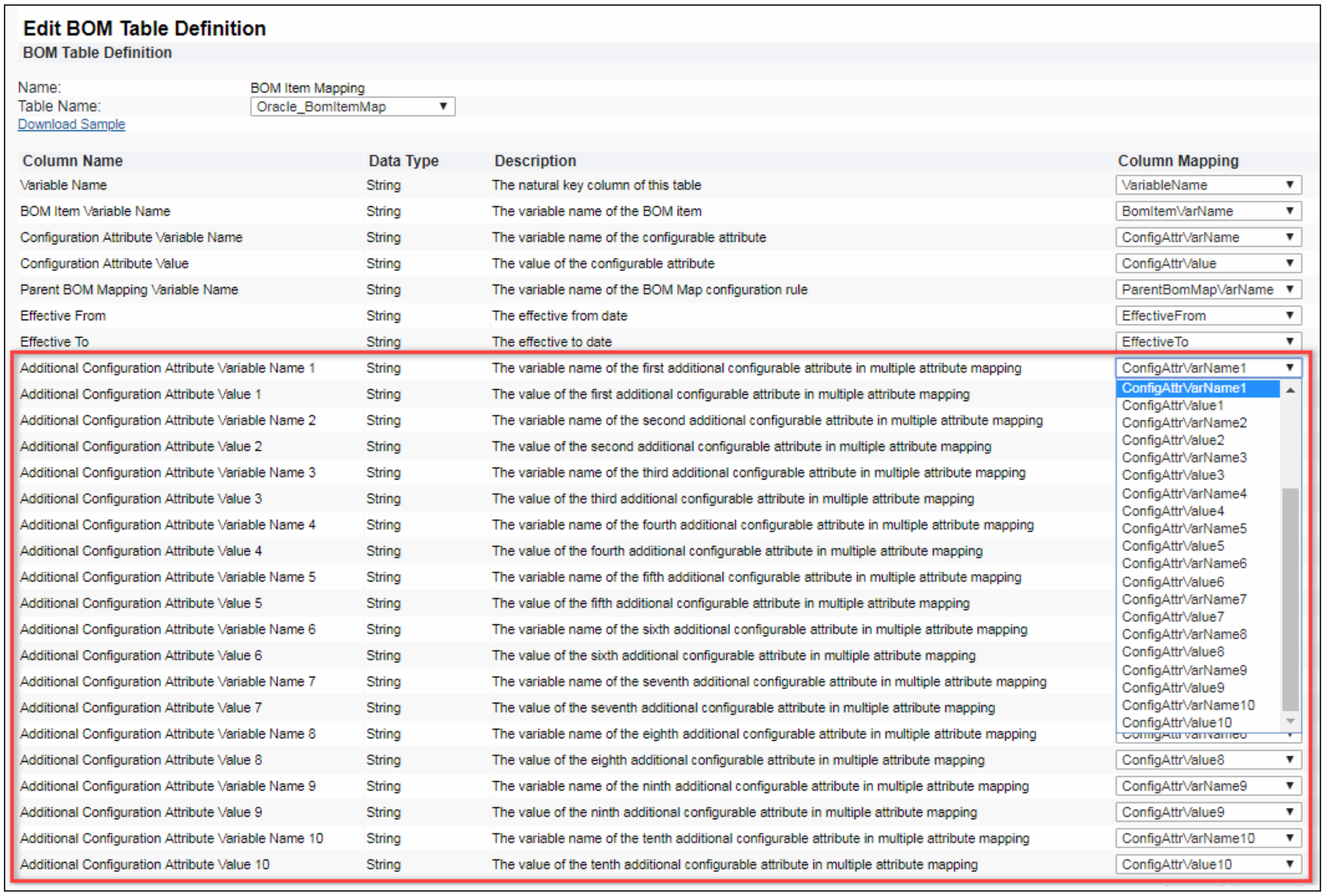
Index	Key	Name	Type	Required	Display Name	Description / Comments
	Y	`VariableName`	String	Y	Variable Name	The primary key column of this table.
		`BomItemVarName`	String	Y	BOM Item Variable Name	The variable name of the BOM item. This is a foreign key to `BomItemDef.VariableName`. The BOM item must exist otherwise, a validation error will occur.
		`ConfigAttrVarName`	String		Configurable Attribute Variable Name	The variable name of the Configuration attribute. • If this field is not null, the BOM item is created when the buy-side configurator attribute value matches the value stored in "ConfigAttrValue" column. • If this field is null, the BOM item is created unconditionally.
		`ConfigAttrValue`	String		Configurable Attribute Value	The value of the configurable attribute. This field can be null. A null values is interpreted as an empty string.
Y		`ParentBomMapVarName`	String	Y	Parent BOM Map Variable Name	The fully qualified variable name of the BOM Map Configuration rule. Formatted as: `productFamilyVarName:productLineVarName:modelVarName:configRuleVarName`. Orphan records of invalid BOM mapping rule variable names are ignored.
		`EffectiveFrom`	String		Effective From	The effective from date. Formatted as `YYY-MM-DD HH:mm:ss`
		`EffectiveTo`	String		Effective To	The effective to date. Formatted as `YYY-MM-DD HH:mm:ss`
		`ConfigAttrVarName1` *through* `ConfigAttrVarName10`	String		Configurable Attribute Variable Name 1 *through* Configurable Attribute Variable Name 10	Additional mapping attribute name fields to support mapping to multiple Configurable Attributes. Refer to Multiple Attribute Mapping for BOM Mapping Items.
		`ConfigAttrValue1` *through* `ConfigAttrValue10`	String		Configurable Attribute Value 1 *through* Configurable Attribute Value 10	Additional mapping attribute value fields to support mapping to multiple Configurable Attributes. Refer to Multiple Attribute Mapping for BOM Mapping Items.

MULTIPLE ATTRIBUTE MAPPING FOR BOM MAPPING ITEMS

Beginning in Release 18B, ten additional mapping attribute and value fields are available for the BOM Item Mapping table to support mapping to multiple Configurable Attributes. This allows the selection of multiple configuration options to add a single Line Item to a BOM-related Transaction.

Additional Configurable Attributes for BOM Item Mapping

The BOM Item Mapping Table associates BOM items to Configuration attributes. Administrators will upload or migrate BOM structures to CPQ Data Tables using CPQ Cloud's standard importing features. The Data Table can then be linked to the corresponding BOM Mapping platform tables. Activating this table enables simple Table-Based BOM Mapping Configuration rules. The items that are available for mapping are displayed in the Column Mapping drop-down menus. The list of menu options is derived from the BOM Item Mapping Data Table columns Names.



Edit BOM Table Definition

BOM Table Definition

Name:	BOM Item Mapping
Table Name:	Oracle_BomItemMap

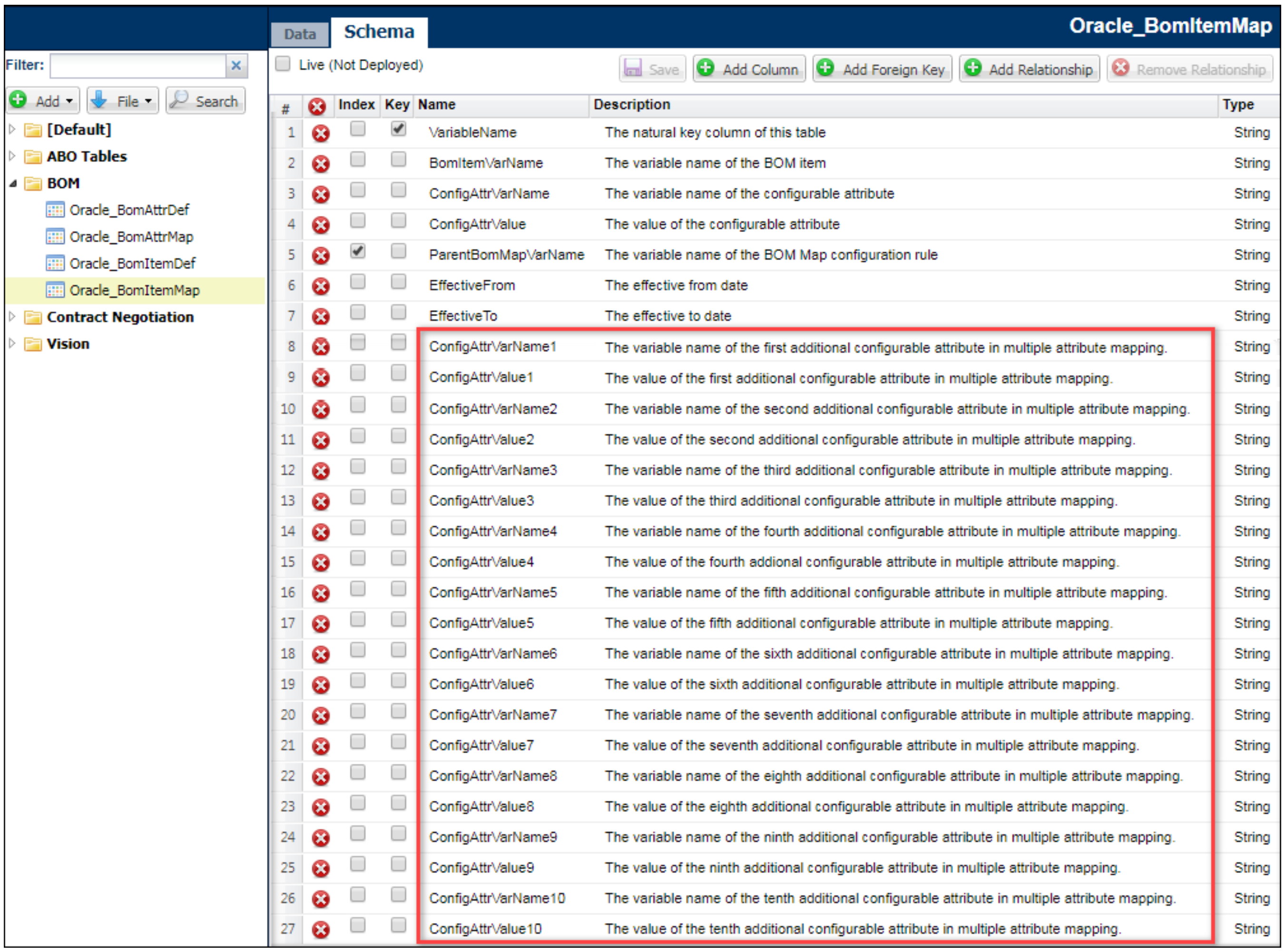
Download Sample

Column Name	Data Type	Description	Column Mapping
Variable Name	String	The natural key column of this table	VariableName
BOM Item Variable Name	String	The variable name of the BOM item	BomItemVarName
Configuration Attribute Variable Name	String	The variable name of the configurable attribute	ConfigAttrVarName
Configuration Attribute Value	String	The value of the configurable attribute	ConfigAttrValue
Parent BOM Mapping Variable Name	String	The variable name of the BOM Map configuration rule	ParentBomMapVarName
Effective From	String	The effective from date	EffectiveFrom
Effective To	String	The effective to date	EffectiveTo
Additional Configuration Attribute Variable Name 1	String	The variable name of the first additional configurable attribute in multiple attribute mapping	ConfigAttrVarName1
Additional Configuration Attribute Value 1	String	The value of the first additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 2	String	The variable name of the second additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 2	String	The value of the second additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 3	String	The variable name of the third additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 3	String	The value of the third additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 4	String	The variable name of the fourth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 4	String	The value of the fourth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 5	String	The variable name of the fifth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 5	String	The value of the fifth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 6	String	The variable name of the sixth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 6	String	The value of the sixth additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 7	String	The variable name of the seventh additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Value 7	String	The value of the seventh additional configurable attribute in multiple attribute mapping	
Additional Configuration Attribute Variable Name 8	String	The variable name of the eighth additional configurable attribute in multiple attribute mapping	ConfigAttrVarName8
Additional Configuration Attribute Value 8	String	The value of the eighth additional configurable attribute in multiple attribute mapping	ConfigAttrValue8
Additional Configuration Attribute Variable Name 9	String	The variable name of the ninth additional configurable attribute in multiple attribute mapping	ConfigAttrVarName9
Additional Configuration Attribute Value 9	String	The value of the ninth additional configurable attribute in multiple attribute mapping	ConfigAttrValue9
Additional Configuration Attribute Variable Name 10	String	The variable name of the tenth additional configurable attribute in multiple attribute mapping	ConfigAttrVarName10
Additional Configuration Attribute Value 10	String	The value of the tenth additional configurable attribute in multiple attribute mapping	ConfigAttrValue10

Edit BOM Table Definition Page for BOM Item Mapping

Configurable Attribute Columns in the BOM Item Mapping Data Table

Prior to mapping, the additional Configurable Attribute and Value fields must be added to the BOM Item Mapping Data Table Schema tab. The following image displays a BOM Item Mapping Data Table with ten additional Configurable Attribute and Value columns added to the Data Table schema.



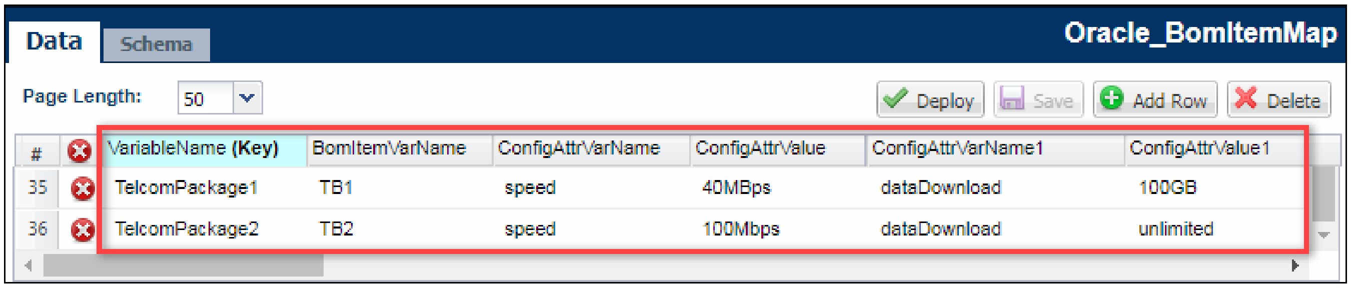
#	✖	Index	Key	Name	Description	Type
1	✖		✓	VariableName	The natural key column of this table	String
2	✖			BomItemVarName	The variable name of the BOM item	String
3	✖			ConfigAttrVarName	The variable name of the configurable attribute	String
4	✖			ConfigAttrValue	The value of the configurable attribute	String
5	✖	✓		ParentBomMapVarName	The variable name of the BOM Map configuration rule	String
6	✖			EffectiveFrom	The effective from date	String
7	✖			EffectiveTo	The effective to date	String
8	✖			ConfigAttrVarName1	The variable name of the first additional configurable attribute in multiple attribute mapping.	String
9	✖			ConfigAttrValue1	The value of the first additional configurable attribute in multiple attribute mapping.	String
10	✖			ConfigAttrVarName2	The variable name of the second additional configurable attribute in multiple attribute mapping.	String
11	✖			ConfigAttrValue2	The value of the second additional configurable attribute in multiple attribute mapping.	String
12	✖			ConfigAttrVarName3	The variable name of the third additional configurable attribute in multiple attribute mapping.	String
13	✖			ConfigAttrValue3	The value of the third additional configurable attribute in multiple attribute mapping.	String
14	✖			ConfigAttrVarName4	The variable name of the fourth additional configurable attribute in multiple attribute mapping.	String
15	✖			ConfigAttrValue4	The value of the fourth addional configurable attribute in multiple attribute mapping.	String
16	✖			ConfigAttrVarName5	The variable name of the fifth additional configurable attribute in multiple attribute mapping.	String
17	✖			ConfigAttrValue5	The value of the fifth additional configurable attribute in multiple attribute mapping.	String
18	✖			ConfigAttrVarName6	The variable name of the sixth additional configurable attribute in multiple attribute mapping.	String
19	✖			ConfigAttrValue6	The value of the sixth additional configurable attribute in multiple attribute mapping.	String
20	✖			ConfigAttrVarName7	The variable name of the seventh additional configurable attribute in multiple attribute mapping.	String
21	✖			ConfigAttrValue7	The value of the seventh additional configurable attribute in multiple attribute mapping.	String
22	✖			ConfigAttrVarName8	The variable name of the eighth additional configurable attribute in multiple attribute mapping.	String
23	✖			ConfigAttrValue8	The value of the eighth additional configurable attribute in multiple attribute mapping.	String
24	✖			ConfigAttrVarName9	The variable name of the ninth additional configurable attribute in multiple attribute mapping.	String
25	✖			ConfigAttrValue9	The value of the ninth additional configurable attribute in multiple attribute mapping.	String
26	✖			ConfigAttrVarName10	The variable name of the tenth additional configurable attribute in multiple attribute mapping.	String
27	✖			ConfigAttrValue10	The value of the tenth additional configurable attribute in multiple attribute mapping.	String

Oracle BomItemMap Data Table with Additional Configurable Attribute Columns

Multiple Attribute Mapping Example

After the additional Configurable Attribute and Value fields have been added to the BOM Item Mapping Data Table schema, the Data Table can be populated with the additional Configurable Attributes and Values. BOM Mapping can now evaluate the combination of several different Configuration attributes and then add a single child part to the quote based on multiple attribute selections.

For example: The following BOM Item Mapping Data displays two BOM items with "speed" and "dataDownload" attributes. If a user selects a speed of 40Mbps and a data download option of 100GB, then the TelecomPackage1 part will be added to the quote.



#	VariableName (Key)	BomItemVarName	ConfigAttrVarName	ConfigAttrValue	ConfigAttrVarName1	ConfigAttrValue1
35	TelcomPackage1	TB1	speed	40MBps	dataDownload	100GB
36	TelcomPackage2	TB2	speed	100Mbps	dataDownload	unlimited

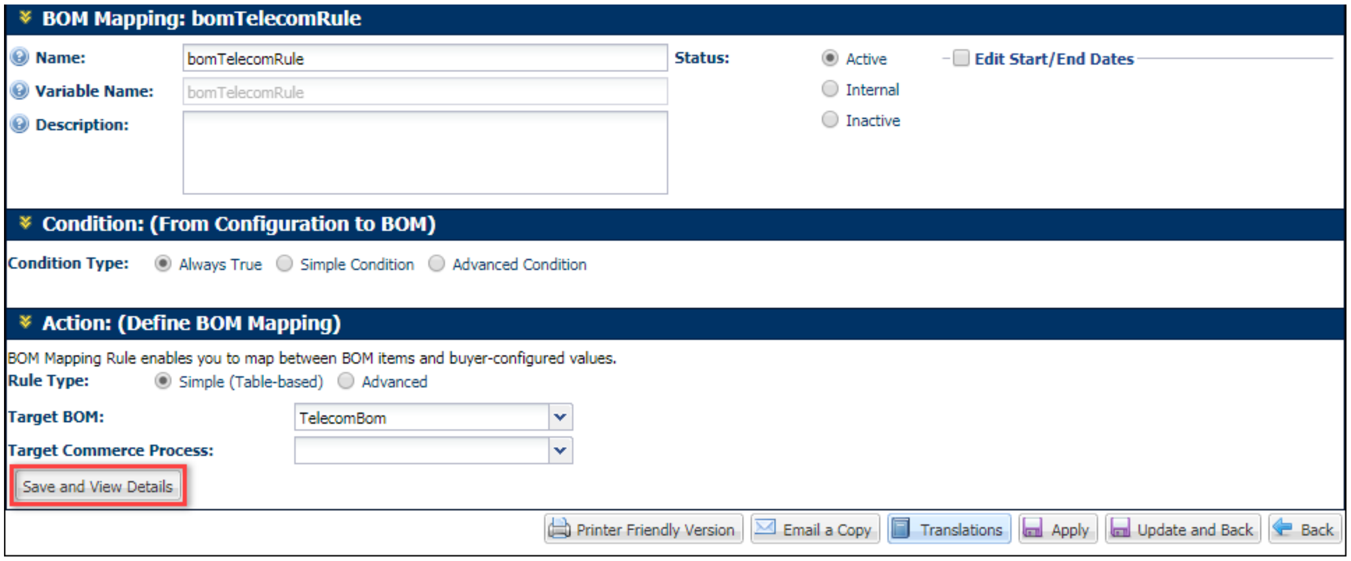
BOM Item Mapping Data Table

When additional Configurable attributes have been added and mapped, administrators can view the mapped attributes in the BOM Item Mapping Administration List and BOM Item Mapping Administration pages.

View BOM Item Mapping Administration Pages

Complete the following steps to view additional configurable attributes.

1. Navigate to the Admin Home Page.
2. Under **Products**, click **Catalog Definition**.
   The **Supported Products** page opens.
3. Select **Product Families** from the **Navigation** drop-down.
4. Click **List**.
   The **Supported Product Families** page opens.
5. Next to the applicable Product Family, select **Product Lines** from the **Navigation** drop-down.
6. Click **List**.
   The **Product Line Administration List** page opens.
7. Next to the applicable Product Line, select **Models** from the **Navigation** drop-down.
8. Click **List**.
   The **Model Administration List** page opens.
9. Next to the applicable Model, select **BOM Mapping** from the **Navigation** drop-down.
10. Click **List**.
    The **BOM Mapping: Rules List** page opens.
11. Select the applicable rule.
    The **BOM Mapping Rule** page opens.



BOM Mapping: bomTelecomRule

Name:	bomTelecomRule	Status:	[✓ Active] Internal Inactive
Variable Name:	bomTelecomRule		Edit Start/End Dates
Description:			

Condition: (From Configuration to BOM)

Condition Type: [✓ Always True] Simple Condition Advanced Condition

Action: (Define BOM Mapping)

BOM Mapping Rule enables you to map between BOM items and buyer-configured values.

Rule Type: [✓ Simple (Table-based)] Advanced

Target BOM: TelecomBom

Target Commerce Process:

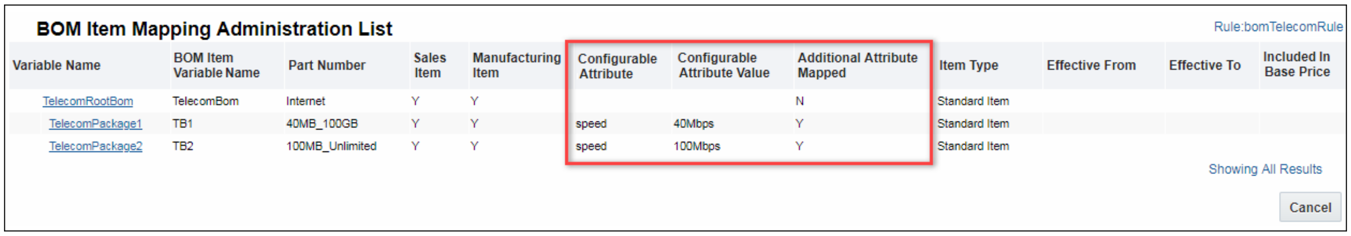
Save and View Details

Printer Friendly Version | Email a Copy | Translations | Apply | Update and Back | Back

BOM Mapping Rule Page

12. Click **Save and View Details**.
    The **BOM Mapping Administration List** page opens.

    In the following image, the mapped Configuration Attributes are shown. If there are any Additional Configurable Attributes, the **Additional Attribute Mapped** column will be set to **Y**.



BOM Item Mapping Administration List

Rule:bomTelecomRule

Variable Name	BOM Item Variable Name	Part Number	Sales Item	Manufacturing Item	Configurable Attribute	Configurable Attribute Value	Additional Attribute Mapped	Item Type	Effective From	Effective To	Included In Base Price
TelecomRootBom	TelecomBom	Internet	Y	Y			N	Standard Item			
TelecomPackage1	TB1	40MB_100GB	Y	Y	speed	40Mbps	Y	Standard Item			
TelecomPackage2	TB2	100MB_Unlimited	Y	Y	speed	100Mbps	Y	Standard Item			

Showing All Results

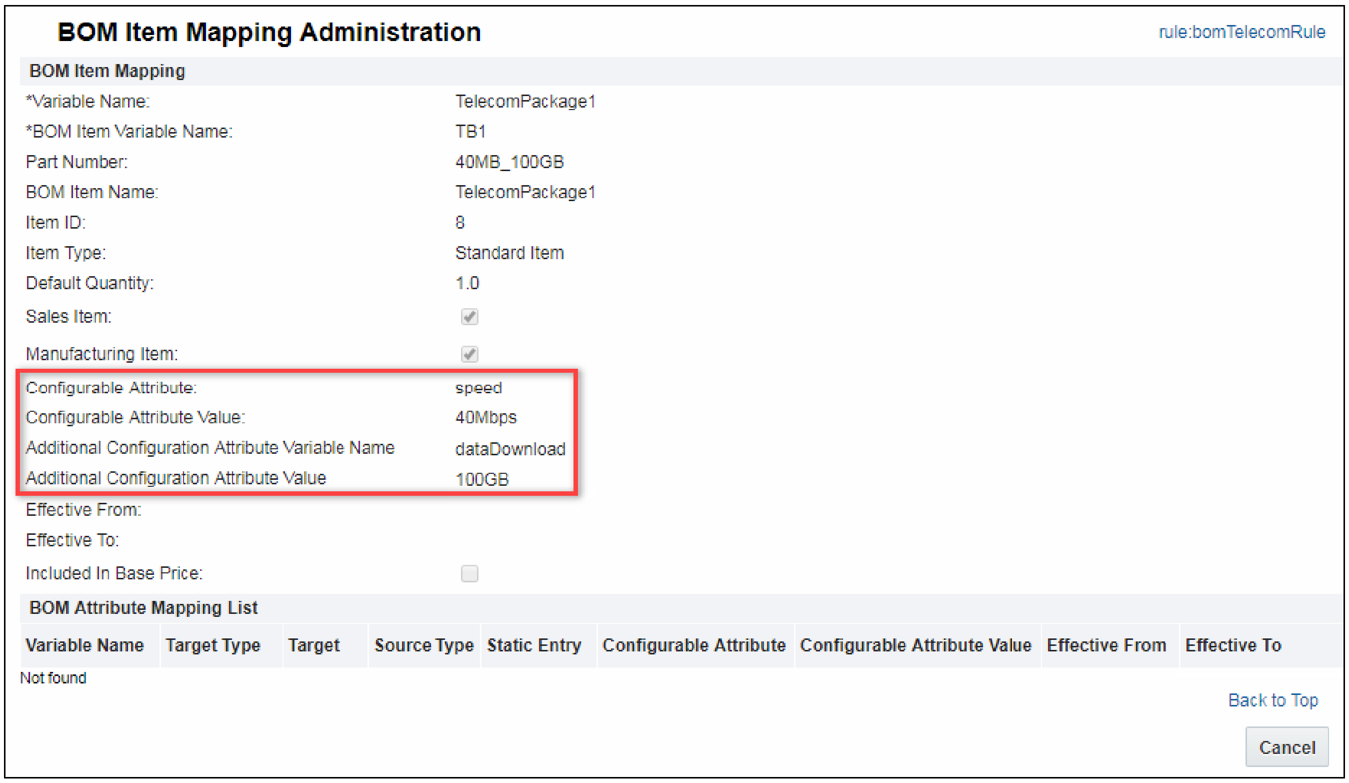
Cancel

BOM Item Mapping Administration List Page

13. Select the applicable Variable Name to view the Additional Configurable Attributes.
    The **BOM Item Mapping Administration** page opens.

    The associated Configurable Attribute and Additional Configurable Attributes are displayed.

    For example: The BOM Item Mapping details for "TelecomPackage1" are displayed in the following image. When "40Mbps" is selected for the "speed" attribute and "100GB" is selected for the "dataDownload" attribute, the TelecomPackage1 part will be added to the quote.



BOM Item Mapping Administration

rule:bomTelecomRule

BOM Item Mapping

*Variable Name:	TelecomPackage1
*BOM Item Variable Name:	TB1
Part Number:	40MB_100GB
BOM Item Name:	TelecomPackage1
Item ID:	8
Item Type:	Standard Item
Default Quantity:	1.0
Sales Item:	✓
Manufacturing Item:	✓
Configurable Attribute:	speed
Configurable Attribute Value:	40Mbps
Additional Configuration Attribute Variable Name	dataDownload
Additional Configuration Attribute Value	100GB
Effective From:	
Effective To:	
Included In Base Price:	☐

BOM Attribute Mapping List

Variable Name	Target Type	Target	Source Type	Static Entry	Configurable Attribute	Configurable Attribute Value	Effective From	Effective To
Not found								

Back to Top

Cancel

BOM Item Mapping Administration Page

APPENDIX D: BOM ATTRIBUTE DEFINITION TABLE SCHEMA

Default table name: Oracle_BomAttrDef

Index	Key	Name	Type	Required	Display Name	Description / Comments
Y	Y	`VariableName`	String	Y	Variable Name	The primary key column of this table.
		`Name`	String		Name	Display name of the attribute, in the site base language.
		`Values`	String		Values	Domain values of the attribute if the attribute is a menu type. Values of the attribute separated by ~. Applicable only if the attribute is a menu type, otherwise this value is empty. The values use the site base language.
		`DisplayValues`	String		Display Values	Display values of the attribute if the attribute is a menu type separated by ~. Must be empty if Values column is empty. If Values is not empty, the number of display values as separated by ~ must match that of values. The values use the site base language,
		`EffectiveFrom`	String		Effective From	The effective from date. Formatted as: `YYYY-MM-DD HH:mm:ss`
		`EffectiveTo`	String		Effective To	The effective to date. Formatted as : `YYYY-MM-DD HH:mm:ss`
		`BomItemVarName`	String	Y	BOM Item Variable Name	The variable name of the parent BOM item. Key to `bomAttrDef.VariableName` Orphan records of dirty BOM item variable names are ignored.
		`DataType`	String		Data Type	The data type of the attribute values. Legal Values: String, Integer, Float, Date, Boolean
Y		`RootBomItemVarName`	String	Y	Root BOM Item Variable Name	The variable name of the root BOM item. Key to `BomItemDef.VariableName` of the root BOM item definition. Orphan records of dirty root BOM item variable names are ignored.

APPENDIX E: BOM ATTRIBUTE MAPPING TABLE SCHEMA

Default table name: Oracle_BomAttrMap

Index	Key	Name	Type	Required	Display Name	Description / Comments
	Y	`VariableName`	String	Y	Variable Name	This is the primary key for the table.
		`TargetType`	String	Y	Target Type	The mapping target type. Acceptable values are: • BOM_ATTTRIBUTE: mapped to a BOM item attribute • LINE_ATTRIBUTE: mapped to commerce (quote) line attribute • QUANTITY: The BOM line item quantity.
		`TargetVariableName`	String		Target Variable Name	The variable name of the target BOM attribute or commerce line attribute. Depending on the target type, this is the variable name of the target BOM attribute or line attribute.
		`SourceType`	String	Y	Source Type	The mapping source type. Acceptable values are: • STATIC_ENTRY: The target is always set to the value in StaticEntry • CONDITIONAL_STATIC_ENTRY: The target is set to the value in Static Entry if the configuration attribute at runtime matches the value to ConfigAttrValue. • CONFIG_ATTRIBUTE: the target is set to the value of the configuration attribute
		`StaticEntry`	String		Static Entry	A static value that is used as the source of mapping
		`ConfigAttrVarName`	String		Configurable Attribute Variable Name	The variable name of the configurable attribute.
		`ConfigAttrValue`	String		Configurable Attribute Value	The value of the configurable attribute.
		`EffectiveFrom`	String		Effective From	The effective from date. Formatted as `YY-MM-DD HH:mm:ss`
		`EffectiveTo`	String		Effective To	The effective to date. Formatted as `YYY-MM-DD HH:mm:ss`

Index	Key	Name	Type	Required	Display Name	Description / Comments
		`BomItemMapVarName`	String	Y	BOM Item Map Variable Name	The variable name of the parent BOM item map. Key to `BomItemMap.VariableName.` Orphan records of invalid BOM item mapping variable names are ignored.
Y		`RootBomMapVarName`	String	Y	Root BOM Map Variable Name	The fully qualified variable name of the BOM Mapping Configuration rule. Formatted as: `productFamilyVarName:productLineVarName:modelVarName:configRuleVarName`. Orphan records of invalid root BOM mapping rule variable names are ignored.

APPENDIX F: BOM ATTRIBUTE TRANSLATION TABLE SCHEMA

Default table name: Oracle_BomAttr_Tr

Index	**Key**	**Name**	**Type**	**Required**	**Display Name**	**Description / Comment**
Y	Y	`VariableName`	String	Y	Variable Name	The primary key column of this table.
		`Language`	String	Y	Language	The language in which the translations of the given record is defined. For example: de for German, zh_CN for Chinese Simplified
		`BomAttrVariableName`	String	Y	BOM Attribute Variable Name	The variable name of the parent BOM attribute. Key to `BomAttrDef.VariableName.` Orphan records of dirty BOM attribute variable names are ignored.
		`Name`	String		Name	The translation of display name, corresponding to `BomAttrDef.Name`.
		`Values`	String		Values	The translation of values of the attribute, corresponding to `BomAttrDef.Values`
Y		`RootBomItemVarName`	String	Y	Root BOM Item Variable Name	The variable name of the root BOM item. Key to `BomItemDef.VariableName` of the root BOM item definition. Orphan records of dirty root BOM item variable names are ignored.

APPENDIX G: BOM MAPPING CONFIGURABLE ARRAY ATTRIBUTE RESTRICTIONS

The following restrictions on configurable array attribute mapping are enforced, primarily to maintain one-to-one mapping between a row in the array set and a BOM item.

Sequence	Mapping	Rule	Sample Validation Error Message
1	BOM Item Mapping	A BOM item mapping can use more than one configurable array attribute, but they must be all from the same array set.	The following example is illegal: `BOM1|drinkType=Coffee|CarType=SUV` The BOM item mapping BOM1 is mapped from the following configuration array attributes in different array sets: Drink, Car.
2	BOM Item Mapping	A configurable array attribute should not be used more than once in a single BOM item mapping.	The following example is illegal: `BOM1|DrinkType=Coffee|DrinkType=Tea` The configuration attribute DrinkType is used more than once in the BOM item mapping BOM1.

3	BOM Item Mapping	If two BOM item mappings use configurable array attributes from the same array set, the array attributes must be the same in both BOM item mappings.	The following example is illegal:
			`BOM1|drinkType=Coffee|drinkSize=Medium`
			`BOM2|drinkType=Tea`
			BOM item mapping BOM1 and BOM2 must be mapped from same configuration array attributes from the same array set.

4	BOM Item Mapping	If two BOM item mappings use configurable array attributes, the combination of array attributes and values must be unique.
		In other words, the same combination of configurable array attributes and values cannot be used to map to two different BOM items.
			The following example is illegal:
			`BOM1|drinkType=Coffee|drinkSize=Medium`
			`BOM2|drinkType=Coffee|drinkSize=Medium`
			BOM item mapping BOM1 and BOM2 are mapped from exactly same values for configuration array attributes.

Sequence	Mapping	Rule	Sample Validation Error Message

5	BOM Item Mapping	If a BOM item mapping uses a configuration array attribute, it is illegal for one of its hierarchical parent BOM item mapping (including parent, grandparent, ...) to use configurable array attributes (whether or not from the same array set).	The following example is illegal:
			`BOM1|drinkType=Coffee`
			`BOM2 (parent: BOM1) drinkSize=Medium`
			Configuration array attributes cannot be used in nested BOM item mapping. Configuration attribute drinkSize is an array attribute, but BOM item mapping coffeeMapping, a hierarchical parent, is already mapped with array attribute(s) drinkType.

6	BOM Attribute Mapping	If an attribute mapping uses a configurable array attribute, its containing BOM item mapping must use at least one configurable array attributes.	The following example is illegal: `BOM1|eventType=FundRaising (attribute mapping: drinkType=Coffee)` Cannot map to an array type configuration attribute drinkType as the parent BOM item mapping is not mapped to an array type attribute.
7	BOM Attribute Mapping	If an attribute mapping uses a configurable array attribute, the containing BOM item mapping must use one or more configurable array attributes from the same array set.	The following example is illegal: `BOM1|carType=SUV (attribute mapping: drinkType=Coffee)` Cannot map to a configuration attribute in the array set Car as its parent BOM item mapping maps to an attribute in a different array set Dessert.
8	BOM Attribute Mapping	Attribute mapping cannot use the same configurable array attributes that are used in its containing BOM item mapping.	The following example is illegal: `BOM1|drinkType=Coffee (attribute mapping: drinkType=Coffee)` Cannot map to the same array attribute drinkType as used in the parent BOM item mapping.

9	BOM Attribute Mapping	If configurable array attributes are used in BOM attribute mappings, the same array attributes should be used in BOM attribute mappings of all BOM item mappings that use the same configurable array set. This is similar to Rule (3) of BOM item mapping.	The following example is illegal:
			`BOM1|drinkType=Coffee (attribute mapping: drinkSize=Medium)`
			`BOM2|drinkType=Tea (no attribute mapping)`
			Validation error will be added in a future release.

IMPORTANT: BOM Attributes can be mapped to both Configurable Array Attributes and non-array type Configurable Attributes.

APPENDIX H: BML FUNCTIONS FOR BOM MAPPING

The following BML functions to support the BOM Mapping feature: getbom, savebom, convertbomtoflat, and convertbomtohier.

GET BOM BML FUNCTION

For fulfillment system integrations, the getbom function retrieves the saved Sales BOM or Manufacturing BOM from a quote, to submit to the fulfillment system for order fulfillment.

For Subscription Ordering, the getbom function retrieves the saved Sales BOM from open orders.

getbom			
Parameters	`bsId`	Integer	Use this parameter to specify the Commerce Transaction ID.
	`lineNumber`	Integer	Use this parameter to specify the document number of the model line. The line number also represents the root BOM line in the quote.
	`lineFields`	String	Use this parameter to identify additional line attributes fetched from the quote line, then stored in the returned BOM instance.
			Optional, the default value is null if not provided.
	`validateBomModel`	Boolean	Use this parameter to validate the returned BOM against the latest BOM item definition. Validation will: • Verify the BOM instance tree (parts and hierarchy) against the BOM item definition. • Populate the BOM item variable names. • Correct the BOM instance hierarchy according to the latest definition. • Exclude items that no longer exist in the latest definition.
			Optional, the default value is true if not provided.
	`flattenChildItems`	Boolean	Use this parameter to flatten child items and return all descendant BOM items as direct children of the root BOM item.
			Optional, the default value is false if not provided.
	`isSalesBom`	Boolean	This parameter returns a Sales BOM if true, and a Manufacturing BOM if false.
			Optional, the default is true if not provided.
Syntax	`Json getbom(Integer bsId, Integer lineNumber [, String[] lineFields [, Boolean validateBomModel [, Boolean flattenChildItems [, Boolean isSalesBom]]]])`		
Sample Input	`bsId`	`18430319`	
	`lineNumber`	`2`	

getbom

Sample Return

```
{
    "partNumber": "part49",
    "quantity": 10,
    "id": "BOM_root",
    "parentId": "",
    "attributes": {},
    "fields": {
        "_line_bom_level": "0"
    },
    "explodedQuantity": 10,
    "category": "sales",
    "variableName": "root",
    "definition": {
        "SequenceNum": 814,
        "ItemId": "814",
        "ItemType": "Standard Item",
        "Optional": "Y"
    },
    "children": [
        {
            "partNumber": "part50",
            "quantity": 5,
            "id": "BOM_text_bom",
            "parentId": "BOM_root",
            "attributes": {},
            "fields": {
                "_line_bom_level": "1"
            },
            "explodedQuantity": 50,
            "variableName": "text_bom",
            "definition": {
                "SequenceNum": 815,
                "ItemId": "815",
                "ItemType": "Standard Item",
                "Optional": "Y"
            }
        }
    ]
}
```

SAVE BOM BML FUNCTION

The savebom function saves a BOM into a quote without Configuration attributes and returns the document number of the saved quote. For Subscription Ordering, the savebom function saves discontinued assets into a quote.

savebom			
Parameters	`bsID`	Integer	Use this parameter to specify the Commerce Transaction ID.
	`bomJson`	JSON	Use this parameter to hold the BOM instance JSON data.
Syntax	`Integer savebom(Integer bsId, Json bomJson)`		
Sample Input	`bsId`	`18430319`	
	`bomJson`	(see below)	

```
{
    "partNumber": "part49",
    "id": "BOM_root",
    "quantity": 10,
    "parentId": "",
    "attributes": {},
    "fields": {"_line_bom_level": "0"},
    "explodedQuantity": 10,
    "category": "sales",
    "variableName": "root",
    "definition": {
      "SequenceNum": 814,
      "ItemId": "814",
      "ItemType": "Standard Item",
      "Optional": "Y"
   },  "children":[{
      "partNumber": "part50",
      "quantity": 5,
      "id": "BOM_text_bom",
      "parentId": "BOM_root",
      "attributes": {},
      "fields": {"_line_bom_level": "1"},
      "explodedQuantity": 50,
      "variableName": "text_bom",
      "definition": {
         "SequenceNum": 815,
         "ItemId": "815",
         "ItemType": "Standard Item",
         "Optional": "Y"
      }
   }]
}
```

Sample Return	5

GET CONFIGURATION BOM BML FUNCTION

Retrieves the configbom stored via the saveConfigBom API and the configBom created via an external client application Configurator UI session. The library function extracts and returns a client integration BOM instance from the CPQ Cloud configBomInstance resource using the "configId".

getConfigbom			
Parameters	`configId`	Integer	The Configuration ID for the client side integration action This 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. • For other actions such as Terminate, Renew, Suspend, and Resume order, RESTful calls generated from the saveBomConfig BML function return the lineId.
	`flattenChildProducts`	Boolean	Use this parameter to flatten child items and return all descendant BOM items as direct children of the root BOM item.
			Optional, the default value is false if not provided.
Syntax	`getconfigbom(configId [, flattenChildProducts])`		

SAVE CONFIGURATION BOM BML FUNCTION

Allows users to save the BOM for non-configurator UI integration scenarios such as suspend, resume, and terminate. The library function saves a client integration BOM instance and a "configId" to the CPQ Cloud configBomInstance resource and returns a "configId".

saveConfigbom			
Parameters	`configBomJson`	JSON	The configuration BOM JSON to save.
Syntax	`getconfigbom(configId [, flattenChildProducts])`		
Return	`configId`	Integer	The Configuration Id for the client side integration action.

CONVERT A HIERARCHICAL BOM INTO A FLATTENED BOM BML FUNCTION

The convertbomtoflat function converts a hierarchical BOM into a flattened BOM. A flat BOM stores all descendants as direct children, including children, grandchildren, etc. Flattened BOMs are easier to process.

IMPORTANT: The input "bomJson" is not modified and will remain in a hierarchical BOM format.

convertbomtoflat			
Parameter	`bomJson`	JSON	Use this parameter to hold the JSON target.
Syntax	`Json convertbomtoflat(Json bomJson)`		
Sample Input	`bomJson`	see below	

```
{"partNumber":"part1",
   "quantity":1,
   "id":"Bom1",
   "parentId":"",
   "children":[
      {"partNumber":"part2",
      "quantity":2,"id":"Bom2",
      "parentId":"","children":[
         {"partNumber":"part4",
         "quantity":4,
         "id":"Bom4",
         "parentId":""
            },
            {"partNumber":"part5",
            "quantity":5,
            "id":"Bom5",
            "parentId":""
            }
   ]
      },
      {"partNumber":"part3",
      "quantity":3,
      "id":"Bom3",
      "parentId":""
      }
   ]
}
```

convertbomtoflat	

Sample Return	`{`
	`   "partNumber": "testbed:systemConfiguration:rootSystem",`
	`   "quantity": 1,`
	`   "isModel": true,`
	`   "id": "BOM_SysConfigRoot",`
	`   "parentId": "",`
	`   "explodedQuantity": 1,`
	`   "category": "sales",`
	`   "children": [{`
	`         "partNumber":`
	`"testbed:systemConfiguration:conditionalChildren",`
	`         "quantity": 1,`
	`         "isModel": true,`
	`         "id": "BOM_SysConfigNestKid3",`
	`         "parentId": "BOM_SysConfigNest",`
	`         "explodedQuantity": 1`
	`      }, {`
	`         "partNumber": "testbed:systemConfiguration:noBOMRule",`
	`         "quantity": 1,`
	`         "isModel": true,`
	`         "id": "BOM_SysConfigNestKid2",`
	`         "parentId": "BOM_SysConfigNest",`
	`         "explodedQuantity": 1`
	`      }, {`
	`         "partNumber":`
	`"testbed:systemConfiguration:differentConfiguration",`
	`         "quantity": 1,`
	`         "isModel": true,`
	`         "id": "BOM_SysConfigNestKid1",`
	`         "parentId": "BOM_SysConfigNest",`
	`         "explodedQuantity": 1`
	`      }, {`
	`         "partNumber": "testbed:systemConfiguration:nestedHierarchies",`
	`         "quantity": 1,`
	`         "isModel": true,`
	`         "id": "BOM_SysConfigNest",`
	`         "parentId": "BOM_SysConfigRoot",`
	`         "explodedQuantity": 1`
	`      }`
	`   ]`
	`}`

CONVERT A FLATTENED BOM INTO A HIERARCHICAL BOM BML FUNCTION

The convertbomtohier function converts a flattened BOM into a hierarchical BOM. Occasionally, administrators flatten hierarchical BOMs for easier processing; this function returns the processed flattened BOM back into a hierarchical BOM.

IMPORTANT: The input "bomJson" is not modified and will remain in a hierarchical BOM format.

convertbomtohier			
Parameter	`bomJson`	JSON data type	Use this parameter to hold the JSON target.
Syntax	`Json convertbomtohier(Json bomJson)`		
Sample Input	`bomJson`	`{ "partNumber": "part1", "quantity": 1, "id": "Bom1", "parentId": "", "children": [{ "partNumber": "part2", "quantity": 2, "id": "Bom2", "parentId": "Bom1" }, { "partNumber": "part3", "quantity": 3, "id": "Bom3", "parentId": "Bom1" }, { "partNumber": "part4", "quantity": 4, "id": "Bom4", "parentId": "Bom2" }, { "partNumber": "part5", "quantity": 5, "id": "Bom5", "parentId": "Bom2" }] }`	

convertbomtohier

Sample Return

```
{
   "partNumber": "testbed:systemConfiguration:rootSystem",
   "quantity": 1,
   "isModel": true,
   "id": "BOM_SysConfigRoot",
   "parentId": "",
   "explodedQuantity": 1,
   "category": "sales",
   "children": [{
         "partNumber": "testbed:systemConfiguration:nestedHierarchies",
         "quantity": 1,
         "isModel": true,
         "id": "BOM_SysConfigNest",
         "parentId": "",
         "explodedQuantity": 1,
         "children": [{
               "partNumber":
"testbed:systemConfiguration:conditionalChildren",
               "quantity": 1,
               "isModel": true,
               "id": "BOM_SysConfigNestKid3",
               "parentId": "",
               "explodedQuantity": 1
            }, {
               "partNumber": "testbed:systemConfiguration:noBOMRule",
               "quantity": 1,
               "isModel": true,
               "id": "BOM_SysConfigNestKid2",
               "parentId": "",
               "explodedQuantity": 1
            }, {
               "partNumber":
"testbed:systemConfiguration:differentConfiguration",
               "quantity": 1,
               "isModel": true,
               "id": "BOM_SysConfigNestKid1",
               "parentId": "",
               "explodedQuantity": 1
            }
         ]
      }
   ]
}
```

APPENDIX I: BML FUNCTIONS FOR SYSTEM CONFIGURATION

The following BML functions are used to retrieve System Configuration attribute values from other configured models within the system. These BML functions are available in Library functions, Util libraries, Configuration Advanced rules, Commerce Advanced rules, Step Transitions, Advanced Step Notifications, and mass update of Transactions.

IMPORTANT:

- These functions should only be used with System Configurations. System Configurations are BOM hierarchies that contain one or more nested child models.
- BML functions `getSystemMultipleAttrValues`, `getSystemAttrValues`, `getSystemData` will not return the value of HTML attributes. This behavior is consistent with other configuration rules that do not allow the selection of HTML attributes.

GET SYSTEM DATA BML FUNCTION

This BML function returns a JSON object representing the entire System Configuration for the current Transaction.

getSystemData	
Syntax	`Json getsystemdata()`

Example	`systemJson = json();`
	`systemJson = getsystemdata();`

IMPORTANT:

- When not in the context of a Transaction, an empty JSON object is returned.
- If System Configuration Data does not exist, an empty JSON object is returned.
- This function will also return an empty JSON object if a System Configuration has not yet been configured.
- The empty JSON object should be handled accordingly.

GET SYSTEM ATTRIBUTES VALUES BML FUNCTION

This BML function returns a string containing a single attribute's values from a System Configuration.

getSystemAttrValues			
Parameter	`jsonPath`	String	A string containing the JSON path
Syntax	`getsystemattrvalues(String jsonPath)`		

Example	`modelValue = String[];`
	`modelValue = getsystemattrvalues("$.configAttributes.attributeVarname");`		

IMPORTANT:

- If the JSON Path does not return an array of single values, empty string array ("[]") will be returned.
- The function will also return empty values for any models that are yet to be configured and paths that do not return values. The empty array should be handled accordingly.

GET SYSTEM MULTIPLE ATTRIBUTE VALUES BML FUNCTION

This BML function returns dictionary key and value string arrays containing attribute values from a System Configuration.

getSystemMultipleAttrValues	
Parameter	Dictionary with String keys and String values. The values are expected to be JsonPath Expressions. • The key should be an identifier for the attributes identified in the associated value. • The values are expected to be JSON Path expressions that identify the location of the attribute in the fully expanded system definition (BOM).
Syntax	`Dictionary<String[]> getsystemmultipleattrvalues(Dictionary<String>)`
Return	Returns a Dictionary (key: String, value: String[]) containing attribute values from a System Configuration. • The keys will be the identifiers provided in the input Dictionary. • The values will be the configured attribute values of the attribute(s) at the JSON Path expressions associated with the input key.

Example	`jsonPaths = dict("string");`
	`put(jsonPaths, "attributeVarname", "$.configAttributes.attributeVarname");`
	`put(jsonPaths, "childAttributeVarname",`
	`"$.children[*].configAttributes.childAttributeVarname");`
	`interModelValues = dict("string[]");`
	`interModelValues = getsystemmultipleattrvalues(jsonPaths);`
	`values = String[];`
	`values = get(interModelValues, "attributeVarname");`

IMPORTANT:

- If a JSON Path does not return an array of single values, that key-value pair will result in an empty array.
- The function will return empty arrays for models that have not been configured and paths that do not return values. The empty arrays should be handled accordingly.

APPENDIX J: BOM JSON OBJECTS AND EXAMPLES

BOM ITEM JSON OBJECT

The following table describes BOM Item fields in the JSON object.

Field	Data Type	Required	Description
`id`	String	N	The instance id of the BOM item.
`variableName`	String	N	The variable name of the BOM Item Definition. Required when BML-based rules are used to create the BOM instance during Configuration.
`category`	String	N	Legal values: sales, manufacturing. Designates if a BOM is a Sales BOM or a Manufacturing BOM. Applicable to the root BOM item only.
`effectiveDate`	String	N	The effective date. In ISO format yyyy-MM-ddTHH:mm:ssZ. Applicable to the root BOM item only.
`parentId`	String	N	The parent BOM item instance id. The field is not defined unless children items are flattened.
`partNumber`	String	Y	The Part Number of the item. • A sales item exists in the CPQ Parts Database and the back-end fulfillment system with the same Part Number. • A manufacturing item only exists in the back-end fulfillment system; it is not required in the CPQ Parts Database.
`quantity`	Long	Y	The line item quantity
`isModel`	Boolean	N	Specifies if the BOM item is a model item. Used in the output BOM item instance for system configuration. The flag does not need to be explicitly set in BOM mapping BML.
`explodedQuantity`	Long	N	The exploded line quantity
`children`	Array	N	The child BOM items. A JSON array of BOM items.
`attributes`	Object	N	A JSON object that stores the BOM attributes. The format is defined in the BOM Attributes section.
`definition`	Object	N	Additional BOM Item Definition items required when submitting BOMs to back-end fulfillment systems, including SequenceNum, ItemId, ItemType, Optional, and IncludedInBasePrice. The node is populated in the output BOM instance for BOM item mapping and the getBOM BML function when either validateBomModel is true or isSalesBom is false. Do not explicitly populate this node in the BOM mapping BML.
`fields`	Object	N	Optional fields for ABO. The JSON object stores fields as key-value pairs. If a BOM is saved to a quote, the fields are mapped to line attributes. Field names must be line attribute variable names.

BOM ITEM ATTRIBUTE JSON OBJECT

The attributes of BOM items are represented by a single JSON object. Commerce sub document lines also use this format to store the BOM attributes in the line attribute (_line_bom_attributes).

The field name is the BOM attribute Variable Name. The following table defines the BOM attribute values

Field	Data Type	Required	Description
`value`	• String • Integer • Number • Boolean • Date • null	N	If the data type is String, the value is in the site base language. If the data type is Date, the value is stored in the JSON object as a as a String using the ISO date time format `yyyy-MM-ddTHH:mm:ssZ`.
`displayValue`	String	N	The display value of the attribute, in the site base language
`label`	String	N	Attribute (display) name, in the site base language.
`translations`	Object	N	A JSON object represents the translation of attributes in the site non-base language. The language code is the field name. `{ "de": {"label": xxx, "displayValue": xxx}, "fr":{"label": xxx, "displayValue":xxx}}` CPQ provides translation of values when the attribute is a menu type and the data type is a String. Other data types are not translated.

BOM INSTANCE JSON EXAMPLE

The following example displays the JSON object for a sample BOM instance.

```
{
   "id": "7345ABCDE",
   "variableName": "BM54888-0",
   "partNumber":  "BM54888",
   "quantity": 1,
   "definition": {"SequenceNum": 20, "ItemId": "EBS56321", "ItemType": "Optional Class"},
   "fields": {"_price_list_price_each": 99.12, "line_action_code":  "Update"},
   //Optional fields injected

    "attributes": {
        "size": {  //The attribute variable name
           "value": "L", //The attribute value in the site base language
           "displayValue": "Large", //The attribute display value in the site base language
           "label": "Size", //The attribute display name in the site base language
           "translations": {
              "de": { "label": "Größe", "displayValue":  "groß" },
              "en": { "label":  "Size",  "displayValue":  "Large" }
            }
        },

        "instruction": {
           "value": "Leave the package at the door.",
           "label": "Special Instruction",
           "translations": {
              "de": {"label": "Spezialanweisung"},
          // Only the label is translated, as the attribute is not a menu type.
              "en": { "label": "Special Instruction" }
            }
        }
    },

    "children": [
       {
         "variableName": "BM54888-1",
         "partNumber": "PT13345"
         "quantity": 1,
         "children": []
       },

       {
         "variableName": "BMDSK781-4",
         "partNumber": "DSK781-4",
         "quantity": 1
       }
    ]
}
```

IMPORTANT: // is used to add comments in the JSON object for explanation only. JSON does not formally support comments.

APPENDIX K: WEB SERVICES FOR BOM MAPPING

GET CONFIGURATION INSTANCE REST API

This operation will use one of the following identifiers to retrieve a saved Configuration BOM Instance:

- The lineId returned from a Terminate, Suspend, Resume, or Renew service
- The config_id returned by client side JSON object for client integration case.

IMPORTANT: Get Configuration Instance is only available for external integrations.

Get Configuration Instance			
URI Endpoint	`/rest/v6/configBomInstance/{config_Id}/actions/getConfigBom`		
Endpoint Parameters	`config_Id`	The Configuration BOM Instance Id	
HTTP Method	POST		
Request Parameters	`flattenHierachy`	True	Returns a flattened BOM structure True is the default value.
		False	Returns a hierarchical BOM structure
Response Parameters	JSON data containing the saved Configuration BOM instance		

URI Endpoint Sample

```
https://sitename.oracle.com/rest/v6/configBomInstance/36503159/actions/getConfigBom
```

Sample Request

```
{
   "flattenHierarchy": "true"
}
```

Sample Response

```
{
  "configBom": {
    "partNumber": "part1",
    "quantity": 3,
    "explodedQuantity": 3,
    "category": "sales",
    "id": "abo_3cf6636c-9119-4960-b185-d13daee2631d",
    "fields": {
      "_price_unit_price_each": "3.0",
      "oRCL_ABO_ActionCode_l": "NO_UPDATE",
      "fulfillmentStatus_l": "CREATED",
      "_is_line_item_mandatory": true,
      "itemInstanceName_l": "part1-36503087-1",
      "itemInstanceId_l": "abo_3cf6636c-9119-4960-b185-d13daee2631d"
    },
    "parentId": "",
    "children": [
      {
        "partNumber": "part22",
        "quantity": 1,
        "explodedQuantity": 6,
        "id": "abo_cdfd49b4-b500-4e11-84e7-3739d1f2625f",
        "fields": {
          "_price_unit_price_each": "US Dollar price not defined.",
          "oRCL_ABO_ActionCode_l": "NO_UPDATE",
          "_is_line_item_mandatory": true,
          "itemInstanceName_l": "part22-36503087-13",
          "itemInstanceId_l": "abo_cdfd49b4-b500-4e11-84e7-3739d1f2625f"
        },
        "parentId": "abo_d37e0168-c455-4748-af6c-da3fa2c0996c"
      },
      {
        "partNumber": "part222",
        "quantity": 1,
        "explodedQuantity": 6,
        "id": "abo_e3ddd440-6928-4231-88b2-3c6d3ed1c09f",
        "fields": {
          "_price_unit_price_each": "US Dollar price not defined.",
          "oRCL_ABO_ActionCode_l": "NO_UPDATE",
          "_is_line_item_mandatory": true,
          "itemInstanceName_l": "part222-36503087-12",
          "itemInstanceId_l": "abo_e3ddd440-6928-4231-88b2-3c6d3ed1c09f"
        },
        "parentId": "abo_d37e0168-c455-4748-af6c-da3fa2c0996c"
      }
     ]
  }
}
```

CONFIGURATION SOAP API SUPPORT FOR BOM MAPPING

The optional bomprice element is available for the Configuration SOAP API (v1 and v2). The response includes the BOM price for sites using BOM Mapping if the BOM price is not zero.

Sample Response:

```
<bm:price>
   <bm:bomPrice>$16.0000</bm:bomPrice>
   <bm:totalPrice>$45.0000</bm:totalPrice>
</bm:price>
The web service WSDL, upon regeneration, includes the newly introduced "bomPrice"
optional element:
<xsd:complexType name="ConfigurationPriceType">
   <xsd:sequence>
   ...
   <xsd:element maxOccurs="1" minOccurs="0" name="bomPrice" nillable="true"
type="xsd:string"/>
   <xsd:element maxOccurs="1" minOccurs="1" name="totalPrice" nillable="false"
type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```