TEAMCENTER

Engineering BOM Management — Deployment and Administration

Teamcenter 2412

SIEMENS

About Siemens Digital Industries Software

Siemens Digital Industries Software is a global leader in the growing field of product lifecycle management (PLM), manufacturing operations management (MOM), and electronic design automation (EDA) software, hardware, and services. Siemens works with more than 100,000 customers, leading the digitalization of their planning and manufacturing processes. At Siemens Digital Industries Software, we blur the boundaries between industry domains by integrating the virtual and physical, hardware and software, design and manufacturing worlds. With the rapid pace of innovation, digitalization is no longer tomorrow’s idea. We take what the future promises tomorrow and make it real for our customers today. Where today meets tomorrow. Our culture encourages creativity, welcomes fresh thinking and focuses on growth, so our people, our business, and our customers can achieve their full potential.

Support Center: **support.sw.siemens.com**

Send Feedback on Documentation: **support.sw.siemens.com/doc_feedback_form**

Contents

1. Plan the engineering BOM management deployment and administration

Managing the part and design data of a product within a single structure results in a complex BOM, which is difficult to maintain, resulting in rework, delays, and losses. It is more efficient to enable independent evolution of part and design lifecycles. Teamcenter Engineering BOM address this challenge by maintaining the parts and designs as separate structures. Aligning the two structures allows you to perform BOM-driven digital mockups.

As an administrator, you must set up Teamcenter for engineering BOM management.

Before you begin

Ensure that your Teamcenter setup has the following applications:

The application	Is used
Business Modeler IDE (BMIDE)	To configure Teamcenter for engineering BOM management.
Dispatcher	To run various user tasks in the background.
Change Manager	To perform engineering BOM management tasks in the context of a change.
Visualization Server (Windows, UNIX)	To visualize products and to perform BOM-driven digital mockups.

Apart from these Teamcenter applications, you may need a CAD application, such as NX or CATIA, to manage the designs and might need to integrate the application with Teamcenter.

Choose additional Teamcenter applications for interoperability

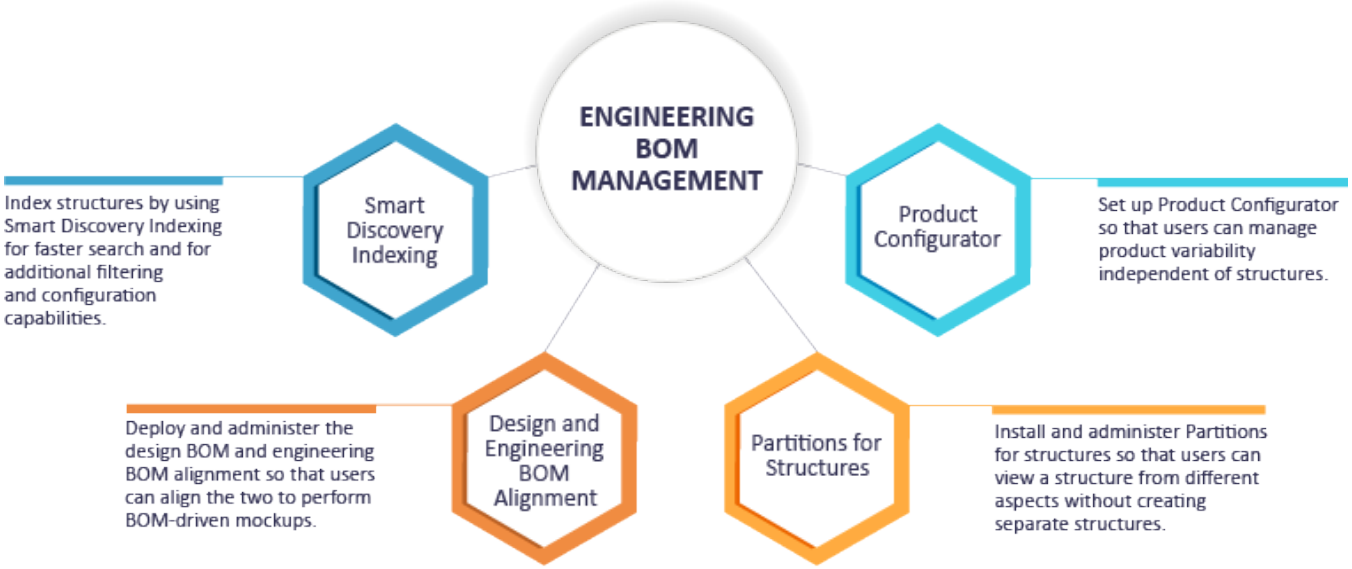
Based on your business requirements, you can install the following applications in your Teamcenter setup:

The application	Is used
Classification	To classify parts so that users can reuse similar, existing parts by performing a classification-based search.
Integrated Material Management (IMM)	To maintain the approved material and substance information of parts. See the IMM documentation available at **Support Center** > Teamcenter > Downloads

The application	Is used
	> Additional Downloads > Integrated Material Management Solution.
Vendor Management	To maintain the vendor information, such as vendor name, contact number, and location, for vendor parts.
Requirements Manager	To capture the customer needs related to products and part enhancements as requirements.
Embedded Software Management	To include software artifacts as engineering parts in a product or a part structure.
Electronic Design Automation (Siemens EDA, non-Siemens EDA)	To include ECAD parts in a product or a part structure.

Set up engineering BOM management

There are several Teamcenter products that provide different functionalities to aid engineering BOM management. After setting up Teamcenter for engineering BOM management, deploy and administer these products so that the business users can perform tasks related to engineering BOM management in Teamcenter.



Where do I go from here?

Business User	• For managing engineering BOM data, see *Engineering BOM Management — Usage*.

	• For organizing BOM content within partitions, see *Structure Partitions — Usage*. • For aligning design BOM and engineering BOM, see *Design and BOM Alignment — Usage*.
Administrator	
New user	Familiarize yourself with the engineering BOM management **terms**.
Set up Teamcenter for engineering BOM management	• **Install** Teamcenter applications required for engineering BOM management. • **Qualify** the existing Teamcenter business objects as engineering BOM data. • Set up the **structure management mode** for parts. • **Extend** the default Teamcenter data model for managing the engineering BOM data. • Perform the required **administration tasks**.
Index structures	Index structures by using Smart Discovery Indexing for faster search and for additional filtering and configuration capabilities.
Set up Partitions for structures	Deploy and administer Partitions for Structures so that users can view a structure from different aspects without creating separate structures.
Set up design BOM and engineering BOM alignment	Deploy and administer design BOM and engineering BOM alignment so that users can perform BOM-driven mockups in a design-part separated environment.
Deploy and administer Product Configurator	Set up Product Configurator so that users can manage variability data outside of structures.
Control Teamcenter's behavior and appearance for structures	Use preferences to control the various aspects of Teamcenter to suit the requirements at your site. For information about retrieving a list of preferences, see *Where can I get a list of preferences?* in *Teamcenter Preferences*.

2. Engineering BOM management terms

<table>
<tr><th>The term</th><th>Refers to</th></tr>
<tr><td>Design</td><td>The virtual representation of a part. It represents what a part will look like in its floor position. For example,  represents a wheel. Designs are attached to design revisions in Teamcenter.</td></tr>
<tr><td>Part</td><td>This is the primary component of a product. For example, the product CrossKart consists of different parts such as Wheel, Engine, and Chassis. Each part is saved as a part revision in Teamcenter.

A part can be of the following types:

<table>
<tr><th>The type</th><th>Refers to</th></tr>
<tr><td>Component</td><td>A single piece part used in a product. You can add child parts to a component but they neither participate in alignment process nor corresponding manufacturing BOM creation. For example, you can add a spare part as a child component.</td></tr>
<tr><td>Generic part</td><td>A placeholder part that cannot be ordered or procured, or assembled in a product. It has variability associated with it. It is used in a configurable assembly and is replaced with a specific part when a 100% product variant is derived. You can add child parts to a generic part but they neither participate in alignment process nor corresponding manufacturing BOM creation.</td></tr>
<tr><td>Fixed assembly</td><td>An assembly part that has a fixed set of child parts. A fixed part cannot have a variability associated with it.

A fixed assembly can contain other fixed assemblies and components.</td></tr>
<tr><td>Configurable assembly</td><td>An assembly part that has variability associated with its child parts.

A configurable assembly can contain other configurable assemblies, fixed assemblies, generic parts, and components.</td></tr>
</table>
</td></tr>
<tr><td>Alternate part</td><td>A part that can be used in place of another part in all products.</td></tr>
<tr><td>Substitute part</td><td>A part that can be used in place of an occurrence of another part within a single assembly.</td></tr>
<tr><td>Flexible part</td><td>A part that can be used in a product in various states. Examples include a coil spring, hose pipe, and wire harness.</td></tr>
<tr><td>Bought out part</td><td>A part that is purchased and not manufactured in-house.</td></tr>
</table>

The term	Refers to
Part structure (engineering BOM)	An assembly part that has child parts.
Part occurrence	An instance of a part in a part structure. For example, the wheel assembly contains the part occurrences of *Tire*, *Rim*, and *Valve*.
Partition	A container to logically organize the engineering BOM data.
Design	A virtual representation of a part.
Design structure	A design structure of a product to represent it virtually from a geometrical standpoint. It consists of designs, which contain the design information such as CAD and product manufacturing information (PMI).
Design occurrence	An occurrence of a design in a design structure. A design occurrence contains the position information of a part. For example, the design occurrence of a wheel contains the information related to its placement in *Crosskart*.
Design-part alignment	The alignment between a design and a part. It shows what a part will look like in its floor position.
Design occurrence-part occurrence alignment	The alignment between a design occurrence and a part occurrence. It shows what a part will look like when placed in a product.
Solution variant	A 100% buildable BOM with unique part numbers.

3. Install Teamcenter engineering BOM using Deployment Center

Add the engineering BOM application to your existing Teamcenter environment.

Prerequisites

- Teamcenter is already setup. If not, follow the instructions in *Teamcenter Installation Using Deployment Center* to deploy Teamcenter.
- If Teamcenter is already set up, verify if **Visualization Server** is installed. If it is not installed, first ensure that the hardware and software requirements for **Visualization Server Manager** are met. For this, refer to *Visualization Server Manager prerequisites* section in the *Teamcenter Installation Using Deployment Center* documentation help. Next, Next, read the *Visualization Server overview* to setup **Visualization Server**.
- Deployment Center is setup. If not, plan its deployment.

Procedure

1. Log on to Deployment Center and select the environment to which you want to add the engineering BOM.
2. Go to the **Applications** task. Click **Add or Remove Selected Applications**.
3. In the **Available Applications** panel, use the web browser search to find the following applications and select them.
   - **Active Content**
   - **Assembly BOM**
   - **Design BOM Alignment Automation**
   - **Assembly BOM Rapid Start**

     This application installs the following items:

     - Sample data that users can use to quickly get started with engineering BOM, along with a readme file providing instructions on how to import this sample data into your environment.
     - A file containing a consolidated list of preferences for engineering BOM.

- The `BMF_Pma0_find_target_items_for_sources` and `BMF_Pma0_populate_target_item_creI_for_sources` files, which help you customize Teamcenter behavior with user exits.

Click **Update Selected Applications**.

Deployment Center automatically selects any additional dependent applications.

4. Go to the **Components** task.

5. In the **Selected Components** list, note any remaining components whose configuration status is not **100%**. Select each incomplete component, enter required parameters, and save component settings until all components in the environment show a configuration status of **100%**.

   When all components are fully configured, the **Deploy** task is enabled.

6. Go to the **Deploy** task. Click **Generate Install Scripts** to generate deployment scripts you will use to update affected machines.

   When script generation is complete, note any special instructions in the **Deploy Instructions** panel.

7. Locate deployment scripts, copy and run each script to its target machine.

   For more information about running deployment scripts, see *Deployment Center — Usage*.

Postrequisites

To organize the EBOM content in partitions, deploy Partitions for Structure.

To index structures for faster search and for additional filtering and configuration capabilities, deploy Smart Discovery Indexing.

To manage variability data outside of structures, deploy Product Configurator.

4. Install Teamcenter engineering BOM using TEM

Before installing ensure that **Visualization Server** is installed in your Teamcenter setup.

Next, update your Teamcenter setup to install the following features through Teamcenter Environment Manager (TEM):

- **Base Install** > **Active Workspace** > **Client** > **Design BOM Alignment Automation**
- **Base Install** > **Active Workspace** > **Client** > **Assembly BOM**
- **Base Install** > **Active Workspace** > **Client** > **Active Content**
- **Base Install** > **Active Workspace** > **Client** > **Solution Variant Support for Active Content Structure**
- **Base Install** > **Active Workspace** > **Server Extensions** > **Active Content Structure**
- **Base Install** > **Extensions** > **Assembly BOM Rapid Start**

  This feature installs the following items:

  - Sample data that users can use to quickly get started with engineering BOM, along with a readme file providing instructions on how to import this sample data into your environment.
  - A file containing a consolidated list of preferences for engineering BOM.
  - The `BMF_Pma0_find_target_items_for_sources` and `BMF_Pma0_populate_target_item_creI_for_sources` files, which help you customize Teamcenter behavior with user exits.
- **Base Install** > **Extensions** > **Assembly BOM**
- **Base Install** > **Extensions** > **Design BOM Alignment Automation**
- **Base Install** > **Extensions** > **BOM Management** > **Design BOM Alignment Automation Foundation**

To organize the EBOM content in partitions, deploy Partitions for Structure.

To index structures for faster search and for additional filtering and configuration capabilities, deploy Smart Discovery Indexing.

To manage variability data outside of structures, deploy Product Configurator.

5. Set BMIDE templates for engineering BOM management

You perform the BMIDE configurations required for engineering BOM management within a BMIDE template project.

Procedure

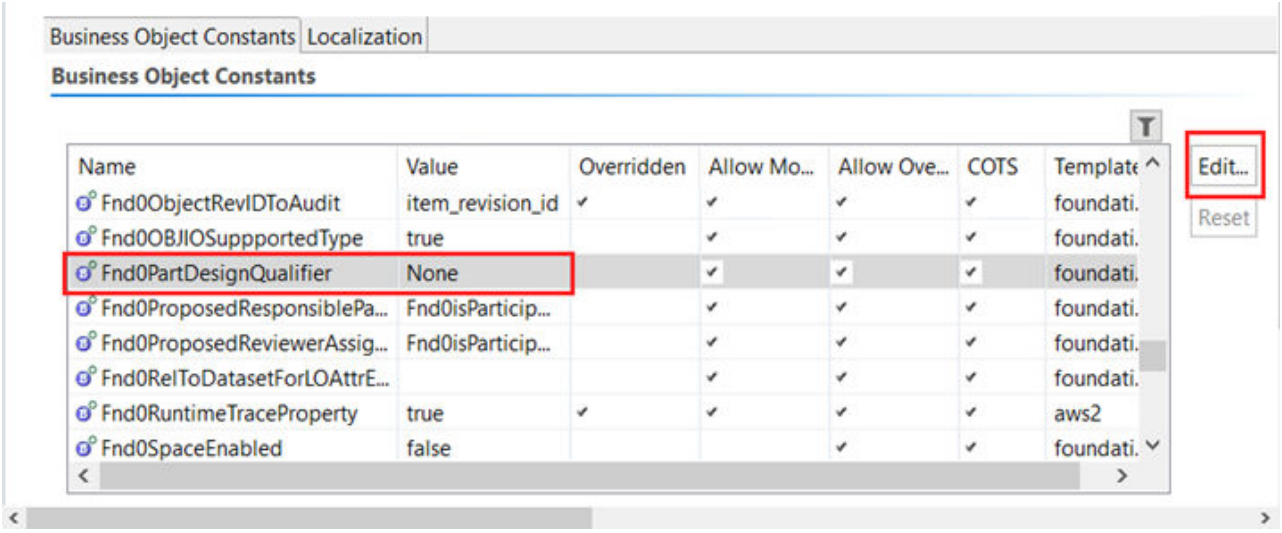
1. Create a new BMIDE template project.
2. Add the following as the dependent templates:
   - **aws2**
   - **activeworkspacebom**
   - **pma0automation**
   - **pma1awautomation**
   - **smc0psmcfgsupport**
   - **smc1activeworkspacebom**

6. Qualify business objects as engineering BOM data

You qualify business objects of type **Item Revision** and its child types as engineering parts and designs so that the business objects can be used to create engineering BOM content.

Procedure

1. In BMIDE, load the BMIDE template project that you want to work with.
2. Expand the template project.
3. Under Extensions, add the new extension (schema) file that contains the business objects.
4. Set the extension file containing the business objects as the active extension file.
5. Locate the required business object, such as **Item Revision**, **Part Revision**, or **Design Revision**.
6. In the **Business Object Constants** tab, locate **Fnd0PartDesignQualifier**, and click **Edit**.



In the **Modify Business Object Constant** dialog box, select one of the following values:

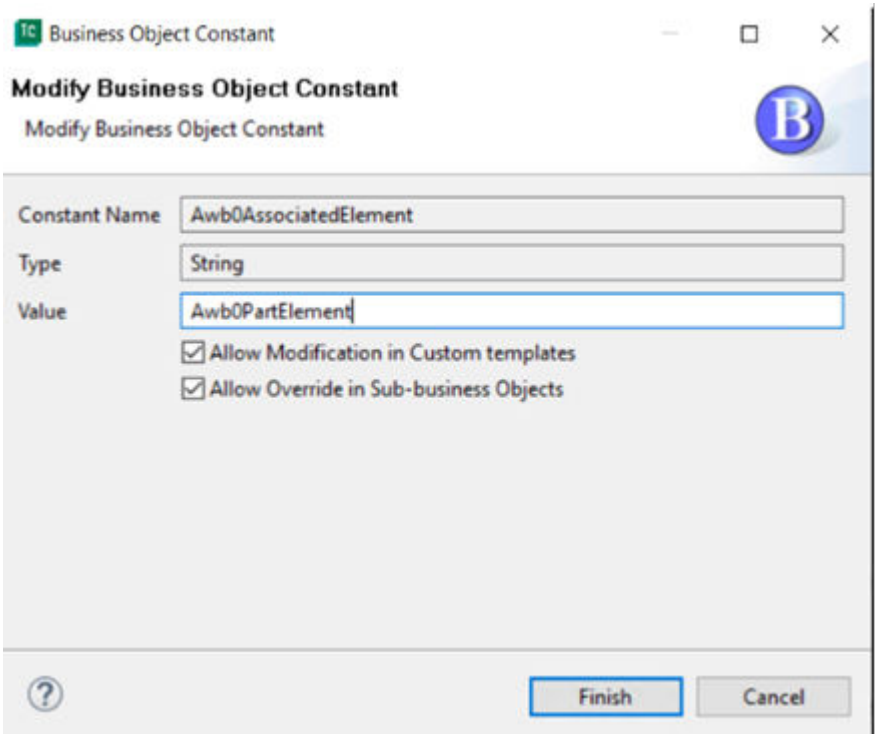
- **Part** to qualify the business object as an engineering part.

  If the selected business object is **CommercialPart Revision**, its **Fnd0PartDesignQualifier** is already set as **MultiDomainPartOrDesign**. This is because commercial parts are qualified as both part and design (dual intent), by default. If you want commercial parts to be included

in an engineering BOM in a design-part separated system, you must change the value of **Fnd0PartDesignQualifier** to **Part**.

- **Design** to qualify the business object as a design.
- **PartAndDesign** to qualify the business object as part as well as design (dual intent).
- None

If you set **Fnd0PartDesignQualifier** as **Part** , then change the value of the **Awb0AssociatedElement** business object constant to **Awb0PartElement**.



> Note:
>
> Make sure these business objects are updated the in value of the **TCAllowedChildTypes_***parent_item_type* preference.

7. Click **Finish**.

Postrequisites

If you deploy the above-mentioned changes through BMIDE, you must run the following utility:

bmide_modeltool -u=*admin_user_name* **-p=***admin_user_password* **-g=dba -tool=tc_entcba_ebom_types_pref_updater -mode=install -target_dir=***TC_DATA*

7. Set the structure management mode for parts

After **qualifying** business objects as engineering parts, you must set the structure management mode.

1. In BMIDE, locate the business objects that you qualified as engineering parts.

2. In the **Business Object Constants** tab, locate **Fnd0StructureManagementMode**.

   Set the value of **Fnd0StructureManagementMode** to **0** if you want to manage the part as an assembly part (represented as a part structure). Here, the assembly part is considered as the unit of change so that changes are tracked per assembly.

If you deploy the above-mentioned changes through BMIDE, you must run the following utility:

bmide_modeltool -u=*admin_user_name* **-p=***admin_user_password* **g=dba**
-tool=tc_entcba_ebom_types_pref_updater -mode=install -target_dir=*TC_DATA*

8. Qualify ECAD parts to be used in an engineering BOM

ECAD parts are used as both design and part in the BOM. Hence, they are shared between the design and part structures. The values for the preference **FND0_MULTIDOMAINPARTORDESIGN_TYPES** list all the types marked as both part and design (referred to as dual intent) by Teamcenter.

You must mark different types as parts with dual intent to display this behavior.

9. Plan the transition from single-overloaded BOM management to design-part-separated BOM management

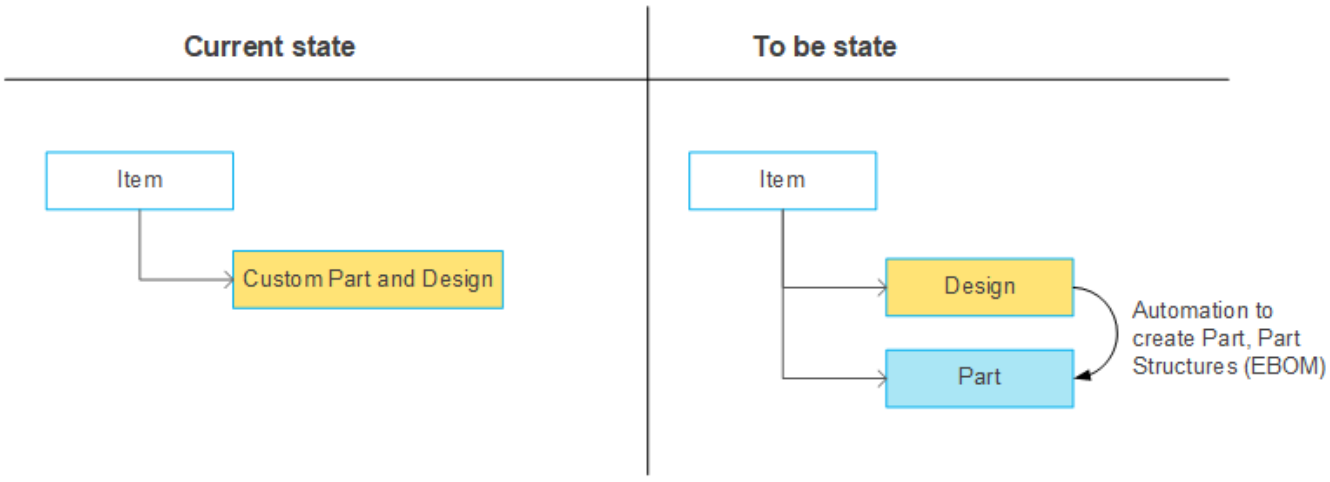
If you currently manage your parts and designs within a single-overloaded structure and want to transition to Teamcenter Engineering BOM for managing designs and parts within separate structures, you must make specific changes to your current data model.

For example, in your current data model, you may be currently managing custom parts and designs in one of the following ways:

- **Scenario 1:** The parts and designs are managed within the **Custom Part and Design** business object under the **Item** business object.
- **Scenario 2:** The parts and designs are managed within the **Custom Part and Design** business object under the default **Part** business object.
- **Scenario 3:** The parts and designs are managed within the **Custom Part and Design** business object under the default **Design** business object.

Scenario 1

Irrespective of whether the design files are available for **Custom Part and Design**, you must qualify this business object to act like a design.



1. Qualify the existing **Custom Part and Design** business object as a design by setting **Fnd0PartDesignQualifier** to **Design**.
2. Add a new persistent property, **Is Part Required**, to the **Design** business object.

In this scenario, because the **Custom Part and Design** business object is qualified as a design, add the persistent property to this business object. Next, add the persistent property to the **Pma0PartReqPropNameOnDesign** global constant.

3. Include the design, that is, the **Custom Part and Design** business object in the **Awb0AssociatedElement** business object.

4. Create a new business object, **Part**, under the default **Part** business object.

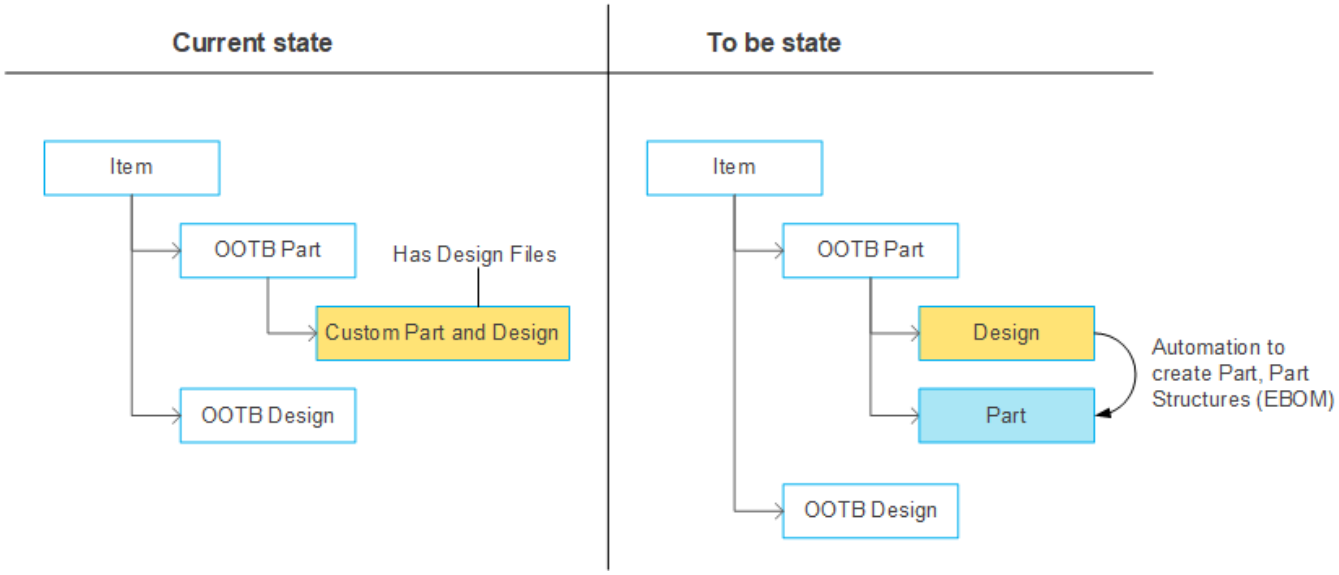
5. Configure the part to be automatically created from the custom design.

6. Map the custom part and design for alignment.

7. **Run the validate_ebom_configurations** utility to verify the configurations are correctly set up for the custom designs and parts.

Scenario 2

If design files are available for **Custom Part and Design**, you must qualify this business object to act like a design.



1. **Qualify** the existing **Custom Part and Design** business object as a design by setting **Fnd0PartDesignQualifier** to **Design**.

2. Add a new persistent property, **Is Part Required**, to the **Design** business object.

   In this scenario, because the **Custom Part and Design** business object is qualified as a design,

   a. Add the persistent property to this business object.

b. Add the persistent property to the **Pma0PartReqPropNameOnDesign** global constant.

3. Include the design, that is, the **Custom Part and Design** business object, in the **Awb0AssociatedElement** business object.

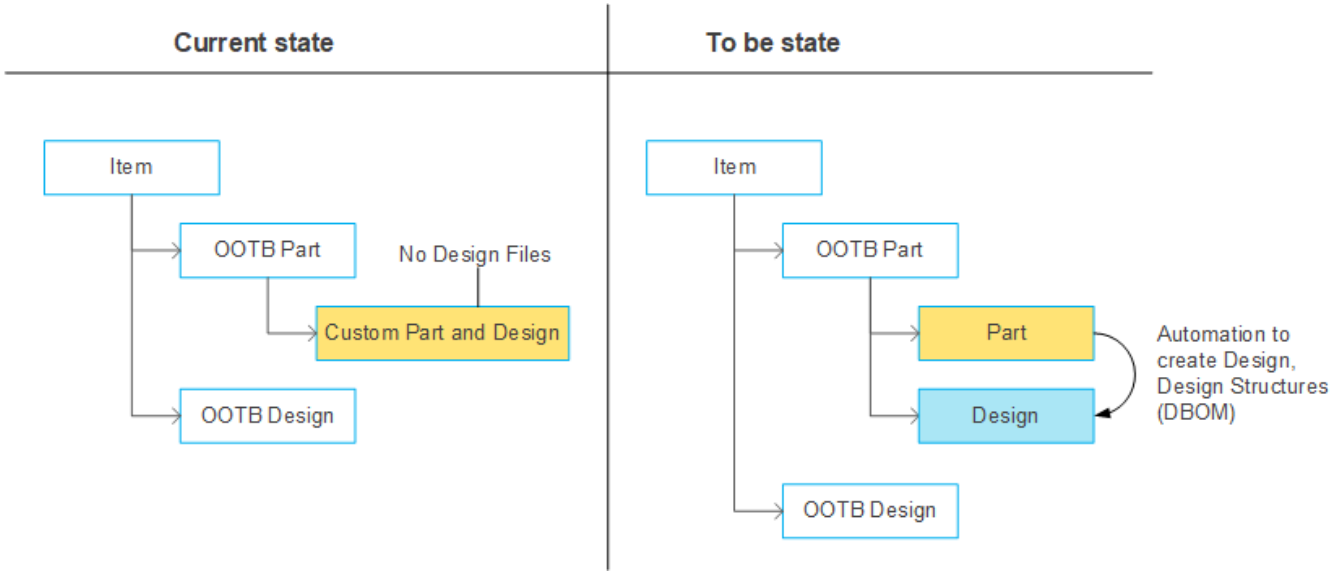
4. Create a new business object, **Part**, under the default (OOTB) **Part** business object.

5. Configure the part to be automatically created from the custom design.

6. Map the custom part and design for alignment.

7. **Run the validate_ebom_configurations** utility to verify that the configurations are correctly set up for the custom designs and parts.

If design files are not available for **Custom Part and Design**, you must qualify this business object to act like a part.



1. **Qualify** the existing **Custom Part and Design** business object as a part by setting **Fnd0PartDesignQualifier** to **Part**.

2. Add a new persistent property, **Is Design Required**, to the **Part** business object.

   In this scenario, because the **Custom Part and Design** business object is qualified as a part,

   a. Add the persistent property to this business object.

   b. Add the persistent property to the **Pma0PartReqPropNameOnDesign** global constant.

3. Include the part, that is, the **Custom Part and Design** business object in the **Awb0AssociatedElement** business object.

4. Create a new business object, **Design**, under the default **Part** business object.

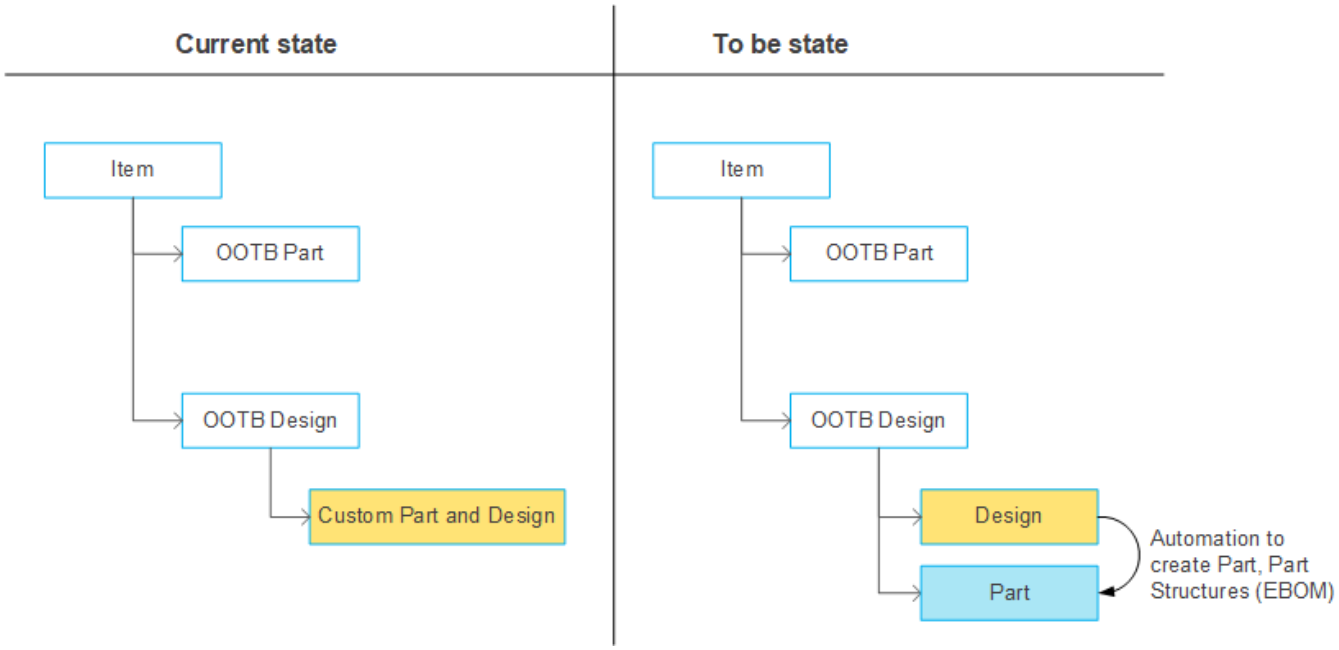
5. Configure the design to be automatically created from the custom part.

6. Map the custom part and design for alignment.

7. **Run the validate_ebom_configurations** utility to verify that the configurations are correctly set up for the custom designs and parts.

Scenario 3

Irrespective of whether design files are available for **Custom Part and Design**, you must qualify this business object to act like a design.



1. **Qualify** the existing **Custom Part and Design** business object as a design by setting **Fnd0PartDesignQualifier** to **Design**.

2. Add a new persistent property, **Is Part Required**, to the **Design** business object.

   In this scenario, because the **Custom Part and Design** business object is qualified as a design,

   a. Add the persistent property to this business object.

   b. Add the persistent property to the **Pma0PartReqPropNameOnDesign** global constant.

3. Include the design, that is, the **Custom Part and Design** business object in the **Awb0AssociatedElement** business object.

4. Create a new business object, **Part**, under the default **Design** business object.

5. Configure the part to be automatically created from the custom design.

6. Map the custom part and design for alignment.

7. **Run the validate_ebom_configurations** utility to verify that the configurations are correctly set up for the custom designs and parts.

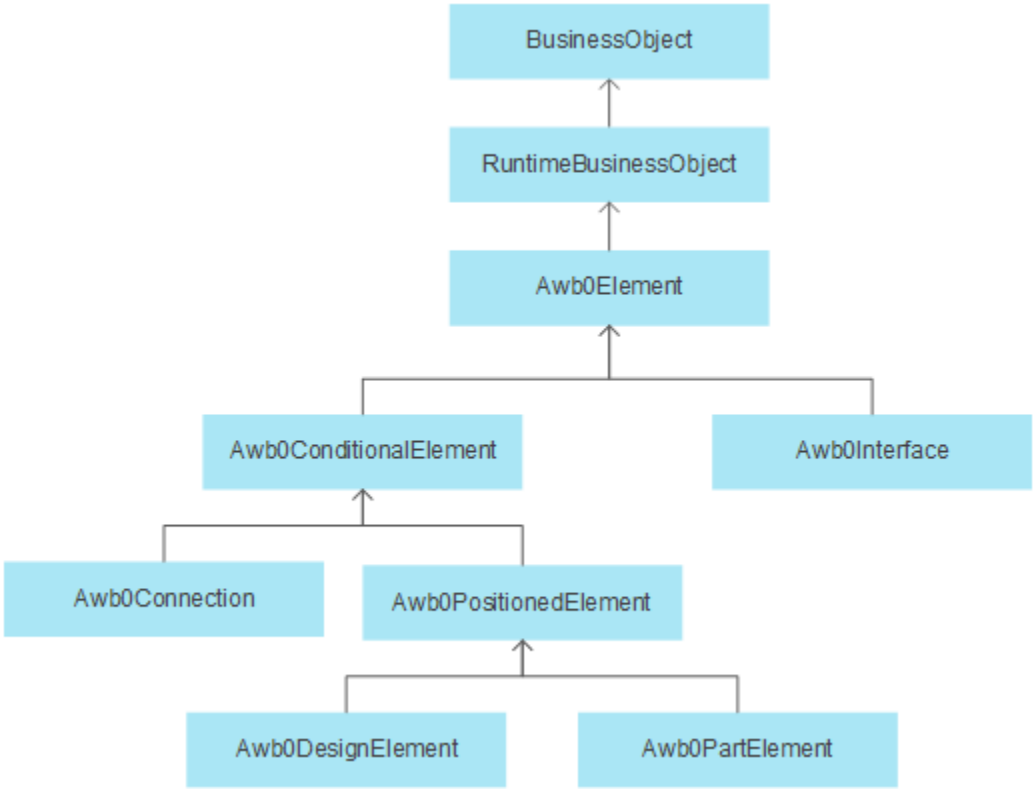
10. Extend the default Teamcenter data model for structures

Business objects required to extend Teamcenter for structures on Active Workspace

Active content provides a number of business objects to represent the occurrences returned by the various structure applications. **Awb0Element** is the root type and is abstract. The following table lists the default active content types that represent occurrence types for BVR structures.

Business object	Purpose
Awb0ConditionalElement	Represents an occurrence with effectivity and variant conditions. Generic design element (GDE) lines do not have effectivity or variant conditions; they should not be mapped to this type or its subtypes.
Awb0Connection	Represents an occurrence that does not have any associated geometry information.
Awb0DesignElement	Represents an occurrence with geometry. It provides the properties for managing geometry, for example, bounding box and transform. Objects of this type can be visualized in the viewers.
Awb0Element	Represents the root business object for all occurrence management objects. An element is identified by a name and can belong in a structure with a parent-child relationship. This is an abstract object and should not be mapped with any item revision type.
Awb0Interface	Represents the generic design element.
Awb0PartElement	Represents a part type and is mapped with the part revision. All the part-specific properties and behavior are associated to the part element.
Awb0PositionedElement	Represents an item revision type with geometry.

Hierarchy of business objects in the active content data model is shown in the figure:



Make BOMLine properties available on the Awb0Element for consistent data representation and accessibility

Making **BOMLine** properties available to **Awb0Element** ensures consistent data representation and accessibility. It also makes **BOMLine** data available in the context of **Awb0Element** and displays it correctly. You can map both out of the box and custom properties. However, runtime configuration properties are automatically mapped.

For parts and designs, it is recommended to map properties to **Awb0ConditionalElement** instead of **Awb0DesignElement** or **Awb0PartElement**. This is because **Awb0ConditionalElement** is the parent runtime business object of both **Awb0DesignElement** and **Awb0PartElement**.

Prerequisites

- Check if the property is already mapped on **Awb0ConditionalElement** or one of its child types. For example, if a property is already mapped on **Awb0DesignElement**, do not map the same property on **Awb0ConditionalElement** to avoid an inconsistent data model state.

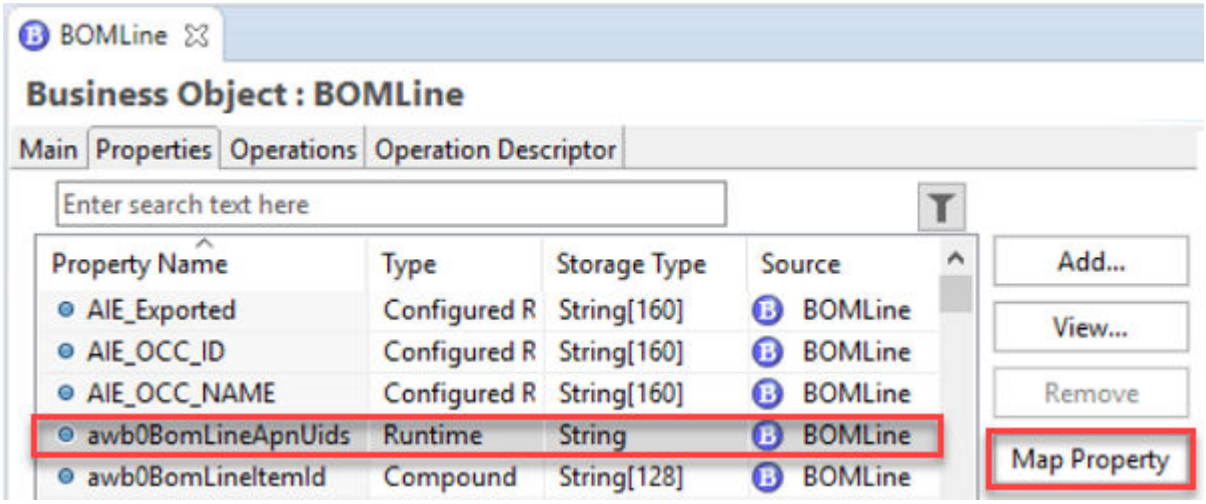
- You can create additional model elements as needed and attach them to the newly created property after mapping. For example, **LOV attaches**, **Property Formatter**, **Property Renderer**, and **Localization**.

Procedure

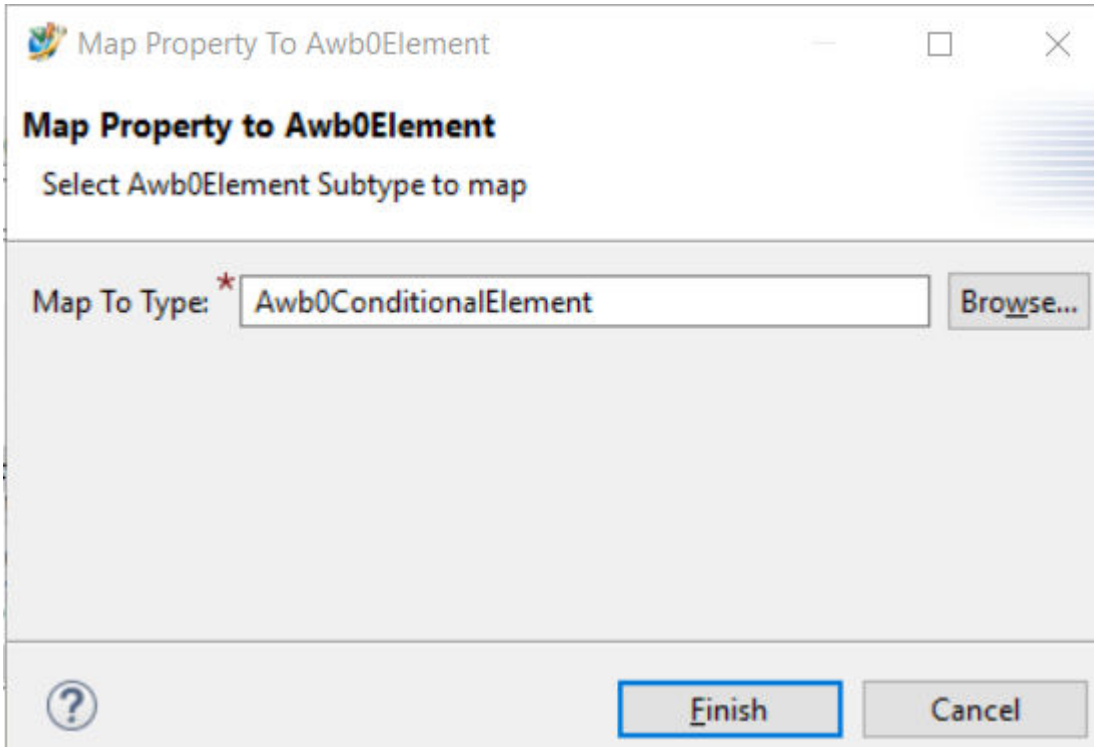
1. In the custom BMIDE template project, search and open the **BOMLine** business object.

2. In the **Properties** tab, search and select the property to map. For example, **awb0BomLineApnUids**.

3. Click the **Map Property** button.



4. In the **Map To Type** box, browse and select the **Awb0ConditionalElement** type.



5. Click **Finish**.

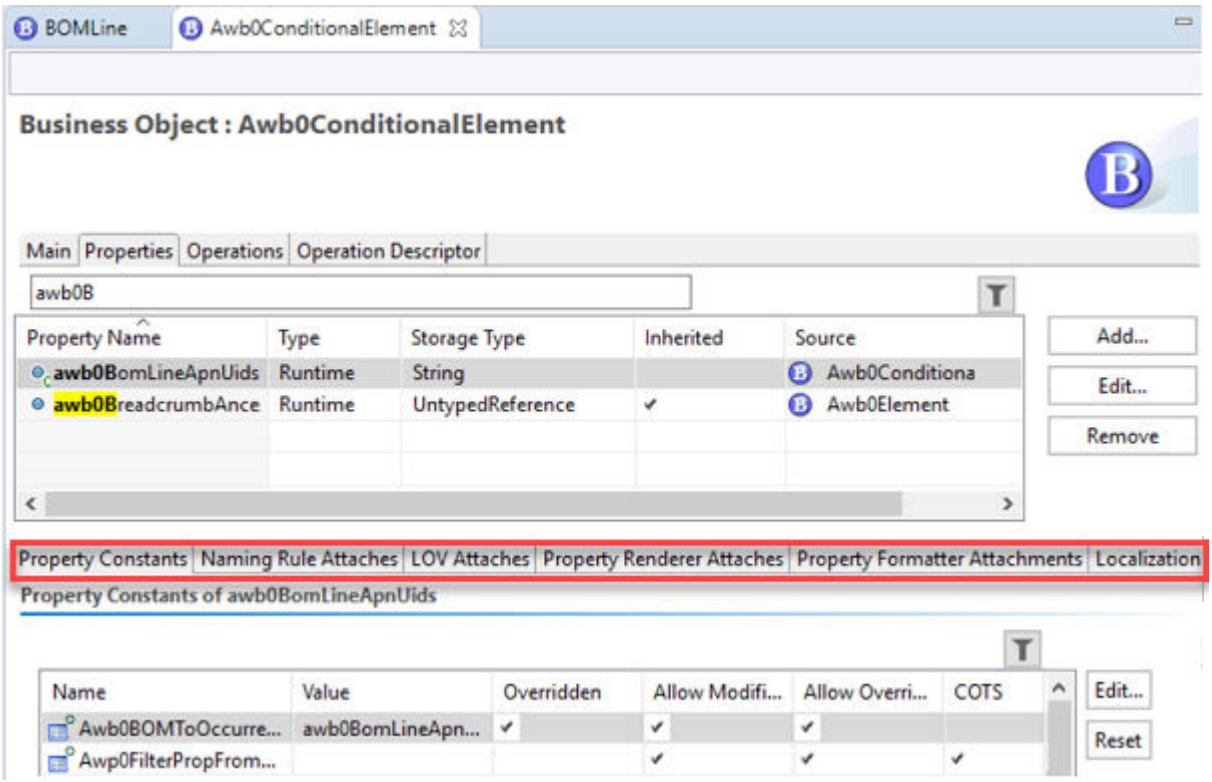
   Now, the property is also available on the **Awb0ConditionalElement** with the same name and type as that of the **BOMLine** property.

   The **Awb0BOMToOccurrence** property constant value of the property **awb0BomLineApnUids** on the **Awb0ConditionalElement** is set to the name of the existing BOMLine property.

   The **LOV attaches**, **Property Renderer**, **Property Formatter**, and **Localization** data copied from the existing BOMLine property to the new property.

   Or

   You can add the custom model data that you created before mapping.



   If you want to remove the newly created mapping, search and open the **Awb0ConditionalElement** business object, select the property (**awb0BomLineApnUids**) and click on the **Remove** button.

6. Save and deploy the template.

Mapping properties to occurrence properties

Domain-specific occurrences contain properties relevant to the specific domain. End users understand and interact with these domain-specific properties. These properties are mapped from the **BOMLine**

or **ModelElement** type onto the occurrence. This mapping is provided by property constants defined in the Business Modeler IDE and the property constants are scoped to the **Awb0Element** type. Default properties are provided on **Awb0Element** and its subtypes, but you can add custom properties necessary for your implementation.

All custom properties must be mapped to a property defined on the type specified in the **Awb0BOMToOccurrence** type constant. The property mapping is then achieved through the **Awb0BOMToOccurrence** property constant. The value of this property constant is inherited and can be overridden at any level.

For example, the **awb0BoundingBox** property on the **Awb0PositionedElement** business object has the value of **bl_bounding_boxes**. It also has the value **BOMLine** for the **Awb0BOMToOccurrence** type constant. Consequently, whenever the **awb0BoundingBox** property is requested on an **Awb0PositionedElement** object, the value is fetched from the **bl_bounding_boxes** property of the BOM line.

The mapping of some common default properties are listed below:

Business object	Property	Awb0BOMToOccurrence value
Awb0Element	**awb0Parent**	**bl_parent**
Awb0Element	**awb0Name**	**bl_line_name**
Awb0Element	**awb0ElementId**	
Awb0Element	**awb0Archetype**	**bl_revision**
Awb0ConditionalElement	**awb0ArchetypeEffFormula**	**bl_revision_effectivity**
Awb0ConditionalElement	**awb0ElementId**	**awb0BomLineItemId**
Awb0PositionedElement	**awb0BoundingBox**	**bl_bounding_boxes**
Awb0PositionedElement	**awb0Transform**	**bl_plmxml_abs_xform**

Default getter and setter methods are registered for all properties of **Awb0Element** and child business objects that have a mapping defined in the **Awb0BOMToOccurrence** property constant. You must provide a custom getter and setter for all properties that are not already mapped. For example, the **awb0NumberOfChildren** property specifies the number of child elements and does not have a value for the property constant; custom getter and setter methods are registered for it. You can use the same mechanism to register getter and setter methods for properties on custom **Awb0Element** subtypes.

Many configured runtime properties on **BOMLine** are derived from **Item**, **ItemRevision**, or another object. These are always of type **string**, irrespective of the type from which they are derived. Mapping to such a property will result in losing the type information about the property that the system requires for proper filtering of Solr results. To avoid this, instead of mapping to a configured runtime property, define a new compound property on the backing object and use the relations or reference properties to get the source property. This maintains the type information for the property so that it can be used in mapping.

For example, you may want to get the last modified date stored in the configured runtime property **bl_rev_last_mod_date** on the BOM line. Instead of using the bl_rev_last_mod_date property, consider the **awb0RevisionLastModifiedDate** compound property on the BOM line. This uses the **bl_revision** property to access the item revision and you can get the **last_mod_date** property from there. An **awb0ArchetypeRevLastModDate** property is defined on the **Awb0DesignElement** business object and then mapped to the **awb0RevisionLastModifiedDate** property.

Enable the display of custom objects set to manage structures

Add custom objects to the Content tab and search

To display your custom business object in the **Content** tab of Active Workspace, you must add it to the **Awb0SupportsStructure** global constant in the Business Modeler IDE. This procedure also allows instances of the custom business object to be returned by in context searches if the object is indexed, and also adds the **Add Element** button for the custom object.

1. In the Business Modeler IDE, ensure the **Active Content Structure** template is loaded and then open the Global Constants Editor.

2. Select the **Awb0SupportsStructure** constant and then click **Edit**.

3. In the **Add** dialog box of the **Edit** window, enter the name of the class corresponding to the first custom object you want to display.

4. Repeat the previous step for each of the other custom objects you want to display.

5. Click **Finish** on both, the **Modify Global Constant** and the **Add Value** dialog boxes to save the additions.

   > Note:
   >
   > On some systems, you may have to reopen each additional object in the **Edit** window and save the entry again for each added item. Failure to save your entries on the first attempt does not necessarily indicate they are incorrect.

6. To enable the custom objects to be returned by in context searches, set the **Awp0SearchIsIndexed** business object constant to **true** on each custom object.

Display the Content tab with custom business object types

If you have custom business object types that have assembly children, the **Content** tab will not be displayed by default for them. To get the **Content** tab to appear in this case, you must perform the following steps:

1. **Ensure that the corresponding custom business object type is listed** under the **Awb0SupportsStructure** global business constant.

2. Add the following page pertaining to the **Content** tab to the **ShowObjectLocation Summary** XRT.

```
<inject type="dataset" src="Awb0ContentTab"/>
```

3. If the **ShowObjectLocation Summary** XRT is not available, add a custom summary XRT for the required business object. Then, add the preference for **Custom_BO_type.showObjectLocation.<***SubLocation***>**. This maps the summary XRT to the business object type.

Add custom properties to default parts and custom parts

About adding custom properties to default parts and custom parts

By default, certain business objects are qualified as engineering parts. You can identify additional business objects and **qualify them as engineering parts** to create custom parts. The default engineering parts contain certain properties, such as **Assembly Indicator**, **Safety Part**, **Finish Type**, and **Part Maturity**, and these are inherited by the custom parts. You can include additional (custom) properties on the default and custom parts. If solution variants are created for these parts, the custom properties are automatically copied over to the solution variants.

To add custom properties:

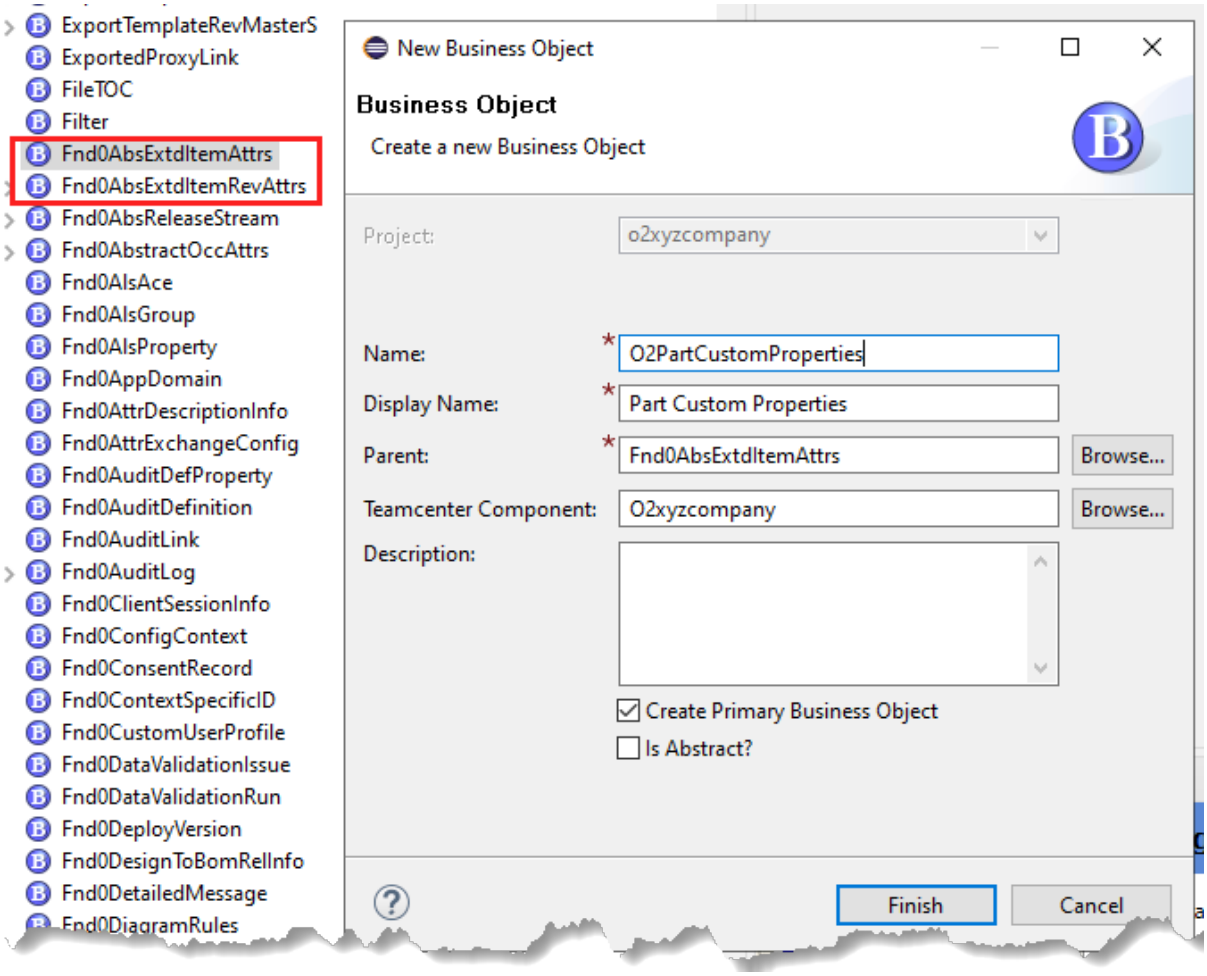
1. Perform the following steps to add custom properties to a part or part revision:

   a. If you want to add custom properties to a part, create a business object under **Fnd0AbsExtdItemAttrs** business object in BMIDE.

      If you want to add custom properties to a part revision, create a business object under **Fnd0AbsExtdItemRevAttrs** business object in BMIDE.

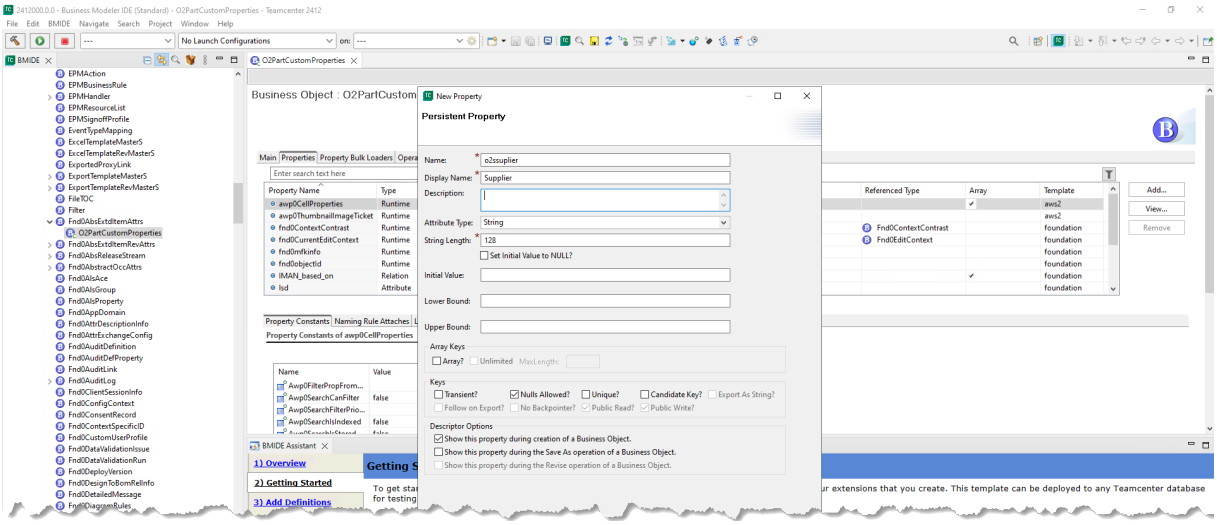
      > Example:
      >
      > Suppose that you want to add custom properties to **Part** item. In such a case, create another business object as an extended attribute object **PartCustomProperties** under **Fnd0AbsExtdItemAttrs** business object.

      The **Fnd0ExtdDesignRevAttrs** business object is an attribute object by default available for a design revision.



b. Add custom properties as persistent properties to the extended attribute object.

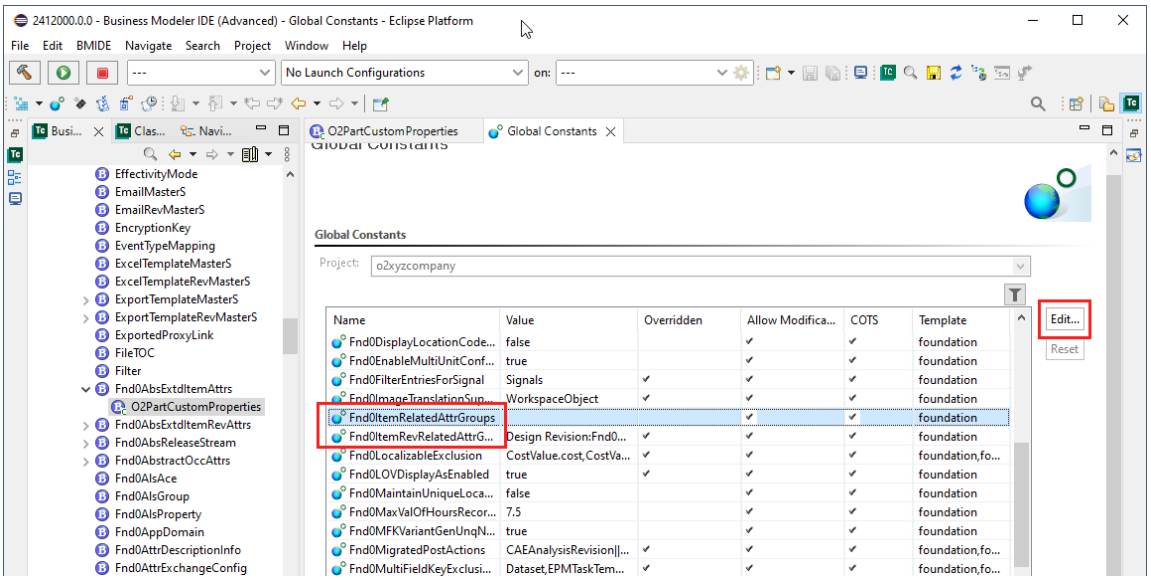
For example, add **Supplier** as a persistent custom property to **PartCustomProperties** object.



c. Use the following steps to assign the newly created extended attribute object to the target business object.

A. If you are adding custom properties to a part, select the **Fnd0ItemRelatedAttrGroups** global constant in BMIDE, and click **Edit**.

If you are adding custom properties to a part revision, select the **Fnd0ItemRevRelatedAttrGroups** global constant in BMIDE, and click **Edit**.



B. On the **Modify Global Constant** dialog box, click **Add**.

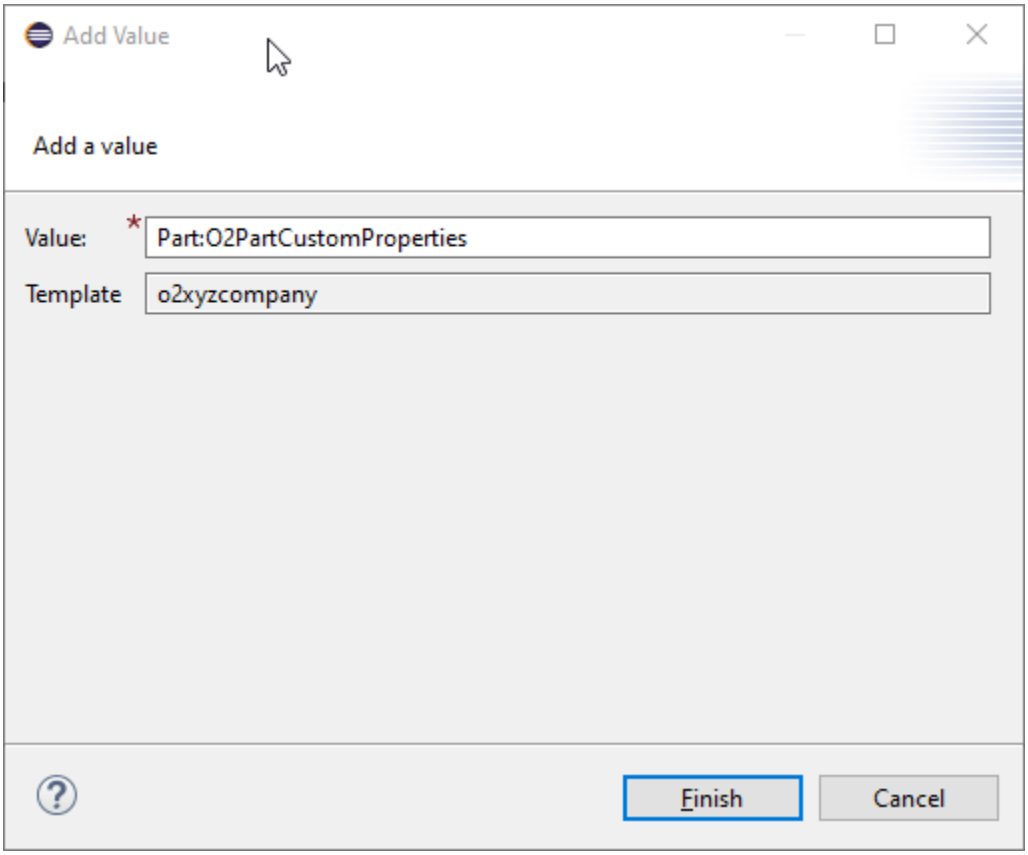
C. In the **Value** box, enter the name of the target business object and the newly created extended attribute object in the following format: **<Target business object>:<Newly created extended attribute object>**.

For example, enter **Part:PartCustomProperties**.

If you want to assign multiple extended attribute objects simultaneously, separate them with a , (comma). You cannot edit an existing value. Instead, add a new value.

D. Click **Finish** to close the **Add Value** dialog box.

E. Click **Finish** again to close the **Modify Global Constant** dialog box.



F. Deploy the BMIDE template.

The custom properties that you added to the newly created extended attribute object become available in the target business object and its revision as compound properties. For example, the **Supplier** property that you added to the **PartCustomProperties** object becomes available to **Part** and **PartRevision**. Additionally, these custom properties are also added to the BOM line objects.

After adding the custom properties, set up the custom properties for indexing, if needed.

2. For the **Assembly Indicator** property that is available by default, perform the following additional steps:

- The default value is set as **Fixed Assembly**. You can **change the default value**.
- For parts existing in Teamcenter from before engineering BOM management was set up, **populate** the **Assembly Indicator** property with the default value that you set.
- For the newly added custom parts, add the **Assembly Indicator** property to the **CreateInput** operation descriptor.

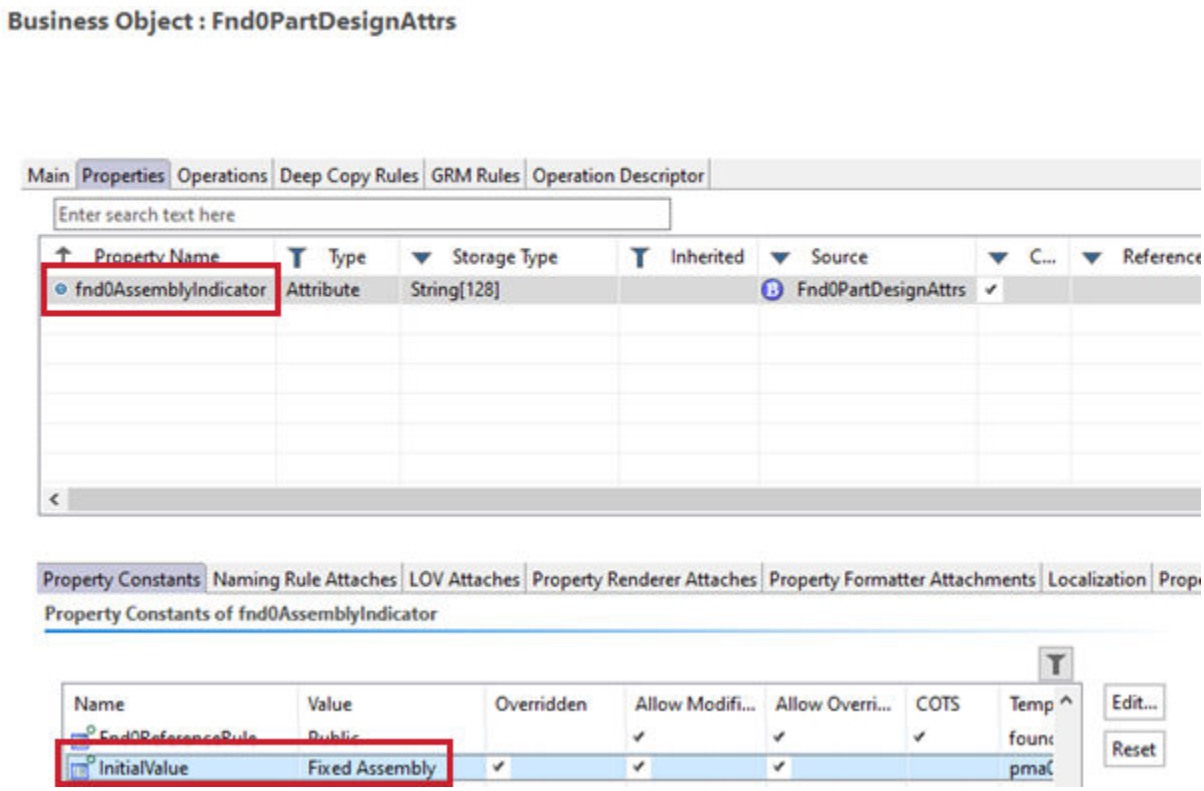
3. Add occurrence properties, if needed, to a subclass of **Fnd0AbstractOccAttrs** business object. After adding the occurrence properties, you can set up the properties for indexing.

4. Add the alignment-specific properties to the **Fnd0OccAlignmentAttrs** business object. This business object contains the default alignment related properties

5. Enable the display of the custom properties as columns, if needed.

Set a default value for assembly indicator

The default value of the **fnd0AssemblyIndicator** of the **Fnd0PartDesignAttrs** business object is empty. When the **Product Master Automation** template is installed, you must set the initial value of **fnd0AssemblyIndicator** to some value such as **Fixed Assembly**. You can set any value from the **Fnd0AssemblyIndicator** list of values (LOV). On doing so, the **Assembly Indicator** field of a newly created part is set to the specified initial value.



Additionally, you must add Assembly Indicator to the parts that already exist in Teamcenter.

Populate the assembly indicator property for parts that already exist in Teamcenter

After you set a default value for the **Assembly Indicator** property, run the following utilities include the default value for parts that are already available in the Teamcenter database:

- **fnd0_create_part_design_attrs_data** – Creates **Fnd0PartDesignAttrs** objects.

  fnd0_create_part_design_attrs_data -u=*user_name* **[-p=***password* **| -pf=***password_file***] -g=***group-***asmIndInitialValue=***initial_value*

- **pma0_create_part_attr_data** – Creates **Fnd0PartDesignAttrs**, **Pma0PartAttrs**, and **Pma0PartRevAttrs** objects.

  pma0_create_part_attr_data -u=*user_name* **[-p=***password* **| -pf=***password_file***] -g=***group-***asmIndInitialValue=***initial_value*

  Note:

  If you run the **pma0_create_part_attr_data** utility, you can skip running the **fnd0_create_part_design_attrs_data** utility as **pma0_create_part_attr_data** creates **Fnd0PartDesignAttrs** objects.

Add custom occurrence properties for parts

By default, certain occurrence properties are available for parts. You can add additional occurrence properties, if needed. You can specify to which item types these properties must belong to in the **FND0_OCC_ATTRS_VALID_ITEM_TYPES** preference.

1. Open the required BMIDE template project.

2. Locate the **Fnd0AbstractOccAttrs** business object.

3. To add additional occurrence properties, create a new business object (for example, **Ebm8CustomOccAttrs**) as the subtype of **Fnd0AbstractOccAttrs**.

4. Open the newly added business object and go to the **Properties** tab. Click **Add** to add new custom properties of the type **Attribute**.

5. For each custom property that you added, a runtime property of the type **Configured Runtime** is added to the **BOMLine** business object. The name of this runtime property is set in the following format:

   bl_occ_*BusinessObjectName_PropertyName*

   For example, if you added a custom property named **fnd0SkipAlignment**, the name of the runtime property would be:

   bl_occ_Ebm8CustomOccAttrs_fnd0SkipAlignment

The configured runtime property also gets added to another runtime business object, **MEViewLine**. To verify this, locate the **MEViewLine** business object, go to its **Properties** tab, and search for the runtime property, for example, **bl_occ_Ebm8CustomOccAttrs_fnd0SkipAlignment**.

6. Map the runtime property added to the **BOMLine** business object with the **Awb0Element** business object. To do so:

   a. Locate the runtime property in the **Properties** tab of **BOMLine**.

   b. Click **Map Property**.

   c. In **Map To Type**, browse and select **Awb0Element**, and click **Finish**.

   Similarly, map the runtime property of **BOMLine** with other business objects starting with **Awb0**, for example, **Awb0ConditionalElement**, **Awb0PartElement**, and **Awb0PositionedElement**.

After adding the occurrence properties, you can set up the properties for indexing. Finally, save and deploy the data model.

Display custom part and design properties as columns

You can display any custom properties added to a default or custom **Part** and **Design** business object as one of the columns when users load a part or a design.

Procedure

1. Export the default column configuration file for **AWClient** in which the parameter **uri** is set as **Awb0OccurrenceManagement**. To do so, you run the **export_uiconfig** utility at the Teamcenter command prompt.

   For part properties:

   export_uiconfig -u=*tc_admin_user_name* **-p=***tc_admin_password* **-g=***admin_group* **-file=***path_of_the_output_file* **-client_scope_URI=Awb0OccurrenceManagement.Part Revision**

   For design properties:

   export_uiconfig -u=*tc_admin_user_name* **-p=***tc_admin_password* **-g=***admin_group* **-file=***path_of_the_output_file* **-client_scope_URI=Awb0OccurrenceManagement.Design Revision**

2. Open the output file that you specified in the **-file** argument of the **export_uiconfig** utility.

3. Include the part properties or design properties in this file.

For example:

```
<ColumnDef ObjectType="Awb0PartElement"
propertyName="awb0CustomPartProperty" width="100"/>
```

4. For part properties, set the **name** and **uri** as **Awb0OccurrenceManagement.Part Revision**. For design properties, set these as **Awb0OccurrenceManagement.Design Revision**.

5. Import the new column configuration file:

   import_uiconfig -u=*tc_admin_user_name* **-p=***tc_admin_password* **-g=***admin_group* **-file=***path_of_the_file_to_be_imported* **-action=merge**

Set a software artifact as an engineering part

Certain parts in a product need to be programmed using embedded software artifacts, such as a software application, so that the parts function in a specific manner.

> Example:
>
> The part *Tire Pressure Sensor* of a product *Crosskart* can be programmed in such a manner that when the air pressure drops below the product's recommended level, the sensor captures this information, and the dashboard indicator lights up.

Software developers create and maintain software artifacts using the Embedded Software Management functionality. You must set the software artifacts as engineering parts so that users can include the software artifacts in a product structure. The software artifacts **Software**, **Application Software**, **Calibration Software**, **Configuration File**, **License**, **Primary Bootloader**, and **Secondary Bootloader** are set as engineering parts by default. For any software artifact other than these, you must set it as an engineering part. However, if the software artifact is a child of **Software**, it is automatically set as an engineering part.

Procedure

1. Add the new software artifact (for example, **Custom_Software**) as a valid value in the following preferences:

   - **TcAllowedChildTypes_Software**
   - **TcAllowedChildTypes_AppSoftware**
   - **TcAllowedChildTypes_Calibration**
   - **TcAllowedChildTypes_ConfigFile**
   - **TcAllowedChildTypes_Ess0License**

- **TcAllowedChildTypes_PriBootloader**
- **TcAllowedChildTypes_SecBootLoader**

2. Add the software artifact (for example, **Custom_Software**) in the **TCAllowedChildTypes_Part** preference so that it can be included in a part structure.

3. (Optional) Create a **TcAllowedChildTypes** preference for the new software artifact (for example, **TcAllowedChildTypes_Custom_Software**) and include a list of valid item types that can be included as children of the software artifact. For example, you can include a document to be a valid item type so that users can add software documents to a software artifact. However, you cannot specify a **Part** or **Design** to be a child of a software artifact.

   The protection scope of this preference is **Site**.

Set custom rules to validate engineering BOM data

By default, Teamcenter provides a few rules to validate engineering BOM data, that is authored and updated in the context of an active change notice. After the data is validated, it can be released. You can also create additional custom validation rules.

The default validation rules are as follows:

Part related business rules	Information
PMA0_validate_part_and_aligned_design_maturity	The maturity of the part cannot be higher than the maturity of the aligned design. By default, the maturity property of a design is not considered for the validation checks. This must be done by setting the **Pma0IsMaturitySupportedForDesign** preference to **True**.
PMA0_validate_parent_child_maturity	The maturity of the parent part cannot be higher than the maturity of the child part.
PMA0_validate_part_design_material	The part cannot be released if the material associated with it is different than the material of the aligned design. To set this validation rule, ensure that you have set up Integrated Material Management.

Procedure

1. To create a custom validation rule, create a new BMIDE template project, and add the following as the dependent templates:

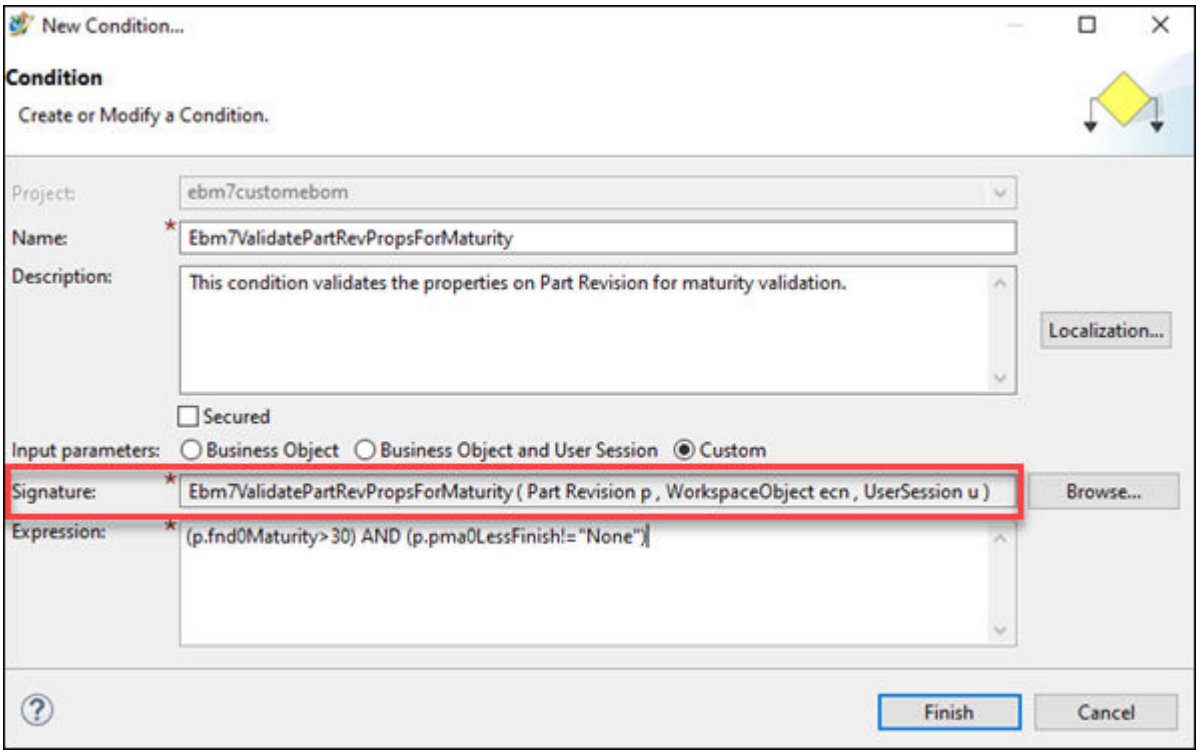
   - **pma0automation**
   - **ebm0enterprisebom**

2. Expand the template project, and under **Extensions**, add the new extension (schema) file that contains the custom parts. Next, set this file as the active extension file.

3. Author a new BMIDE condition within this template.

   The **Signature** of the condition must have the following 3 parameters:

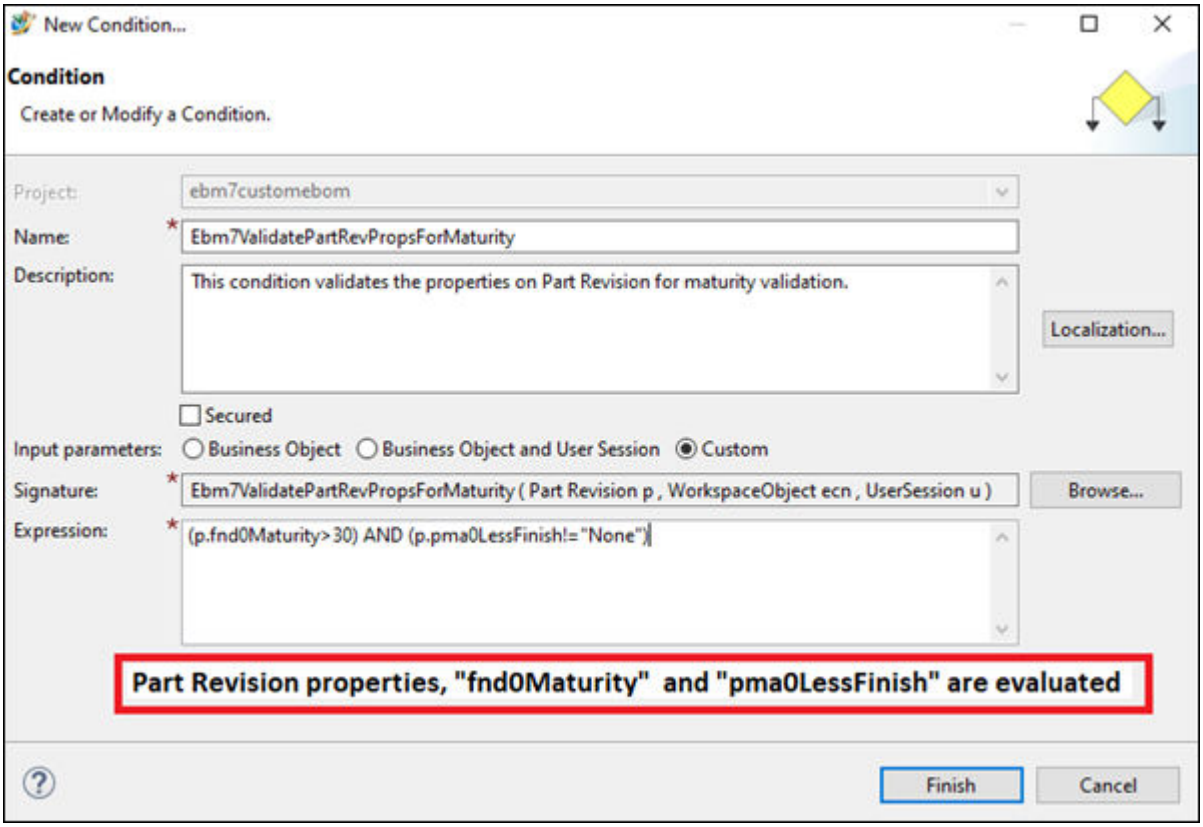
Parameter	Description
Part Revision	This can be any business object that is qualified to be tracked under an active engineering change notice for the engineering BOM. Examples include **Part Revision, Design Revision**. You can also use business objects of the type **ItemRevision** and its child types which are **qualified as engineering parts and designs**.
WorkspaceObject	This must be an engineering change notice.
UserSession	This is a user session object that controls the access-related checks in the code.



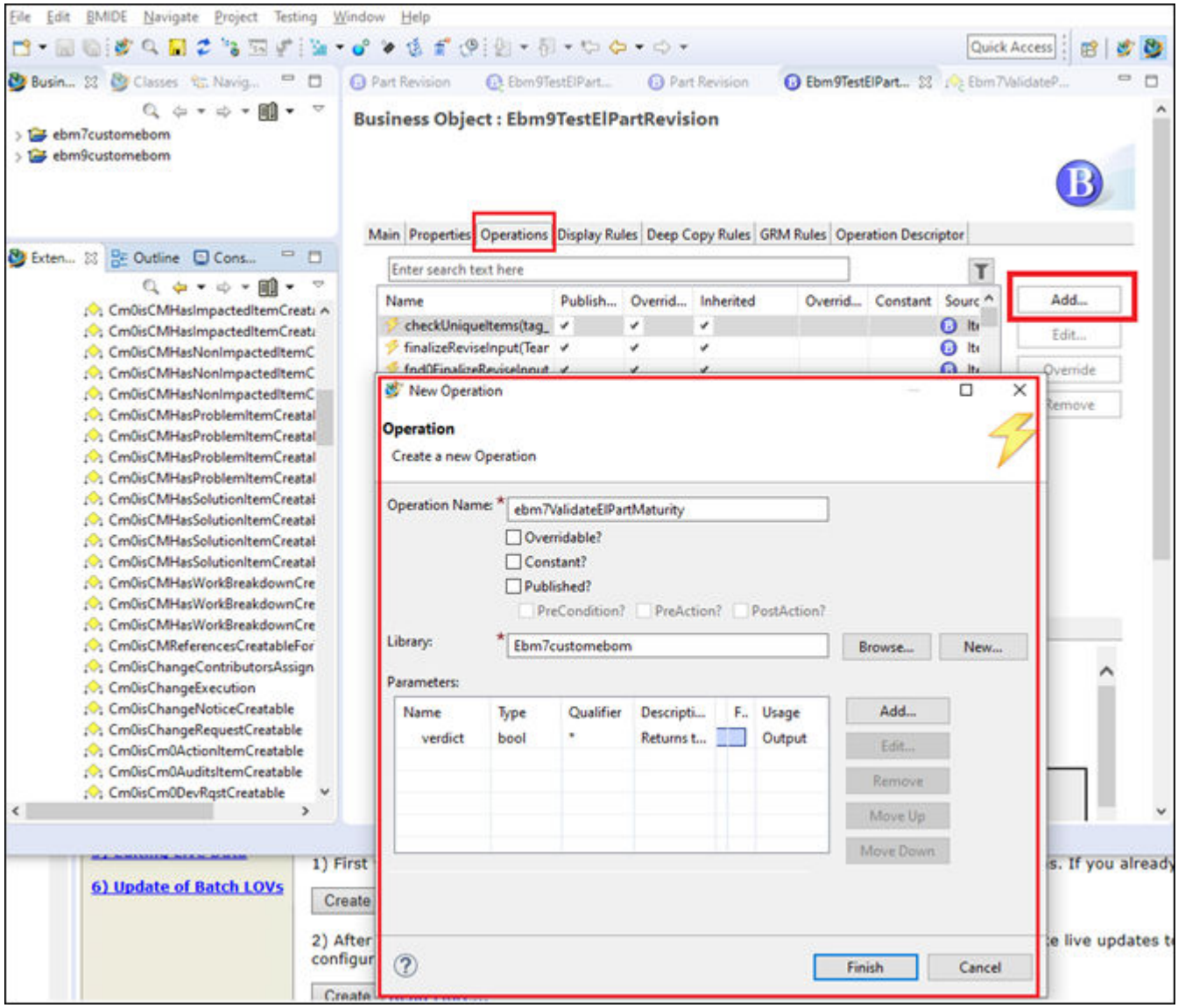
   The **Expression** can be **direct logic** written on a property of the business object (used in parameter 1, **Part Revision**), or it can be an **operation** written on the business object.

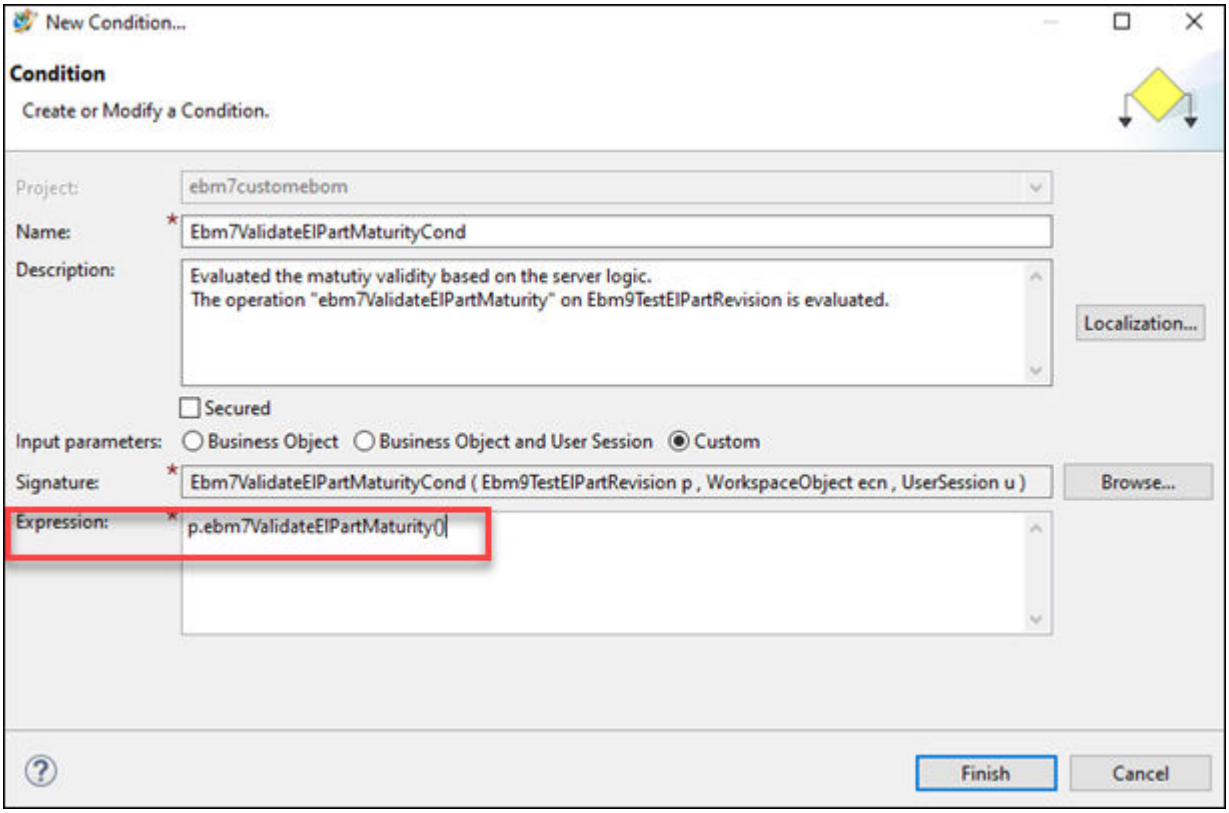
4. Write the logic in **Expression** using a property of the business object:

   For example, (p.fnd0Maturity > 30) AND (p.pma0LessFinish != "None")



5. Author an operation on the business object (used in parameter 1, **Part Revision**), and add the operation in the **Expression** field.





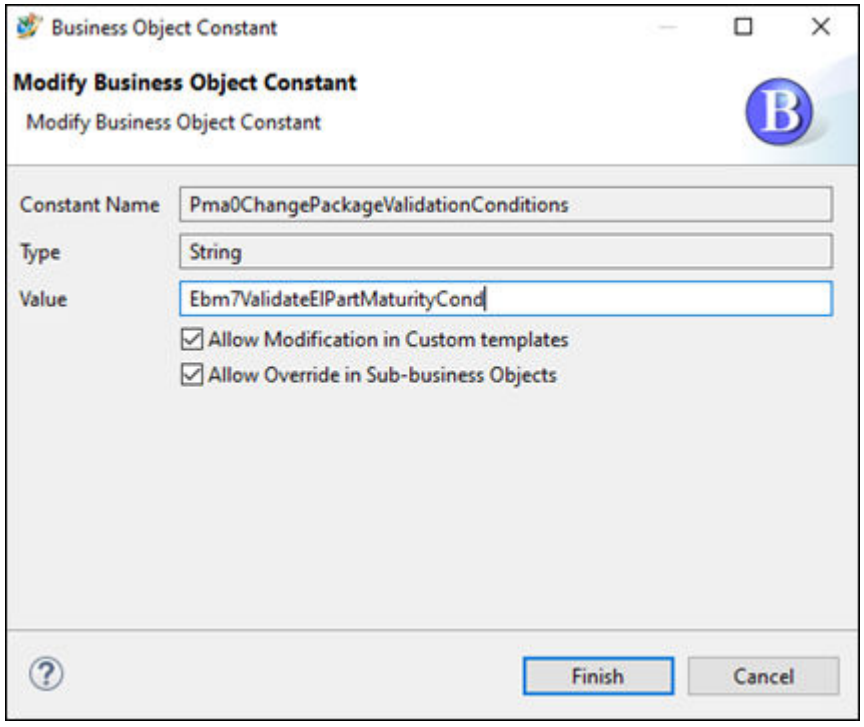
Now, you can generate the C++ classes to implement the operation with the custom logic.

6. Append the custom validation rule to the **Pma0ChangePackageValidationConditions** constant of the respective business object.

   a. Locate the required business object (for example, **Part Revision**).

   b. In the **Business Object Constants** tab, locate **Pma0ChangePackageValidationConditions**, and click **Edit**.

   c. In the **Modify Business Object Constant** dialog box, add the custom rule that you created as a comma-separated value. Here, you can choose to remove any default validation rules from the list.



7. Save and deploy the template.

Display custom icons for business objects in the context search results

As an administrator, you can customize the display of different (custom) icons based on the property of the derived types **Awb0Element** or **Item Revision**. Once you have added custom icons, you must perform the following steps to ensure that these icons are displayed properly in the search results.

1. Modify **AwDataNavigatorViewModel.json** to include objects and properties in the **aceSearchPolicyOverride** section.

2. Modify **occMgmtPropertyPolicy.json** to include objects and properties used in icon customization.

Consider the following example where a custom icon **typeCustomIcon48** is added. The steps required to ensure that the custom icon is displayed properly in the search results are as follows:

For example, you add the following entry in **typeIconsRegistries** to **typeCustomIcon48** when **someCustomProperty** of the underlying object has the value **Test**. For more information about using the **typeIconsRegistry** file, see Registering icons (advanced) .

```
{
        "type": {
          "names": [
```

```
        "Awb0DesignElement"
      ],
      "prop": {
        "names": [
          "awb0UnderlyingObject"
        ],
        "type": {
          "names": ["ItemRevision"],
          "prop": {
              "names": ["someCustomProperty"],
              "iconFileName": "typeCustomIcon48",
              "conditions": {
                  "object_name": {
                    "$eq": "Test"
                  }
              }
          }
        }
      }
    },
    "priority": 11
  }
```

You must perform the following steps to display the custom icons added in the above example to the structure search results on Active Workspace.

Caution:

Before upgrading Active Workspace, you must back up the *AwDataNavigatorViewModel.json* and *occMgmtPropertyPolicy.json* files. And after upgrading Active Workspace, you must merge these files.

1. Modify **AwDataNavigatorViewModel.json** to include **ItemRevision** and **someCustomProperty** in the **aceSearchPolicyOverride** section.

```
"aceSearchPolicyOverride" : {
            "types": [ {
                "name": "Awb0Element",
                "properties": [ {
                        "name": "awp0ThumbnailImageTicket"
                    }, {
                        "name": "object_string"
                    }, {
                        "name": "awp0CellProperties"
                    },
                    {
                        "name": "awb0BreadcrumbAncestor",
```

```
                    "modifiers": [ {
                        "name": "withProperties",
                        "Value": "true"
                    } ]
                },
                {
                    "name": "awb0UnderlyingObject",
                    "modifiers": [ {
                        "name": "withProperties",
                        "Value": "true"
                    } ]
                }
            ]
        }, {
            "name": "Fgd0DesignElement",
            "properties": [ {
                "name": "awb0UnderlyingObject",
                "modifiers": [ {
                    "name": "withProperties",
                    "Value": "true"
                } ]
            } ]
        }, {
            "name": "Cpd0DesignElement",
            "properties": [ {
                "name": "cpd0category"
            } ]
        }, {
            "name": "Wbs0ElementCondElement",
            "properties": [ {
                    "name": "wbs0IsWorkElement"
                },
                {
                    "name": "wbs0RevObjectType"
                } ]
        }, {
            "name": "ItemRevision",
            "properties": [ {
                    "name": "someCustomProperty"
                } ]
        } ]
}
```

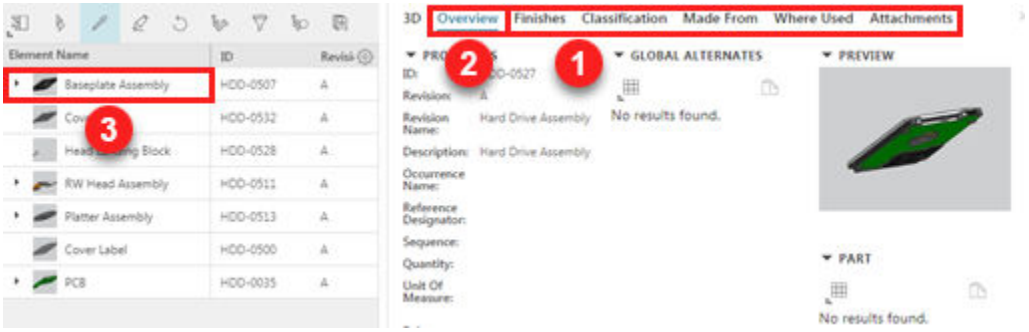
2. Modify **occMgmtPropertyPolicy.json** to include **awb0UnderlyingObject** with properties and the **ItemRevision** object with **someCustomProperty**.

11. Tasks to administer engineering BOM management

12. Enable the display of structures on Active Workspace

Configure the Content tab

The **Content** tab displays the content of a structure. The following diagram shows the configurable parts of the **Content** tab.



Number	Element	How to configure
1	Occurrence sublocation tabs	Add the following preference: **AWC_***item-revision-type***.showObjectLocation.OccurrenceManagementSubLocation.SUMMARYRENDERING**. The value of the preference is the name of the style sheet dataset to take effect.
2	**Overview** tab	Use the summary style sheet of the associated element representing the occurrence. For example, use the **Awb0DesignElementSummary.xml** XML rendering style sheet file to configure the **Overview** tab for any design related item revision and use the **Arm0RequirementElementSummary.xml** file for requirement revisions.
3	Occurrence cell properties	Set the cell properties preference for the occurrence. Two examples are the **Awb0DesignElement.CellProperties** preference and the **Arm0RequirementElement.CellProperties** preference.

Tip:

To show the in-context search icon run the index on the BOM.

To view the **Architecture** tab, set the **Awb0AvailableFor** business object constant on the **Ase0ArchitectureFeature** business object in the Business Modeler IDE. Set the **Awb0AvailableFor** business object constant to list the business object types for which a feature should be made available, for example:

```
Functionality,Fnd0LogicalBlock,RequirementSpec,Requirement,Paragraph,Fnd0SystemModel
```

Modify the display name of the Content tab

You can modify only the display name of the **Content** tab by changing the value of **tc_xrt_Content** in the *activeworkspacebom_text_locale.xml* file. You must not change the **titlekey**.

Now you can update text server to override existing display names.

Control the visibility of commands in the Content tab

Because Active Content uses intermediary objects to represent their underlying objects in the **Content** tab, when you select one of these objects, you're actually selecting the intermediary object. Any declarative functionality, like conditions for example, are based on that selected intermediary object. However, there are many times when you will want the declarative client to base its decisions on the underlying object instead. Command visibility is the most common example of this.

Active Workspace provides the **AWC_AlternateSelectionCommandsList** preference to tell Active Content which commands must consider the underlying object instead of the intermediary object when determining their visibility. All commands listed by this preference will look to the *target object of the selected object* instead of the selected object itself when determining whether it is visible.

The shipped commands that are already available in the **Content** tab are already added to the preference. As an administrator, you can check them before adding new commands.

Add an LOV to a property in the Content tab

An LOV attached to the **BOMLine** property to which the **Awb0Element** property is mapped is not automatically displayed on the **Content** tab in Active Workspace. To display the LOV in the **Content** tab, you must attach the same LOV to the mapped **Awb0Element** property.

This applies to all children of the **Awb0Element** as well.

For more information, see *Attach an LOV to a property* in *BMIDE for Data Model Design*.

Choose properties to be displayed while adding structure elements

A user can create or update a structure on the **Content** tab in Active Workspace by adding new or existing structure elements to it. By default, while adding an element to a structure, users can specify **Quantity** in the **ELEMENT PROPERTIES** section. Depending on your site requirements, you can include additional element properties to be displayed on the **Add** panel in Active Workspace. To do so:

1. In BMIDE, locate the **PSOccurrence** business object.

2. In the **Operation Descriptor** > **CreateInput** tab, click **Add**.

3. In the **New Operation Input Property** dialog box:

   a. Select **Add a Property from Business Object** and click **Next**.

   b. Browse and add the required property in **Source Property**, for example, **ref_designator**.

4. Click **Finish**.

   Similarly, add other properties that you want to display in the **ELEMENT PROPERTIES** section.

After adding the required properties, you must update the **Awb0PSOccurrenceCreate.xml** style sheet to include these properties. For this:

1. Navigate to *TC_ROOT\install\activeworkspacebom\data* and open the **Awb0PSOccurrenceCreate.xml** file.

2. Add the newly added properties. For example, to add the **ref_designator** field:

```
<rendering>
    <page>
        <view>
            <property name="qty_value" />
            <property name="ref_designator" />
        </view>
    </page>
</rendering>
```

The element properties that you add are available for all business objects of type **Item** by default. If you do not want the properties to be available for certain business objects, add those objects as **Values** in the **AWBItemTypesToHideOccurrenceCreatePanel** preference.

By default, the **ELEMENT PROPERTIES** section is displayed in the **New** tab of the **Add** panel for the **Item** and **Part** business objects only. If you want this section to be displayed for other custom objects, you must update the create style sheet of those objects and append the following section:

```
<section>
          <inject type="dataset" src="Awb0ElementCreate"/>
</section>
```

Configure the properties of structured content

You can display the properties of the underlying object (called the **archetype**) in a structure element object tile. For example, you can show the release status of an item revision on the tile of an element in a structure using the following syntax: **awb0Archetype.release_status_list**.

You can use the home page of Active Workspace to provide tiles for your users' most commonly used pages, objects, and saved searches. For more information about defining object tiles, see *Organizing your users' common destinations* in *Active Workspace Customization*.

Specify the archetypes or underlying types that support creation of structures

The **Awb0SupportsStructure** global constant controls the types that can be opened in the active content explorer. This constant can take multiple values and each value is the name of a type that supports structure. Only the names of types that are specified in this constant can be opened in the explorer. The structure property is not inherited by subtypes, that is, you must add each subtype separately.

By default, this constant includes **ItemRevision**, **DesignRevision** and **PartRevision** as the values. For BVR structures, use the **ItemRevision** type as the value.

For more information, see *Change the value of a global constant* in *BMIDE for Data Model Design*.

Display the thumbnail instead of the icon for a structure element

Use the **AWB_ShowTypeIcon** preference to display either the thumbnail or the icon that indicates the type of a structure element.

The default value of the preference is **true**. In this case, the icon that indicates the type of the structure element is displayed everywhere except in the **PREVIEW** section. The icon is displayed even if the structure element has an attached thumbnail. For example, here, the item revision icon is displayed next to the Green Seat of Crosskart.

If you want to display the thumbnail for a structure element, set the value of the preference to **false**. When you do this, the thumbnail is displayed for the structure element if the structure element has a thumbnail attached to it. If the thumbnail is not attached, the icon is displayed. For example, here, a thumbnail showing a green seat is displayed next to the Green Seat of Crosskart.

Enable the display of connections and item elements

You can enable the display of connections (for example, welds) and item elements (for example, ports) in the tree structure and the 3D viewer in Active Workspace, using the following preferences:

Preference	Description
AWBShowConnections	Enables the display of connections, such as welds.
AWBShowPorts	Enables the display of item elements, such as ports.

The default value for these preferences is **false**. To enable the display, set the value for these preferences to **true**.

Simplify the user interface by hiding the configuration parameters that the user does not need

By default, certain configuration parameters are always displayed in the configuration header and configuration panel in Active Workspace. You can choose to hide the configuration parameters that you do not need from view. To do this, use the **AWBEnabledStructureFeatures_Item** preference.

This preference is based on the type of the top-level element in a structure. By default, this preference is available for all item types. You can define a different configuration for any child type. For example, if you have a part or a design or a requirement as the top-level element in your structure, you can create the preference **AWBEnabledStructureFeatures_Part** or **AWBEnabledStructureFeatures_Design** or **AWBEnabledStructureFeatures_Requirement**, respectively.

From the preference, you can remove the values for the configuration parameters that you do not need. For the values that you remove, the corresponding configuration parameters are removed from the configuration header and configuration panel. In the menus on the user interface, the corresponding commands for the configuration parameters are made unavailable.

Based on your need, choose to remove a value from the relevant preference as follows:

To hide	Remove
The *unit effectivity* in the configuration header and the configuration panel.	The value **Awb0UnitEffectivityConfigFeature** from the preference.
The *variant* in the configuration header and the configuration panel.	The value **Awb0VariantFeature** from the preference.
The *expansion rule (closure rule)* in the configuration header and the configuration panel.	The value **Awb0ClosureRuleFeature** from the preference.

Note:

To prevent the business user from authoring *date effectivity*, you can remove the value **Awb0DateEffectivityConfigFeature** from the preference. Even if you remove this value from the preference, date effectivity is always displayed in the configuration header and the configuration panel.

Specify the maximum number of results to be displayed in the Used in Structures section

A business user can view an item revision across all assemblies up to the top level in the **Used in Structures** section on the **Where Used** tab. As a system administrator, you can specify the maximum

number of results that can be displayed in this section using the **TC_WhereUsed_Display_Limit** preference.

By default, a maximum of 50,000 results are displayed in the **Used in Structures** section. If you change this value in the preference, for example, to 20,000, a maximum of 20,000 results are displayed. If the number of available results exceeds this threshold value, only the first page of the results is displayed. Active Workspace subsequently cancels the export process, and the system displays an error message.

If you specify a very high value, one that is greater than 50,000, Active Workspace might become unresponsive and not display any results.

Configure the packing of structure elements with units of measure

You can configure how a structure element with a unit of measure is packed. To do this, you set the **BOM_Additional_Packing_Criteria** preference with the value **bl_uom**.

In this case, the structure elements with different units of measure are packed separately. The elements with the same unit of measure are packed together as long as the **Quantity** is not specified as As Required (**A/R**) and the **Find Number** value is the same. The elements for which the **Quantity** is specified as **A/R** are not packed together.

Configure the display of the quantity value for a structure element

You can configure how the quantity value is displayed for a structure element by specifying a valid value in the **BOM_manage_quantity_display_value** preference:

> Note:
>
> This preference works only on the elements that are not packed.

- **False**: This is the default value. If this value is set in the preference and if the quantity is not already specified, the quantity is displayed as blank irrespective of what unit of measure is specified.
- **True**: If this value is set in the preference and if the quantity is not already specified, the unit of measure setting determines how the quantity is displayed:

  When the unit of measure is specified as **each**, the quantity is displayed as 1 and when it is not specified as **each**, the quantity is displayed as zero (0).

Configure the display of the reference designator value for a structure element

You can configure the display of the reference designator property and validate its format and value.

Procedure

1. For this, you can use the following preferences:

To do this	Set the value to the preference
To pack the product structure lines that include reference designators.	Set the **true** value to the **BOM_Enable_Ref_Designator_Value_Packing** preference. For example, eight BOM lines with the reference designators **C1, C5, C6, C7, C10, C14, C15, C16** will be packed to one BOM line with the reference designator property **C1, C5-7, C10, C14-16**.
In packed mode, view the values in a range in the **Reference Designator** field.	Set the **Range** value to the **BOMRefDesignatorPackMode** preference. For example, C1-C5.
In packed mode, view the concatenated values in the **Reference Designator** field.	Set the **Concatenate** value to the **BOMRefDesignatorPackMode** preference. For example, C1,C2,C3,C4,C5.
Set the separator for the **Reference Designator** values in concatenate mode.	Set the **,** (comma) value to the **BOMRefDesignatorPackSeparator** preference to see them comma separated.
Validate whether the format and uniqueness of the reference designator values must be checked while summarizing.	Set the **PS_Reference_Designator_Validation** preference to **true** to validate the format and uniqueness of the reference generator values. For example, the format can be one or more uppercase letters followed by integer numbers like C7 or C18. The preference also checks if the values of reference generator are unique.
Validate whether the values of the reference designator are to be verified with the structure element quantity.	Set the **BOM_Enable_Quantity_Validation_Against_Ref_Designator** preference to **true**, so that the quantity of the structure element is checked against the number of reference designator values. For example, if there are four capacitors, the reference designator property must also contain 4 values.

Configure the child items in a structure with the parent variant conditions

You can configure whether a child item should be included or excluded from the structure using the variant condition of the parent item, without applying a variant rule. To do this, specify a valid value in the **PSM_nested_variant_solve_product_types** preference. In a nested structure with multiple parent items, the variant condition of the topmost parent item takes precedence in determining the inclusion of child items.

- **None**: If this value is set in the preference, the child items are not configured in a structure using the variant condition of the parent items of no topmost item type.
- **All**: If this value is set in the preference, the child items are configured in a structure using the variant condition of the parent items for topmost item of all types.
- Enter a value of an item type for which you want the nested variant configuration to be available. The default value is set to **Fnd0AbsDgnProd**.

For example, suppose that the variant formula for a fuel tank assembly is set as weight equal to **V**. In such a case, all the child items of the fuel tank assembly whose variant formula does not contain to weight equal to **V** are automatically excluded from the structure.

Additionally, you can configure whether a child item should be included or excluded from the structure using both the variant condition of the parent items and the variant rules. To do this, specify a valid value in both the **PSM_nested_variant_solve_product_types** preference and **PSM_enable_nested_variant_solve_on_apply_svr_only** preference.

The following are the valid values for the **PSM_enable_nested_variant_solve_on_apply_svr_only** preference:

- **false**: This is the default value. If this value is set to the preference, the nested variant is configured with or without a variant rule applied on the structure.
- **true**: If this value is set to the preference, the child items are configured in a structure only if a variant rule is applied on the structure.

You can also view the variability conditions for all items in a structure in the **Net Variability** field. The **Net Variability** field shows the variant conditions for a child item by adding the variant condition of the parent items in a nested structure.

13. Configure product structure creation and updates

Configuring the duplication (cloning) of structures

A structure is duplicated (cloned) using the **Duplicate** command. The cloning action is either executed at the **Item Revision** level or the **Occurrence** level. The site administrator must set the **Structureless** preference to configure the duplication behavior at either the item revision level or at the occurrence level. The default value of the preference is *False* and this implies item revision level duplicate.

```
<preference name="AWBUseOccurrenceLevelStructureClone" type="Logical"
array="false" disabled="false" protectionScope="Site" envEnabled="false">
<preference_description>
Defines if the structure clone operation should be executed at
occurrence
level.
Occurrence level structure cloning is supported from platform TC12.3
onwards.
If this is set to true and platform supports Occurrence level clone
execution,
structure clone operation is executed at Occurrence level.
If this is set to true and platfrom does not support Occurrence level
clone
execution, clone operation is executed at Item Revision level.
If this is set to false, structure clone operation is executed at Item
Revision level.
</preference_description>
<context name="Teamcenter">
<value>false</value>
</context>
</preference>
```

You can customize the duplication operation by implementing the following user exits to:

- Override the duplication for certain BOM lines, for example, standard parts, using the following preprocess user exit.

```
/**
Gets the operation for a BOMLine during structure clone.
<br/>If @p use_default_operation is true, @p duplicate_operation will
be ignored.
Default opeartion configured by the system will be used.
<br/>If the user exit is not overridden, default opeartion configured
by the system will be used.
```

```
The following are the valid operations for @p duplicate_operation.
<ul>
<li>#STRUCTURE_CLONE_OPERATION_CLONE
<li>#STRUCTURE_CLONE_OPERATION_REFERENCE
<li>#STRUCTURE_CLONE_OPERATION_REVISE
<li>#STRUCTURE_CLONE_OPERATION_REPLACE
<li>#STRUCTURE_CLONE_OPERATION_IGNORE
</ul>

@returns
<ul>
<li>#ITK_ok on success.
<li>#BOM_invalid_tag if the @p bomline is invalid.
</ul>
*/
extern TCCORE_API int USER_bom_clone_get_operation(
const tag_t     bomline,
/**< (I) BOMLine to get duplicate operation. */
bool*           use_default_operation,
/**< (O) Use default duplicate operation for bomline. */
int*            duplicate_operation
/**< (O) Duplicate operation for bomline. */
);
```

- Establish a post-process equivalence between the source and cloned lines using the following post-process user exit. This user exit is invoked only once after the duplication process is complete. It establishes a relationship between the source and the cloned structures.

```
/**
Processes cloned objects after clone operation is complete.
<br/>Customizers need to replace the base action for the user exit
BMF_USER_bomline_process_cloned_structure to address their business
needs.

The following are cloning type options for @p cloning_type.
<ul>
<li>#ITEM_REVISION_CLONE
<li>#BOMLINE_CLONE
</ul>

@returns
<ul>
<li>#ITK_ok on success.
<li>#BOM_invalid_tag if the @p cloned_top_item_rev is invalid.
</ul>
*/
extern BOM_API int USER_bomline_process_cloned_structure(
 const tag_t     cloned_top_item_rev,
/**< (I) Top Item Revision of the cloned structure. */
```

```
 const int cloning_type,
/**< (I) Cloning type for clone operation. Expected values are:
<ul>
<li>#ITEM_REVISION_CLONE
<li>#BOMLINE_CLONE
</ul>
*/
const int n_source_bom_lines,
/**< (I) Number of BOMLines to be cloned. */
const tag_t*    source_bom_lines,
/**< (I) n_source_bom_lines List of BOMLines to be cloned. */
const tag_t*    cloned_occurrences
/**< (I) n_source_bom_lines List of cloned PSOccurrences. */
```

Note:

The **USER_bom_clone_process_cloned_structure** post-processor exit does not work. Use the **USER_bomline_process_cloned_structure** post-process user exit for this.

Enable the display of red lines to indicate structure changes

With a change notice set as an active change, BOM modifications and properties are identified by redlines. Your users can review active or closed changes associated with any structure. To enable highlighting the changes with red strikethroughs or in italicized green text, as the system administrator, you must set the **AWC_Enable_RedLine_feature** preference to **TRUE**.

In addition to this, you can also set the following preferences:

To do this	Set the preference to the designated value
Review the changes for the **Reference Designator** and **Quantity** fields in the structure.	Set the value of the **CM_bomline_tracked_properties** preference, to **bl_ref_designator** and **bl_quantity**.

Configure occurrence removal access for an aligned engineering BOM and design structure

In scenarios where a BOM engineer creates a part occurrence and a design engineer creates a design occurrence, an alignment between the two generates a business object called **PublishLink**, which is owned by the engineer who creates the alignment.

Example:

If a BOM engineer creates the alignment, they own the **PublishLink**. If the design engineer tries to delete the aligned design occurrence or the alignment itself, the system attempts to remove

the **PublishLink**. Since the design engineer does not own the publish link, this action results in an access error.

Conversely, if the design engineer creates the alignment, they will own the publish link, and if the BOM engineer tries to remove the aligned part occurrence or the alignment, they also encounter an access error.

To prevent such errors, both the BOM engineer and the design engineer must have access to the **PublishLink** object.

Prerequisites

Ensure that you have the Access Manager license.

Procedure

To grant access:

1. Launch the Access Manager.

2. Create an access rule for the required set of users for the **PublishLink**.

Configure advanced search options within a structure

Suppose that your Teamcenter setup has the Context Management User license and the structure is indexed using Smart Discovery Indexing. In such a case, you can allow users to use the advanced search options within a structure by setting the **AW_Discovery_Advanced_Filter** preference to **true**. If you set this preference to **true**, users can create multiple OR groups for filter criteria, choose a group to modify the filter criteria, and also add a NOT group to the filter expression.

If you set the preference to **false**, users can use the filter expression with NX.

14. Configure BOM line properties

Adding compound properties

Introduction to compound properties

A *compound property* on a business object (the *display* or *target* object) effectively adds some property defined on another object (the *source* object) to the display object so that the source property behaves on the display object as if it is defined on the display object class.

A compound property defines a path between the display object and source object consisting of one or more reference relations, GRM relations, or a combination of the two. The path is used to traverse between the display object and the source object. You can add compound properties to COTS and custom business objects.

> Note:
>
> While a compound property enables you to add a property to a business object without writing custom code or using runtime properties, a compound property is not a replacement for runtime property functionality.

Set a compound property on a BOM line

If you want to set a compound property on a BOM line object in Structure Manager that displays an in-context value or incremental change value, you must use the **BOMLineAbsOccCompProperties** global constant. You can also create a compound property directly on a BOM line object, but it does not display the value of in-context edits (that is, the absolute occurrence value) or incremental change edits of the source property.

1. Set a value on the **BOMLineAbsOccCompProperties** global constant to create a compound property for use on a BOM line, for example:

   ```
   FORM::IMAN_specification::BVRSyncInfo::PROPERTY::BVRSyncInfo::
   last_sync_date::Absocc Last Sync Date
   ```

   This sample compound property displays the value of the **last_sync_date** property; **Absocc Last Sync Date** is the display name of the property.

2. Save the data model and deploy to the rich client.

3. To verify the compound property, perform the following steps in the rich client:

   a. Create a simple structure and send it to Structure Manager.

   b. Right-click the top line and select **Set In Context**.

c. Click the **Show/Hide Data Panel** button in the toolbar.

d. Select the child object in the structure and choose **File→New→Form**. Do this in context mode because the expectation is to display it only when the parent is loaded.

   The form is created as an attachment to item revision with the specification relation. The **Context Line** column in the **Attachments** tab of the data pane shows the context of the created form.

e. Open the form and edit the attribute values.

f. In Structure Manager, right-click the column headers, choose **Insert Columns**, and select the property you added using the global constant. The display name is the last item in the constant, for example, **Absocc Last Sync Date**.

   The column is added, and the value for that property is displayed in the column.

Make compound properties visible on item revisions

You can configure attributes on the item master form, and then use compound property rules so that the attributes are visible on item revisions. To achieve this, you must define and set the following global constants in the **foundation** template of the Business Modeler IDE:

- **BOMLineFormConfiguredProperties**

  Adds the properties from the form types in the constant to the BOM line. This configuration point was provided by the **PSE_add_props_of_item_form_types** preference in previous versions.

- **BOMLineRevConfiguredProperties**

  Adds the properties from the revision form type to the BOM line. This configuration point was provided by the **PSE_add_props_of_rev_form_types** preference in previous versions.

If either constant is set, all the form properties from the form type are added to the BOM line. By default, the foundation template sets the values of the constants to the item master and item revision master, respectively.

You can then add the attribute to display as a column. The attribute name is shown in fully qualified format, for example, if your item master attribute name is **A_SPLM**, the column name is shown as **bl_Item_Master_A_SPLM**.

You can optionally change the displayed column name.

Add a compound property on a GDE line using the bl_line_object property

Use the **bl_line_object** property instead of the **bl_revision** property to create a compound property to display the underlying GDE object's property on the GDE line.

General design element (GDE) objects are nonrevisable objects. These objects can appear in Structure Manager as GDE lines, just as items are seen in Structure Manager as item lines. In Structure Manager terminology, both GDE lines and item lines are BOM lines. Currently, Structure Manager can display properties of these BOM lines. Some of these properties are specific to item lines and others to GDE lines, while some are common to both GDE lines and item lines.

BOM line properties that are specific to item lines are not be applicable for GDE lines. For example, some item revision-specific BOM line properties are specifically applicable only to item lines and are not applicable to GDE lines. Some of these have revision property-specific text in their names, and because GDEs are nonrevisable objects, these are to be used only for item lines.

Following are some examples of properties not applicable for GDE lines:

bl_revision
bl_revision_change
bl_sequence_no
bl_config_string

Creating display names for BOM line properties

About display names

BOM line properties have both display names and system names, and the system names are in the format **bl**_*xxx*. You should define a display name for a system name if you add a custom business object type to your system.

Define a display name for a BOM line using the **BOMLineFormConfiguredProperties** and **BOMLineRevFormConfiguredProperties** global constants and compound properties available on the **BOMLine** business object in the Business Modeler IDE.

Caution:

Siemens Digital Industries Software recommends that you do not include a dot (.) in the property name. Such names cannot be edited in Structure Manager.

Note:

In previous versions, the administrator could create or change display names by editing the **/textserver/xml** files in the **lang** folder, for example, **system_property_names_locale.xml**. Display

names are no longer stored in XML files, and you can now only create or edit them in the Business Modeler IDE.

Create a display name

1. In the Business Modeler IDE, open the **BOMLine** business object and click the **Properties** tab.
2. Select the property whose display name you want to create and scroll to the **Localization** section.
3. Click **Add**.

   The Business Modeler IDE displays the **Add or Modify Localization** dialog box.
4. Enter the display name in the **Value Localization** box, select the locale, and set the status to **Approved**.
5. Save the data model and deploy the template.

BOM line naming behavior

A BOM internally evaluates the BOM line name from the **DisplayName** business object constant on the **Item Revision** business object. However, the lightweight BOM (LWB) functionality determines the BOM line name from the **object_string** property of the **Fnd0ItemLineLite** business object as defined in the **Fnd0ItemLineLite_bl_line_name_expression** preference. The default value of this preference is similar to the naming convention used by regular BOM line.

The default value of the **Fnd0ItemLineLite_bl_line_name_expression** preference is as follows:

```
$bl_item_item_id+"/"+$bl_rev_item_revision_id+";"+$bl_rev_sequence_id+"-"+
$bl_rev_object_name
```

The lightweight BOM naming convention deviates from regular BOM only in cases of subtypes of **ItemRevision** business objects that change their **DisplayName** business object constant expression.

In the following example of two lines in a structure (one with **Item** type and the second with **Part** type), the **DisplayName** business object constant is changed on the **PartRevision** business object.

Item Type	Structure	Regular BOM line name	LWB BOM line name
Item	**ItemComp**	**0000BNL34/A;1-NAME** (View)	**0000BNL34/A;1-NAME** (View)

The **DisplayName** business object constant for this line is:

```
$bl_item_item_id+"/"+$bl_rev_item_revision_id+";"+$bl_rev_sequence_id+"-"+
$bl_rev_object_name
```

Item Type	Structure	Regular BOM line name	LWB BOM line name
Part	**PartComp**	**0000BNL34-NAME** (View)	**0000BNL34/A;1-NAME** (View)

The **DisplayName** business object constant for this line is changed from the default value to the following:

```
$bl_item_item_id+"-"+$bl_rev_object_name
```

Enable validation for find numbers

Teamcenter assigns a unique find number to each line in the product structure. They provide an additional identifier or label for organizing the items in a single-level structure relationship.

You can set the following site preferences to validate find numbers:

Site preference	Description
PS_new_seqno_mode	Determines how new find numbers are allocated when items are inserted into a BOM view or BOM view revision. The default value is **new**. When it is set to this value, every item is given a new find number within the current BOM view, starting with 10 and increasing by 10. This preference has three settings: • **New**: Every item is given a new find number within the current BOM view, starting with **10** and increasing by increments of 10. • **Existing**: If an item with the same identifier already exists in the BOM view, Teamcenter assigns the inserted item the same find number. If not, the item is assigned a new find number according to the default sequence. • **None**: No find number is allocated to items inserted into a BOM view; users can add their own find number later.

Site preference	Description
	Note: After a find number crosses the value 2147483647, for every new line added, the find number starts from 8 and is incremented by 1 instead of 10.
PS_Find_Number_Validation	Determines whether the system validates find numbers. If validation is enabled by setting this preference, an error displays when the find number is zero or is not unique within the parent structure. By default, it is set to **false**. When it is set to this value, the system does not validate the Find Number.
PS_Duplicate_FindNo_Update	Determines whether duplicate find numbers of the same item are updated. By default, it is set to **true**. When it is set to this value, duplicate find numbers of the same item are updated.

Change the start and increment values for a find number

You can configure how *find numbers* are generated for structure elements by changing their starting value and increment value.

To define the starting value to be applied while generating a find number for the first occurrence in a BOM View Revision (BVR), you must set the **TC_BOM_INITIAL_SEQUENCE_NUMBER** preference. The default value of this preference is **10**, and you can specify any positive integer as a valid value.

To define the default increment to be applied while generating a sequence number for a subsequent occurrence under a BVR, you must set the **TC_BOM_SEQUENCE_NUMBER_INCREMENT** preference. The default value of this preference is **10**, and you can specify any positive integer as a valid value.

Define units of measure

Unit of measure is an attribute of the item. In Structure Manager, the BOM engineer specifies the value of the **Quantity** occurrence property in the units of measure for the component item (for example, 1.5 L for an **Oil** item).

The BOM engineer can optionally specify a quantity for a structure line in a user-defined unit of measure. To enable this option, you must set the **Fnd0PSEQtyConversionDSName** and **Fnd0PSEEnableQtyConversionUOM** global constants and create an XML file specifying all UOM conversion rules that are valid at your site.

For more information about these global constants, see the *Global constants addendum* in *Configuring Your Business Data Model in BMIDE*.

15. Set up import and export of structures

Set the properties to uniquely identify an occurrence during a structure import

While a business user is importing a structure from Excel, you can include certain properties as values in the **AWC_Occ_Unique_Identifier** preference to uniquely identify an occurrence that appears multiple times in the structure. By default, this preference contains the **bl_ref_designator**, **bl_occurrence_name**, and **bl_sequence_no** values. Any additional properties that you include in this preference are used to uniquely identify each occurrence of an element.

To know more about preferences in Active Workspace, see *Where can I get a list of preferences?*

Enable the import of remote components of a structure

If you have assembly structures that are replicated at several sites with Multi-Site Collaboration, Teamcenter may prompt users whether to import missing components—that is components that are not yet replicated at your site. If you set the **PSE_prompt_for_remote_import** preference to **on**, Teamcenter prompts if remote items should be imported when you expand the structure. When Structure Manager displays a **Ready** message, all remote components are imported successfully. If this preference is set to **off**, Teamcenter imports missing components without prompting the user.

Configuring structure export from Active Workspace to NX

The administrator must set the following preferences for the site to set up the export to NX feature for Active Workspace:

- Set the **allow_bb_bcz_export_import** preference to allow Briefcase export to complete the export NX data operation. The following example shows the preference details:

```
<preference name="allow_bb_bcz_export_import" type="Logical" array="false"
disabled="false" protectionScope="Site" envEnabled="false">
<preference_description>Allows briefcase export during Export NX Data operation.</
preference_description>
        <context name="Teamcenter">
            <value>true</value>
         </context>
    </preference>
```

- Set the **AWN0NX_NX_UnmanagedSite** preference to define the site to which you want to export the structure. The following example shows the preference details:

```
<preference name="AWN0NX_NX_UnmanagedSite" type="String" array="false"
disabled="false" protectionScope="Site" envEnabled="false">
      <preference_description>Defines the name of unmanaged offline site for Export NX
Data.</preference_description>
      <context name="Teamcenter">
```

```
      <value></value>
    </context>
  </preference>
```

- Set the **AWN0NX_ExportNotificationsCleanupDays** preference to specify the maximum duration for which the notification message and associated Briefcase file are retained after an export. The following example shows the preference to set the duration for 15 days:

```
<preference name="AWN0NX_ExportNotificationsCleanupDays" type="Integer" array="false"
disabled="false" protectionScope="Site" envEnabled="false">
      <preference_description>
        Specifies the maximum age (in days) for a export notification message and
associated briefcase dataset,
        after which the export notification and associated briefcase datasets is
deleted.
        Valid values are 1 through 365. If value is set to more than 365 then datasets
older than 365 days will be deleted.
      </preference_description>
      <context name="Teamcenter">
        <value>15</value>
      </context>
    </preference>
```

- Set the **AWN0NX_export_exclude_file_types** preference to specify the file types to be included in the export when the **Export Associated Files** option is selected during export. This preference is set at an individual user level.

```
<preference name="AWN0NX_export_exclude_file_types" type="String" array="true"
disabled="false" protectionScope="User" envEnabled="false">
      <preference_description>Controls the export of physical file types during Export
NX Data operation.</preference_description>
      <context name="Teamcenter">
        <value>UGMASTER:qaf,tso</value>
        <value>UGPART:qaf,tso</value>
        <value>UGALTREP:qaf,tso</value>
      </context>
    </preference>
```

Create a Microsoft Excel template for exporting product structures

To perform a simple, static export of the product structure and view the formatted results in Microsoft Excel, do the following:

1. Add a custom Excel template, as follows:

    a. Ensure you are logged on as a user with Teamcenter administration privileges.

    b. Create a new item of type **ExcelTemplate** in the Teamcenter administrator's account **Home→Requirements Management Templates→ExcelTemplates** folder.

c. Add the **MSExcelX** dataset underneath the **ExcelTemplateRevision** object.

Teamcenter uses the named reference of this dataset as your template for structure export.

> Note:
>
> Choose appropriate naming conventions for item, dataset, and named reference.
>
> Bypass naming rules if required.

2. To include any custom form properties in the report, create a custom transfer mode and add it to the Excel template. The transfer mode comprises a custom closure rule and a custom property set.

a. In the PLM XML/TC XML Export Import Administration application, create a custom closure rule to export the item revision master form. For example:

Primary object class	Primary object	Secondary object class	Secondary object	Relation type	Related property or object	Action type
Class	*****	*****	**Item Version Master**	**PROPERTY**	**IMAN_master_form_rev**	**PROCESS + TRAVERSE**

b. Create a custom property set that defines the custom form properties you want to export. The property set should contain a line similar to the following example for each custom property you want to export.

Primary object class	Primary object	Relation type	Related property or object	Action type
Class	**Form**	**PROPERTY**	*custom_form_name*	**DO**

3. Before configuring the Excel template, test the custom transfer mode works by performing a PLM XML export.

a. In Structure Manager, select an item revision that includes your custom form properties.

b. Choose **Tools→Export→Object→PLMXML→***your custom transfer mode*.

Teamcenter creates a PLM XML output file.

c. Verify the output file holds the values of the custom properties.

4. Configure the Excel template by adding the necessary columns for the custom properties. logos, and hyperlinks. Refer to the sample files for examples.

5. By default, every level in the Excel template is displayed in a different row in the final result. To merge values for a single revision into one row, you must apply packing to the **ExcelTemplate** item type.

   a. Check out the item.

   b. Select the **apply_packing** value for the **Excel Template Rules** property and modify it as necessary.

   c. Check in the item.

6. Perform the structure export by choosing **Tools→Export→Objects to Excel→Use Excel Template** and selecting your custom Excel template.

16. Create predefined occurrence note types

You can associate occurrence note types with an occurrence in the product structure. Users can specify a value for any note type that is defined for the site.

Some standard note types are supplied to allow synchronization of occurrence attributes with NX. These note types can be displayed but not deleted.

You can define lists of values (LOVs) and default values for occurrence notes.

To create a new occurrence note type:

1. Start the Business Modeler IDE and, in the Extensions view, select the project in which you want to create the new note type. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new note type.

2. Expand the project and the **Options→Note** folders.

3. Right-click the **Note** folder and choose **New Note Type** from the shortcut menu.

   The New Note Type wizard runs.

4. Perform the following steps in the **Note Type** dialog box:

   > Note:
   >
   > The **Project** box defaults to the already selected project.

   a. In the **Name** box, enter the name you want to assign to the new note type.

   b. In the **Description** box, enter a description of the new note type.

   c. Select the **Attach Value List** check box if you want to attach a list of values (LOV) to the note.

   d. If you selected the **Attach Value List** check box, click the **Browse** button to the right of the **LOV** box to locate the list of values to attach to the note. Type an asterisk * in the **Find** dialog box to see all possible selections. Click the **Browse** button to the right of the **Default Value** box to choose the value from the list of values that you want to use for the note type.

   e. Click **Finish**.

      The new occurrence note type displays under the **Note** folder in the **Extensions** view.

5. To save the changes to the data model, chose **File→Save Data Model**.

17. Configure sessions

About configuring sessions

Users can save the filtered and configured structures in *sessions*, which help them return to the product definitions that they are currently working with. You can configure a session so that:

- It **loads** either with static data or with the latest data.
- Users can **share** their sessions with other users.
- One user cannot **overwrite** the changes made by the other user when two users work with the same session.

Set how data must be loaded in a session

Users can save the filtered and configured structures in a session. You can choose whether to load a session with the static data stored in the session or with the latest data. To do this, you edit the **DesignContextLoadRDVContextObjectMode** preference. Set its value to:

- **0** if you want to load the session with the data already stored in it.
- **1** (default) if you want to load the session with the latest data. In this case, the stored data is re-evaluated against the current data, and the latest data is loaded in the session.

Change the default sharing behavior of a session

Users can share a session with other users for collaboration. The default sharing behavior of a session is set in the **Has Class(Fnd0AppSession)** Access Manager rule. This rule is available under **Has Class(POM Object)** > **Has Class(POM_application_object)**.

The **Fnd0AppSession** rule has three values:

- Private (**PrivateAppSessionACL**) — only the owning user can view or modify sessions. Other users cannot view the sessions.
- ReadOnly (**ReadOnlyAppSessionACL**) — only the owning user can modify sessions. Other users can view the sessions.
- ReadWrite (**ReadWriteAppSessionACL**) — all users can view or modify sessions.

To change the default sharing behavior of a session, you must modify these values.

Restrict users from overwriting sessions

A user can overwrite a session that is opened by another user for making updates. As an administrator, you can restrict users from overwriting sessions. To do so, you set the **AWBShowOverwriteForConcurrentSave** preference to **false**.

18. Specify structures as precise or imprecise

In Active Workspace, you can specify if a structure is precise or imprecise by editing the top line of the structure from the **Properties** panel and selecting the **Precise** property. The default value is **Imprecise**.

If you do not see the **Precise** property for an element such as a logical block, you must edit the XRT of that element and add the following property under the **tc_xrt_Properties** section:

```
<property name="awb0IsPrecise">
```

> Note:
>
> To set the default value for newly created product structures as precise, change the value of the **TC_BOM_Precision_Preference** preference to **Precise**. (The default value is **Imprecise**.)

19. Administer revision rules

About revision rules

A *revision rule* sets the criteria for selecting the appropriate revision of elements in a product structure. For example, users can choose whether to load working revisions or release revisions. As a privileged user (or an administrator), you set up revision rules by defining *revision rule entries*. Each rule entry defines a parameter that is used for evaluating which structure configuration must be loaded. The revision rule is evaluated based on the order of precedence of the rule entries. As soon as a revision is successfully configured, the rest of the rule entries are not evaluated. Therefore, you must set the order based on your site requirement.

The different rule entries are:

Revision rule entry	Description	Example

Working	It is used for loading the working item revision that do not have a release status. By default, the latest working revision is selected according to the date it was created.
	It can be made more specific by specifying an owing user and owning group.
		To load elements that John or Project X group are currently working on, you set up the revision rule as:
		`Working( Owning User = John )`
		`Working( Owning Group = Project X)`
		To load elements that belongs to the user currently logged on:
		`Working( Owning User = Current )`
		To load elements that belong to the group to which the user is currently logged on to:
		`Working( Owning Group = Current )`

Status	It is used for loading item revisions that are released with a specific status. It can be made more specific by specifying a specific status, release date, effective date, and effective unit number.	To load the latest revision of elements as per their effective production date: `Has Status (Production, Configured by Effective Date )` To load the most recently released revision of elements that are either in the **Production** or **Pre-Production** state:

Revision rule entry	Description	Example
		`Has Status( Production, Configured by Date Released ) Has Status( Pre-Production, Configured by Date Released)`

Latest	It is used for loading item revisions regardless of whether they are released or not. It can be made more specific by including the creation date, alphanumeric revision ID, numeric revision ID, and a combination of alpha and number revision IDs.	To load the latest revisions of elements based on their creation date:
		`Latest( Creation Date )`
		To load the latest revisions of elements based on numeric revision ID:
		`Latest( Numeric Rev ID)`
		Here, elements with non-numeric revision IDs are not loaded.

Date	It is used for loading item revision with a specific date. You can use this entry only with other entries.	To load the latest revision of elements as on today:
		`Latest, today`
		To load the latest revision of elements that are in **Production** as of **1-Jan-2007**:
		`Has Status( Production, Release Date = 1-Jan-2007)`

Unit Number	It is used in combination with the status rule entry with unit number effectivity. It is used for loading item revisions with a unit number as specified by the user. Typically, a unit number is a property of the end product or a major module of a product. As Teamcenter may manage many units, you typically qualify a unit number entry with an end item entry.	To load the element revisions that are in **Production** based on a specific unit number: `Has Status( Production, Configured by Unit Number )`
End item	It is a product, system, or module with respect to which you configure a structure by effectivity. For example, you can configure the structure of unit number **110** in product **X400**, where **X400** is the end item.	

Revision rule entry	Description	Example
Precise	It is used for loading item revisions in a precise product structure.	
Override	It is used for overriding all item revisions available in an **override folder**. The revision rule will not be evaluated for these item revisions.	

You can modify the order of the rule entries to change the precedence used when evaluating the revision rule. Certain rule entries can also be **grouped** so they are evaluated with equal precedence. In the case of status entries, you can group two or more different statuses with equal precedence. In the case of working entries, you can group different owners with equal precedence.

> Example:
>
> To load recently released revisions of structure elements irrespective of whether they are in the **Production** or **Pre-Productions** state, you group the status rule entry as follows:
>
> ```
> [ Has Status( Production ), Configured by Date Released )
> Has Status( Pre-Production, Configured by Date Released )]
> ```

You can also group entries according to **item type** and **occurrence type**. Certain entries cannot be grouped together. The following are not valid combinations for grouping:

- ```
  Latest
  Has Status, Configured By Release Date
  ```

- ```
  Has Status( TCM Released ), Configured by EffectiveDate
  Has Status( Pending ), Configured by Release Date
  ```

Revision rule entries: Scenarios

Working entry: Scenario

A *working entry* is used for selecting working item revisions that do no have a release status. By default, the latest working revision is selected according to the date it was created. It can be made more specific by specifying an owing user and owning group.

> Tip:
>
> There may be more than one working entry within a revision rule. For example, a rule may configure the current user's working revision and, if none is found, configure the current group's

working revision instead. If a user changes group, it is necessary to reapply the revision rule to configure the appropriate revisions for the new group.

However, in many circumstances, it is good practice to limit the number of working revisions, typically to a single revision. Your Teamcenter administrator can do this in Business Modeler IDE.

Example of a Working entry

```
Working( Owning User = John )
Working( Owning Group = Project X )
```

Assume you configure the following two items with this rule:

```
Part1/A : Status = Production
Part1/B : Working, Owning User = John, Owning Group = Project Y
Part1/C : Working, Owning User = Jane, Owning Group = Project X
Part2/A : Status = Production
Part2/B : Working, Owning User = Jane, Owning Group = Project X
```

Teamcenter configures Revision B of Part 1, because it is owned by John. The owning groups are not relevant.

Teamcenter configures Revision B of Part 2, because it is owned by Project X. There is no revision owned by John.

Note:

If John or Project X owned more than one revision, Teamcenter would configure the latest created revision of the matching revisions.

Example of a Working entry with current user/group

```
Working( Owning User = Current )
```

Assume you configure the following item with this rule:

```
Part1/A : Status = Production
Part1/B : Working, Owning User = John, Owning Group = Project Y
Part1/C : Working, Owning User = Jane, Owning Group = Project X
Part1/D : Working, Owning User = Jane, Owning group = Project Y
```

This rule configures a revision that depends on the identity of the *user* logged into Teamcenter.

- If Jane is logged in, Teamcenter configures Revision D.
- If John is logged in, Teamcenter configures Revision B.

Assume you use the following rule to configure the same item:

```
Working( Owning Group = Current )
```

The revision configured by this rule is dependent on which group the user is logged into Teamcenter.

This rule configures a revision that depends on the *group* in which the user logged into Teamcenter.

- If the user is logged into Group Project X, Teamcenter configures Revision C.
- If the user is logged into Group Project Y, Teamcenter configures Revision D.

Status entry: Scenario

A status entry is used for selecting item revisions that are released with a particular status. The following settings are available for status entries:

Any release status	Teamcenter configures the latest item revision with a released status, regardless of the actual status.
Selected status	Teamcenter configures the latest item revision with a selected status type. This setting allows you to configure a structure that contains only item revisions with a specified status.
Released date	Teamcenter selects the latest item revision according to the date the revision was released (that is, the date the particular status was added).
Effective date	Teamcenter selects the latest item revision according to effectivity dates defined on the release status.
Effective unit number	Teamcenter selects the latest item revision according to unit numbers defined on the release status.

Example of status hierarchy in a revision rule

The following example shows how to use status hierarchy in a revision rule:

```
Has Status( Production, Configured by Date Released )
Has Status ( Pre-Production, Configured by Date Released )
```

Assume you configure the following item with this rule (dates are release dates):

```
Part1/A : Status = Pre-Production [1-Apr-2007]
Part1/B : Status = Pre-Production [1-Jun-2007]
Part1/C : Status = Production [1-Aug-2007]
Part1/D : Working
Part2/A : Status = Pre-Production [1-May-2007]
```

```
Part2/B : Status = Production [1-Jul-2007]
Part2/C : Status = Pre-Production [1-Sep-2007]
Part3/A : Status = Pre-Production [1-Aug-2007]
Part3/B : Working
```

Teamcenter configures Revision C of Part 1 because it is the most recently released revision with **Production** status.

It configures Revision B of Part 2 because it is also the most recently released revision with **Production** status. The later preproduction revision is not configured.

It configures Revision A of Part 3. There is no revision with **Production** status, so it configures the latest **Pre-Production revision**.

> Note:
>
> The rule in this example creates a status hierarchy. If possible, the rules configures a **Production** release, but if one is not available, it configures a **Pre-Production** release.

Example of effective date configuration in a revision rule

The following example shows effective date configuration in a revision rule:

```
Has Status( Production, Configured by Effective Date )
```

Assume you configure the following item with this rule (dates are effective date ranges):

```
Part1/A : Status = Production [1-Apr-2007 to ... ]
Part1/B : Status = Production [1-Aug-2007 to ... ]
Part1/C : Status = Production [1-Nov-2008 to ... ]
```

The release status has both start date and end date attributes, but typically the end date attribute is not populated. The system selects the latest effective revision of the appropriate status type with a start date before the date specified in the date revision rule entry.

If no date is set in the rule, it defaults to today, and the revision with the latest start date is selected.

In the previous example, if today's date is sometime in 2007, Teamcenter configures revision B, because it has **Production** status and also has the later effective start date of the two revisions effective in 2007.

If no start date or end date are defined on the release status of an item revision, it is assumed to be not effective; that is,effectivity criteria must be defined for an item revision to become effective,

Example of effective unit number configuration in a revision rule

The following example shows effective unit number configuration in a revision rule:

```
Has Status( Production, Configured by Unit Number )
```

Assume you configure the following item with this rule:

```
Part1/A : Status = Production [3 to 10]
Part1/B : Status = Production [11 to 15]
Part1/C : Status = Production [16 to UP]
```

- If the unit number is set to 3, Teamcenter configures revision A. Revision A falls within the range 3 to 10, but neither of the other ranges.
- If the unit number is set to 11, Teamcenter configures revision B. Revision B falls within the range of 11 to 15 upward, but neither of the other ranges.
- If the unit number is set to 16, Teamcenter configures revision C. Revision C falls within the range of 16 upward, but neither of the other ranges.
- If the unit number is set to 2, no revision of Part1 is configured. Unit number 2 is outside the effectivity ranges of all three parts.

The release status has both start unit and end unit attributes, and typically both attributes are populated. The system selects the revision of the appropriate status type with the start unit closest to, and before, the unit number specified in the unit number revision rule entry. If no unit number is set in the rule, there is no practical default and no revisions are selected

If no start unit or end unit are defined on the release status of an item revision, it is assumed to be not effective. That is, the effectivity criteria must be defined on an item revision to make it effective.

Split unit number ranges can be defined by attaching multiple release statuses to the status type.

If multiple revisions are eligible for configuration, the one with the highest start unit number is selected.

Latest entry: Scenario

A *latest* entry is used for selecting the latest item revisions regardless of whether they are released or not. For this entry, Teamcenter does not differentiate between working revisions and revisions with status. The following settings are available for **Latest** entries:

Creation Date	Selects the latest item revision by the date the revision was created.
Alphanumeric Rev ID	Selects the latest item revision by revision ID in alphanumeric (individual characters) order.

Numeric Rev ID	Selects the latest item revision by revision ID in numeric order. Revisions with non-numeric IDs are not configured.
Alpha+Number Rev ID	Selects the latest item revision by revision ID, sorting letter portions as an alphabetic group, and sorting digit portions as a number.

For example:

Setting	Rev ID 1	Rev ID 2	Rev ID 3	Latest	Notes
Alphanumeric Rev ID	aa	b		b	Letters sorted alphabetically (individual letters). ANSI value of b is > a.
Alpha + Number Rev ID	aa	b		aa	Letters grouped and sorted numerically. ANSI value of aa is > b.
Alphanumeric Rev ID	21	3		3	Digits sorted as alphabetic (individual digits). ANSI value of 3 is > 2.
Alpha + Number Rev ID	21	3		21	Digits grouped and sorted as number. 21 is > 3.
Alphanumeric Rev ID	a5	b3	b23	b3	ANSI value of b > a, so on first character comparison, b3 and b23 are selected. Continuing to the second character, ANSI value of 3 > 2.
Alpha + Number Rev ID	a5	b3	b23	b23	ANSI value of b > a, so on first character comparison b3 and b23 are selected. Continuing, numerical digits are grouped and compared by number value.

Setting	Rev ID 1	Rev ID 2	Rev ID 3	Latest	Notes
					23 > 3.
Alphanumeric Rev ID	aa4	bc123	bc22	bc22	ANSI value of b > a, so on first character comparison bc123 and bc22 are selected. Continuing, as the first two characters are the same it goes for third character where ANSI value of 2 > 1.
Alpha + Number Rev ID	aa4	bc123	bc22	bc123	ANSI value of bc > aa, so on first comparison bc123 and bc22 are selected. Continuing, numerical digits are grouped and compared by number value. 123 > 22.

Example of a revision rule with latest by creation date entry

The following example shows a revision rule that configures with the latest by creation date:

```
Latest( Creation Date )
```

If you configure the following item with this rule (dates are creation dates):

```
Part1/A : Status = Production [1-Apr-2006]
Part1/B : Working [1-Jun-2006]
```

Teamcenter configures Revision B because it was created later than revision A. It is not relevant if the revision is working or has status.

Date entry: Scenario

A date entry is used for specifying a date to configure the structure against. You can only use this entry type with other entries. These types of entry find the latest entry before the specified date:

- Status entry – released date or effective date
- Latest entry – creation date

You can set the date in a date entry to **Today**, and Teamcenter evaluates the configuration criteria against the current date and time.

You cannot configure working revisions against a date in the past. Teamcenter does not maintain information about the revisions that were in a working state at a particular time in the past.

> Note:
>
> If no date entry is present in a revision rule, Teamcenter evaluates the date by default to today's date.

Grouped entries: Scenario

You can group status entries and working entries for equal precedence. In the case of status entries, you can group two or more different statuses with equal precedence.

Example of revision rule with grouped entries

```
[ Has Status( Production, Configured by Date Released )
Has Status( Pre-Production, Configured by Date Released ) ]
```

Assume you configure the following three items with this rule (dates are release dates)

```
Part1/A : Status = Pre-Production [1-Apr-2007]
Part1/B : Status = Production [1-Jun-2007]
Part1/C : Status = Production [1-Aug-2007]
Part1/D : Working
Part2/A : Status = Pre-Production [1-May-2007]
Part2/B : Status = Production [1-Jul-2007]
Part2/C : Status = Pre-Production [1-Sep-2007}
Part3/A : Status = Pre-Production [1-Aug-2007]
Part3/B : Working
```

Teamcenter configures Revision C of Part 1, because it is the latest released revision with either **Production** or **Pre-Production** status.

Teamcenter configures Revision C of Part 2 for the same reason.

Teamcenter configures Revision A of Part 3, again for the same reason.

Precise entry: Scenario

If an assembly is precise and the revision rule contains a **precise** entry, the assembly is configured according to that entry. However, if the assembly is precise and the revision rule does *not* contain a **precise** entry, the assembly is treated as imprecise.

For example:

- You have an item called Part1 with two revisions, Part1/A is **Working** and Part1/B is also **Working**.

  A precise assembly **Asm** is created containing Part1/B. That is:

  ```
  Asm
   |--Part1/B
  ```

  If User2 is denied read access to Part1/B and opens **Asm** in Structure Manager with an **AnyStatus;Working** revision rule, the system performs the following evaluation:

  1. The **AnyStatus** entry does not configure any revisions.
  2. The **Working** entry configures in Part1/B but access rules make this revision **<UNREADABLE>**.
  3. Regardless of the setting of the **PSE_check_read_access_for_Rev_Rule_entry** preference, evaluation continues. The next revision is Part1/A, and access rules do not deny access to it, so the final result is Part1/A.

  If User2 then changes the revision rule to **Latest by Alpha Revision Order**, the system performs the following evaluation:

  1. The **Latest by Alpha Revision** entry configures in Part1/B but access rules make this revision **<UNREADABLE>**.
  2. The system continues evaluation only if the **PSE_check_read_access_for_Rev_Rule_entry** preference is **TRUE**. The next revision is Part1/A, and access rules do not deny access to it, so the final result is Part1/A.

- You have an item called Part1 with three revisions, Part1/A is released with a **TcReleased** status, Part1/B is released with a **TcBaseline** status, and Part1/C is released with a **TcReleased** status.

  A precise assembly **Asm** is created with Part1/C. That is:

  ```
  Asm
   |--Part1/C
  ```

  If User2 is denied read access for the **TcReleased** status and opens **Asm** in Structure Manager with a **AnyStatus;Working** revision rule, the system performs the following evaluation:

1. The **AnyStatus** entry configures in Part1/C but access rules make this revision **<UNREADABLE>**.

2. The system continues evaluation only if the **PSE_check_read_access_for_Rev_Rule_entry** preference is **TRUE**. The next revision with status is Part1/B, and access rules do not deny access, so the final result is Part1/B.

- You have an item called Part1 with two revisions, Part1/A is released with a **TcReleased** status and Part1/B is working.

  A precise assembly **Asm** is created with Part1/A. That is:

  ```
  Asm
   |--Part1/A
  ```

  If User2 is denied read access for status **TcReleased** and opens **Asm** in Structure Manager with a **AnyStatus;Working** revision rule, the system performs the following evaluation:

  1. The **AnyStatus** entry configures in Part1/A but access rules make this revision **<UNREADABLE>**.

  2. The system continues evaluation only if the **PSE_check_read_access_for_Rev_Rule_entry** preference is **TRUE**. Because no other revisions have status, the next rule entry of **Working** configures in Part1/B. Access rules do not deny access, so the final result is Part1/B.

  If User2 changes the revision rule to **Latest Working**, the system performs the following evaluation:

  1. The **LatestWorking** rule contains a **Precise** entry, that configures the revision saved with the assembly, namely, Part1/A.

  2. Because access rules make Part1/A unreadable for User2, the final result is **<UNREADABLE>**.

     The **PSE_check_read_access_for_Rev_Rule_entry** preference does not affect the result because the structure and the revision rule are precise.

Revision rules for incremental changes: Scenario

Review the example of how the **Pre-Released** status can be applied to the item revisions and the **IC in Process** status to the incremental change revisions.

The revision rule needed for that example is as follows:

Has Status = IC in Process, Configured by Unit:
Has Status = Pre-Released, Configured by Unit:
Has Status=Released, Configured by Unit
Has Status = Released, Configured by Release Date
Has Status = Pre-Released, Configured by Release Date
Working

The purpose of each entry in this revision rule is as follows:

Has Status = IC in Process, Configured by Unit:
Has Status = Pre-Released, Configured by Unit:

Allows Teamcenter to configure both the incremental change and the structure revisions by unit number. Having separate statuses allows the application of different access controls to the item revision and BOM view revision and to the incremental change.

Has Status=Released, Configured by Unit

Ensures the upper levels of the structure are configured correctly by standard revision configuration, as well as any finally released revisions at lower levels.

Has Status = Released, Configured by Release Date
Has Status = Pre-Released, Configured by Release Date
Working

Configure item revisions that do not have unit effectivity applied or are unreleased, that is, **working**.

Group revision rule entries in the rich client

Group revision rule entries by item type

You can modify the order of precedence of revision rule entries by grouping revision rules by item type. Teamcenter evaluates any revision rule under the item type group for configuration of item revision only if the item type of the revision rule group and underlying item revision matches. By using this grouping mechanism in a product structure that has item revisions of different item types, you can create revision rules that selectively apply the revision rule entries according to the given item type.

For example, you may use an item to manage parts and equipment, but need to distinguish between a production part and the equipment used to manufacture the part. Each of these item types may have the same status but can be configured in a different way, for example, by unit or date released. To achieve this, you can define a revision rule with a clause that restricts some of the entries to certain item types. To do so, create a **Has Item Type** entry in a revision rule in the following format:

```
Has Item Type (<item type 1, item type 2, etc) {

<Entry 1>

<Entry 2>
etc.}
```

Example:

```
Has Item Type (Part) }
 Has status (Approved, Configured with Released Date) }
 Has status (Production, Configured with Unit Number)
```

Here the **Has status (Approved, Configured with Released Date)** revision rule entry configures the product structure only if the item type is **Part**. For all other item types in the product structure, Teamcenter applies the **Has status (Production, Configured with Unit Number)** revision rule entry. If neither of these entries applies, the part is not configured by this revision rule.

Grouping entries for multiple item types

Each **Has Item Type** entry may have one or more item type arguments, but cannot be left blank, that is, you cannot create a revision rule entry for **Has Item Type (Any)**. To include more than one item type argument, use the syntax in the following example:

```
Has Item Type ( Prototype, Virtual Build ) {
...}
```

Grouping combinations of item type entries

You can only use one level of grouping and may not create nesting inside an existing group entry, as shown in the following examples:

- Single revision rule entry grouped for item type:

```
Has Item Type ( Prototype ) {
  entry 1 }
entry2
```

- Multiple revision rule entries grouped for item type without equal precedence:

```
Has Item Type ( Prototype ) {
  entry 1
  entry 2 }
entry3
```

- Multiple revision rule entries grouped for item type with and without equal precedence:

```
Has Item Type ( Part ) {
  [Status (Released, Configured Using Release Date) }
Has Item Type ( Module, Incremental Change ) {
  [Status (Released, Configured Using Unit No) }
Working
```

This example shows how you can use the same status type in both situations, but configured differently—in this case, by date for parts, but by unit number and incremental change for modules.

You can only group **Working** and **Status** entries with equal precedence.

Group revision rule entries by occurrence type

You, as a configuration management analyst, can define revision rules to treat different occurrence types with variations of the configuration logic. In some cases, the existing grouping mechanisms based on the item type are not sufficient because a part appears in some assemblies only for reference purposes. In such cases, you can use a **Working revision** rule to author the BOM significant parts in an assembly. However, the system may reference the content, using a **Released revision** rule. For example, you create a **Has Occurrence** for the **MEAssign** occurrence type. Then you create a **Working()** revision rule to configure the product structure only if the occurrence type is **MEAssign**. For all other occurrence types in the product structure, Teamcenter applies the **Has Status(Any Released Status, Configured Using Released Date)** rule.

You can modify the order of precedence of revision rule entries by grouping revision rules by occurrence type. For the configuration of an item revision, Teamcenter evaluates a revision rule within the occurrence type group only if there is a match between the occurrence type of the revision rule group and the underlying item revision on the line. By using this grouping mechanism with item revisions of different occurrence types, you can create revision rules to selectively apply the revision rule entries according to a given occurrence type.

You cannot create item type groups within occurrence type groups.

You create the **Has Occurrence** type in the following format:

```
Has Occurrence Type(Occurrence type1, Occurrence type 2, etc) {
<Revision Rule Entry 1>
<Revision Rule Entry 2>
……
etc. }
```

Example:

```
Has Occurrence Type(MEAssign) {
Working()  }
Has Status(Any Release Status, Configured Using Released Date)
```

In this example, the **Working()** revision rule entry configures the product structure only if the occurrence type is **MEAssign**. For all other occurrence types in the product structure, Teamcenter applies the **Has Status(Any Released Status, Configured Using Released Date)** rule.

Group combinations of occurrence type entries

You can only use one level of grouping as shown in the following examples.

Do not create nested entries within an existing group entry.

- Single revision rule entry grouped for occurrence type:

```
Has Occurrence Type ( OccurrenceType1 ) {
  entry 1 }
entry2
```

- Multiple revision rule entries grouped for occurrence type without equal precedence:

```
Has Occurrence Type ( OccurrenceType2 ) {
  entry 1
  entry 2 }
entry3
```

- Multiple revision rule entries grouped for occurrence type with and without equal precedence:

```
Has Occurrence Type ( MEAssign ) {
  [Status (Released, Configured Using Release Date) }
Has Occurrence Type ( MeAssemble ) {
  [Status (Released, Configured Using Unit No) }
Working
```

This example shows how you can use the same status type for both situations by configuring it differently. In this case, you use the by date for **MEAssign** and the by unit number for **MeAssemble**.

Group existing revision rule entries by item type

1. Choose **Tools→Revision Rule→Create/Edit**.

   Teamcenter displays the **Revision Rule** dialog box.

2. In the dialog box, select one or more revision rule entries and click the **Group** button.

   Teamcenter displays the **Group Entries** dialog box.

3. In the dialog box, click the **Group Entries by Item Types** button or the **Group Entries by Equal Precedence** button, then select the item types you want to group in the **Available Item Types** list. You can transfer the selected item types to the **Configure By→Has Item Type** box in the dialog box or remove them by clicking the **+** or **–** buttons respectively. Click **OK** when you have grouped the entries.

   Teamcenter updates the selected item types in the **Configure By→Has Item Type** box of the dialog box, with a partially created **Has Item Type** revision rule entry.

> Note:
>
> You can only group **Working** and **Status** entries with equal precedence.

Ungroup combinations of item type entries

1. Choose **Tools→Revision Rule→Create/Edit**.

   Teamcenter displays the **Revision Rule** dialog box.

2. In the dialog box, select one or more **Has Item Type** revision rule entries that are grouped by item type, and click the **Ungroup** button.

   Teamcenter removes the selected entries and updates the dialog box.

Create a revision rule

How to create revision rules

If you are a privileged user, you can use two dialog boxes to create or edit a revision rule:

- The **Modify Current Rule** dialog box to create temporary revision rules.
- The **Create/Edit Rules** dialog box to create persistent revision rules.

Both dialog boxes contain the Revision Rule Editor, which comprises two main panes.

- The upper pane lists all the entries in the rule, with buttons you can use to manipulate the entries.
- The lower pane allows you to create or edit an entry, and add it into the rule.

Set an override folder

To add an override folder into the currently active revision rule, choose **Tools→Revision Rule→Set Override Folder**. This command is available to unprivileged users, but is only active for revision rules that contain a special empty override entry (no folder specified). Thus privileged users who define the revision rules should place an empty override entry into a revision rule if they want to allow unprivileged users to use their own override folders with the rule.

The **Set Override Folder** dialog box allows you to set an override folder and also to see the folder that is currently set. The dialog box has the following boxes:

- A **Name** box into which you can enter a case sensitive search string to find the required override folder. Click the **Find**  button next to this box to perform the search.

- The search results box. Double-click the folder you want to set.
- A **Folder** box that displays the folder currently used as the override folder in the revision rule.

To use the existing override folder, simply click **OK** or **Cancel**.

Note:

- The first time you make a change (such as setting an override) to a saved revision rule, Teamcenter creates a modified copy of the rule and applies it to the window. This is the rule to which you then make the override change.
- When you make a change to a folder (for example, adding an item revision to it when Teamcenter Integration for NX is active) and open an assembly configured by a rule that uses the override folder, Teamcenter does not automatically reevaluate the revision rule. Consequently, it does not take into account the change to the override folder. This rule is reevaluated only if you choose **File→Options→Load Options** in Teamcenter Integration for NX. Click **OK** before opening the part file.

Create a rule entry

1. In the Revision Rule Editor, select the type of entry you want to create. By default, the combination box shows **Working**. Teamcenter updates the entry boxes appropriately.
2. Enter values into the boxes and click one of the following buttons:
   - Click **Append** to add your entry at the end of the list of rule entries.
   - Click **Insert** to add your entry above the selected entry.
   - Click **Replace** to replace the selected entry with your entry.

Create a revision rule for nested effectivity

When you create a revision rule for a nested effectivity scheme, ensure you select the **Nested Effectivity** check box. This action adds a **Nested Effectivity** entry to the revision rule and ensures that Teamcenter modifies the revision rule according to the mapping on any configuration items it encounters. If you do not check this check box, Teamcenter ignores any configuration items. The position of the **Nested Effectivity** entry in the rule is not significant and never changes.

The following example structure has date effectivity and an end item defined on the item revision of each part:

Item	Status, effectivity	Note
NE-A1000/A (CI) – Product X		
NE-A2000/A (CI)	Released, EI: NE-A1000, 15 July-UP	Rev Effect wrt NE-A1000 – Product X
NE-A100/A	Released, EI: NE-A2000, 20 July-UP	
NE-A200/A	Released, EI: NE-A2000, 20 July-UP	
NE-PP30/A	Released, EI: NE-A2000, 20 July-UP	
NE-PP10/A	Released, EI: NE-A2000, 20 July-UP	
NE-PP20/A	Released, EI: NE-A2000, 20 July-UP	
NE-A3100/A (CI)	Released, EI: NE-A1000, 16 July-UP	Rev Effect wrt NE-A1000 – Product X
NE-A400/A	Released, EI: NE-A3100, 23 July-UP	
NE-PP70/A	Released, EI: NE-A3100, 25 July-UP	
NE-PP60/A	Released, EI: NE-A3100, 25 July-UP	
NE-A5000/A (CI) – Product Y		
NE-A2000/A	Released, EI: NE-A5000, 15 July-UP	Rev Effect wrt NE-A5000 – Product Y

The resulting configuration item effectivity mappings are:

- Mapping on NE-A2000 (CI):

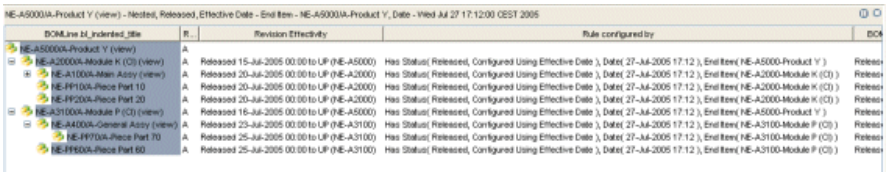
  NE-A1000 (CI) – Product X : NE-A2000 (CI)
  NE-A5000 (CI) – Product Y : NE-A2000 (CI)

- Mapping on NE-A3100 (CI):

  NE-A2000 (CI) – Product Y : NE-A3100 (CI)

If you apply the following revision rule, the structure is configured as shown in the following figure:

Status = Released, Configured by Effective Date
Working
Nested Effectivity
EI = NE-A5000
Date = 27 July 2006



Nested effectivity 1

Conversely, if you apply the following revision rule with no nested effectivity entry, the structure is configured as shown in the following figure:

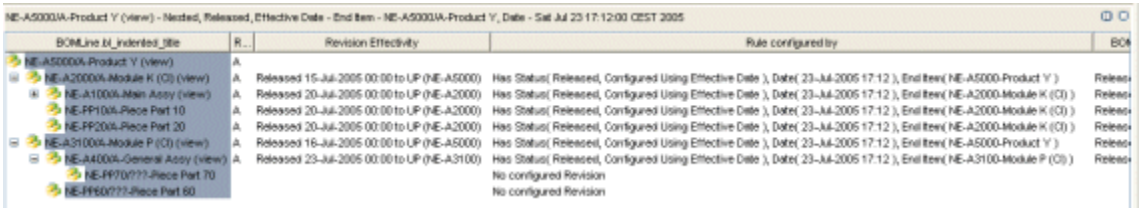
Status = Released, Configured by Effective Date
Working
EI = NE-A5000
Date = 16 July 2005

Nested effectivity 2

Teamcenter ignores the mappings, so the end item does not switch from A5000 to NE-A2000/NE-A3100 at the configuration items. Consequently, none of the components below the configuration items are configured, as the end item on the effectivity is not NE-A5000.

If you apply the following revision rule, Teamcenter applies effectivity mapping due to the **Nested Effectivity** entry and the structure is configured as shown in the following figure. However, some of the components are not effective on 23 July and have no configured revision; Teamcenter marks these components as **???**.

Status = Released, Configured by Effective Date
Working
Nested Effectivity
EI = NE-A5000
Date = 23 July 2005



Nested effectivity 3

Create and group revision rule entries by equal precedence

1. Choose **Tools→Revision Rules→Create/Edit**.

   Teamcenter displays the **Revision Rules** dialog box.

2. In the dialog box, click the **Create/Edit** button.

   Teamcenter displays the **Revision Rule** dialog box with any existing revision rule entries.

3. In the dialog box, define and select the entries of the same type you want to group and click the **Group** button.

   Teamcenter displays the **Group Entries** dialog box.

4. In the dialog box, click the **Group Entries by Equal Precedence** button.

   Click **OK** and Teamcenter updates the **Revision Rule** dialog box with the newly grouped revision rule entries.

   > Note:
   >
   > When grouping by equal precedence, you should only group **Working** entries with other **Working** entries and **Status** entries with other **Status** entries. Do not group other combinations of entries.

5. In the **Revision Rule** dialog box, complete the definition of the revision rule and click **OK**.

> Note:
>
> To remove an equal precedence entry from a revision rule, you must highlight *all* the lines of the entry in the dialog box.
>
> You can also group revision rule entries using engineering change types.

Create and group Has Item Type revision rule entries

1. Choose **Tools→Revision Rules→Create/Edit**.

   Teamcenter displays the **Revision Rules** dialog box.

2. In the dialog box, click the **Create/Edit** button.

   Teamcenter displays the **Create New Revision Rule** dialog box with any existing revision rule entries.

3. In the dialog box, define and select the entries you want to group and click the **Group** button.

   Teamcenter displays the **Group Entries** dialog box.

4. In the dialog box, click the **Group Entries by Item Types** button, and then select the item types you want to group in the **Available Item Types** list. You can transfer the selected item types to the **Configure By→Has Item Type** box in the dialog box or remove them by clicking the **+** or **–** buttons, respectively.

   Click **OK** to update the selected item types in the **Configure By→Has Item Type** box of the dialog box, with a partially created **Has Item Type** revision rule entry. It also updates the **Revision Rule** dialog box with the newly grouped revision rule entries.

5. In the **New Revision Rule** dialog box, complete the definition of the revision rule and click **OK**.

Modify a rule entry

1. Select an entry in the list of entries.

2. Click **Copy**. Teamcenter copies the entry into the editing area and displays the appropriate entry type and boxes.

3. Edit the values of the entry and click **Replace** with the original entry still selected. Teamcenter replaces the old entry with the new entry.

Set dates, units, and end items

To set the date, unit number, and/or end item of the currently active revision rule, choose **Tools→Revision Rule→Set Date/Unit/End Item**.

Teamcenter displays the **Set Date/Unit/End Item** dialog box into which you can enter a date, a unit number, and an end item for the current revision rule.

The first time you make a change to a saved revision rule (including setting a date, unit number, or end item), Teamcenter creates a modified copy of the rule and applies it to the window. This is the rule to which you make the unit number change.

If any date, unit number, or end item has been explicitly set in the current rule, you cannot change the value in this dialog box and the relevant boxes are grayed out.

> Tip:
>
> You can add a button to the toolbar to initiate the **Tools→Revision Rule→Set Date/Unit/End Item** command, by right-clicking the toolbar, choosing **Customize**, and selecting the required button.

If you specify an end item identifier that is shared by multiple objects, Teamcenter displays the **Select Unique Item** dialog box, allowing you to select the object you require.

You can create or modify a revision rule in applications other than Structure Manager, for example, in 4G Designer. Structure Manager does display such revision rules and their entries, and you can use them to configure the structure based on the displayed entries. However, a product structure in Structure Manager and a collaborative design in 4G Designer may have different data conditions. If so, the configurations in the applications are different, even if you use the same revision rule.

Delete a rule entry

1. Select an entry in the list of entries.

2. Click **Remove** and Teamcenter removes it from the rule.

Edit the entries within a rule

You can use the arrow buttons to move entries up or down within the rule.

You can use the **Group/Ungroup** command to add entries to a group or remove them from a group.

Modify the current revision rule

Users can make modifications to the window's current revision rule. This does not affect the saved revision rule, but instead, Teamcenter creates a copy of the rule, modifies it as specified, and applies it to the window. To modify the current rule, choose **Tools→Revision Rule→Modify Current**; Teamcenter displays the **Modify Current Rule** dialog box.

Use the **Modify Current Rule** dialog box to modify the current rule with the Revision Rule Editor. The editor contains a copy of the saved revision rule that is applied to the product structure window. The name of the copied rule is the name of the original rule with **(Modified)** appended. You cannot edit the **Name** box, but you can change the **Description** box.

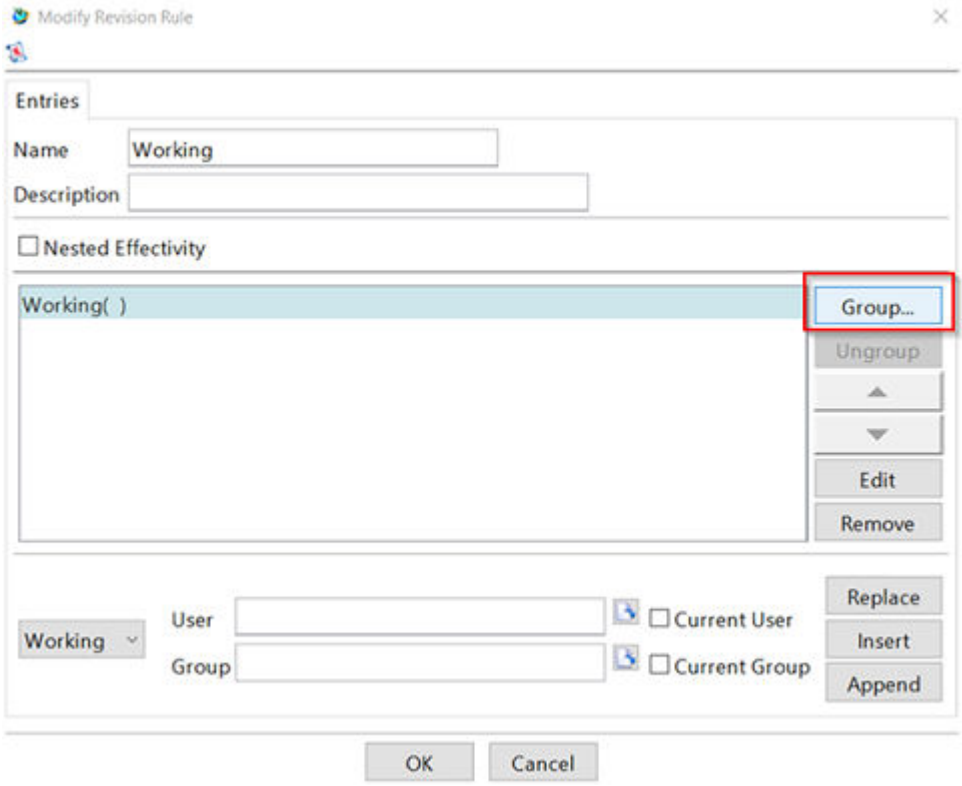
After you edit the revision rule, apply your changes by clicking **OK** or **Apply**.

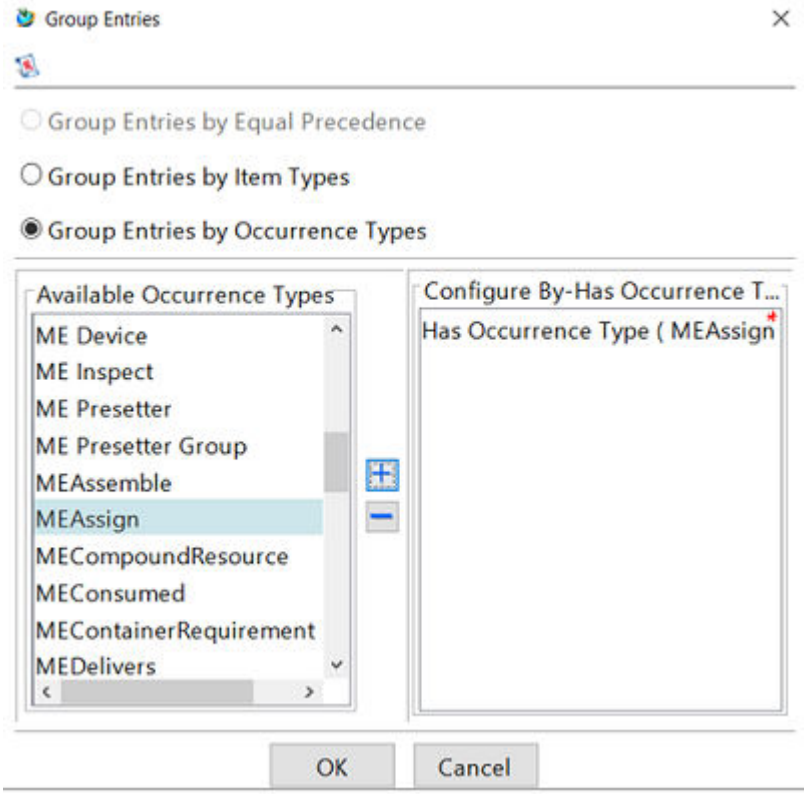
If you have write access to the original rule, you can save your changes back to the original rule by clicking **Save**.

Modify the occurrence types in existing revision rule entries

You can modify the occurrence types in existing revision rule entries grouped by the occurrence type.

1. Choose **Tools→Revision Rules→Create/Edit**.





2. In the dialog box, select one or more **Has Occurrence Type** revision rule and click the **Edit** button or double-click a single line.

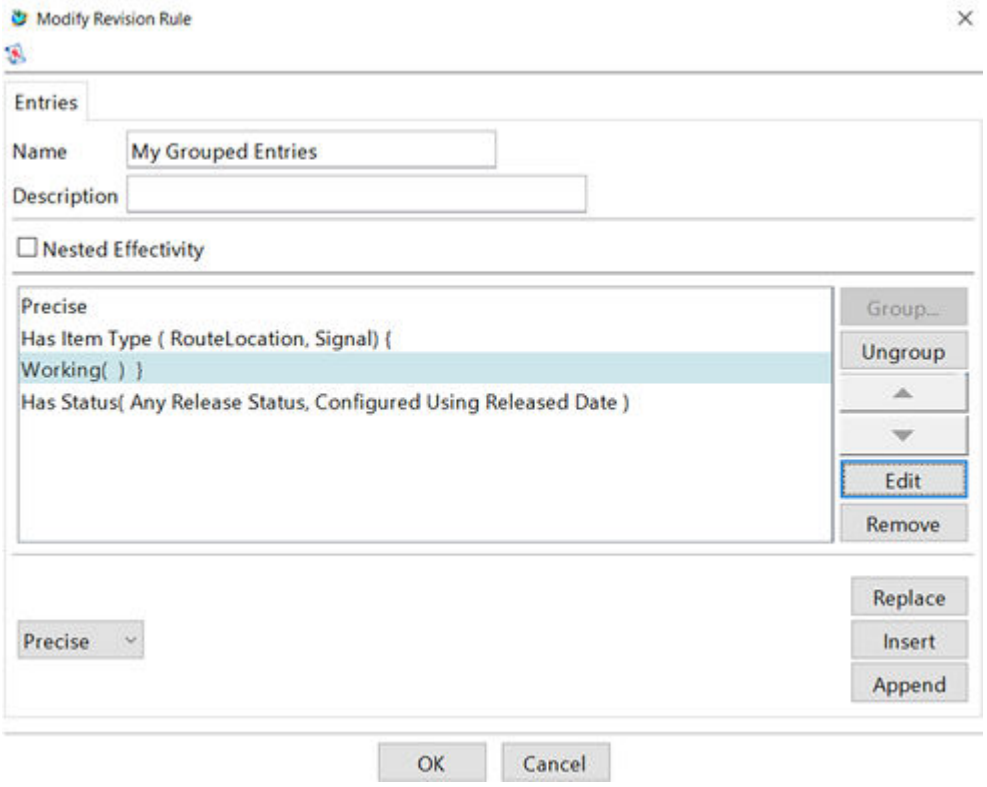
3. To add an occurrence type to the **Configure By-Has Item Type** list, select the occurrence type from the **Available Occurrence Types** list and click the **+** button.

4. To update the revision rule, click **OK**.

Modify the item types in existing revision rule entries grouped by item type

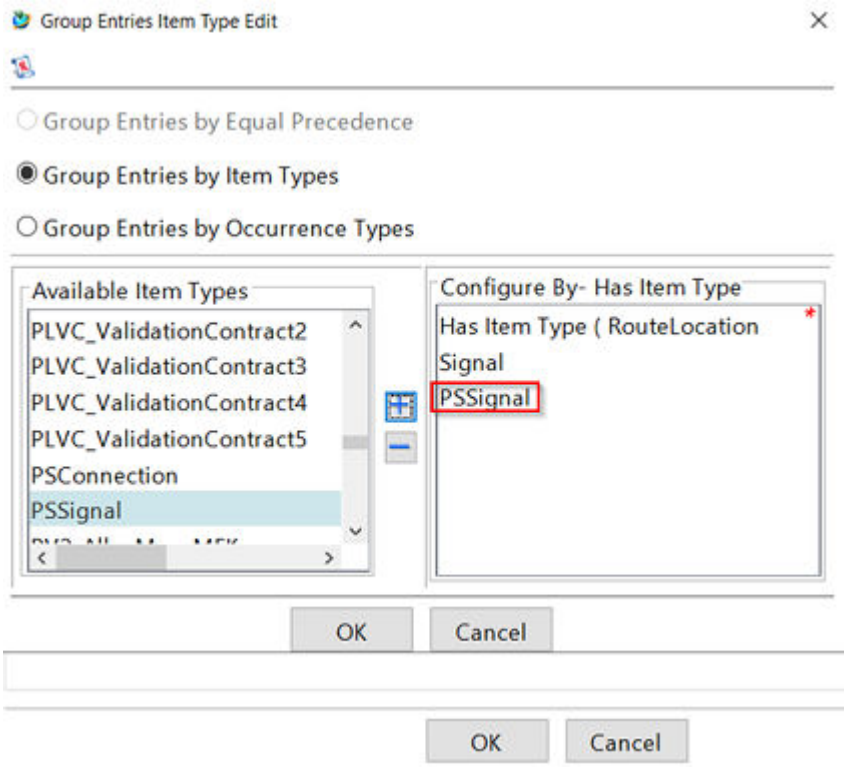
1. Choose **Tools→Revision Rules→Create/Edit**.

   Teamcenter displays the **Modify Revision Rules** dialog box, as shown in the following example.



2. In the dialog box, select one or more **Has Item Type** revision rule entries and click the **Edit** button or double-click a single line.

   Teamcenter displays the **Group Entries Item Type Edit** dialog box, similar to the following example.



3. The item types that are currently grouped in the selected revision rule entry are displayed in the **Configure By-Has Item Type** list. To add a selected item type from the **Available Item Types** list, click the **+** button. To remove a item type, select it in the **Configure By-Has Item Type** list and click the **–** button. When the list of item types is shown correctly, click **OK** to update the revision rule.

> Note:
>
> You must check at least one item type; otherwise, Teamcenter displays an error message.

Create a revision rule with an override clause

You can create a revision rule with an override clause that engineers can use this revision rule for various tasks, such as marking the structure as precise, checking temporary changes, performing what-if analyses, and analyzing changes to the structures based on business requirements.

Procedure

1. Create a folder that can be used by the users as an override folder.

To create a folder	Do this
In Active Workspace	a. In the **EXPLORER** tile, navigate to a location where you want to create a folder and click **More Commands** ▫▫▫ > **New** ✳ > **Add** ⊕.

To create a folder	Do this
	b. On the **Add** panel, select **Folder** type. c. Enter a name and description for the folder. d. Click **Add**.
In the rich client	a. In My Teamcenter, click **File→New →Folder**. b. In the **New Folder** dialog box, select **Folder** and click **Next**. c. Enter a name and description for the folder. d. Click **Finish** and **Close** to close the New Folder dialog box.

2. In Active Workspace, set the **AWC_display_configured_revs_for_pwa** preference to **false**.

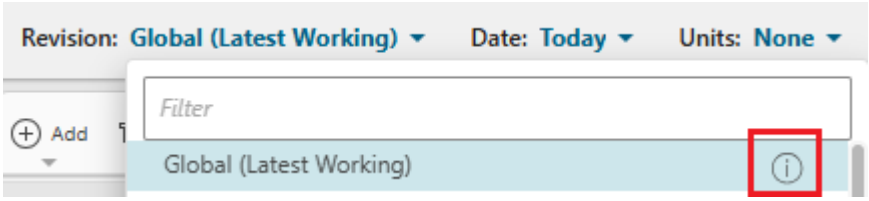
   Setting the **AWC_display_configured_revs_for_pwa** preference to **false** allows an engineer to copy the revision of an item, instead of copying the item itself, to the folder.

3. Create a revision rule with an override clause.

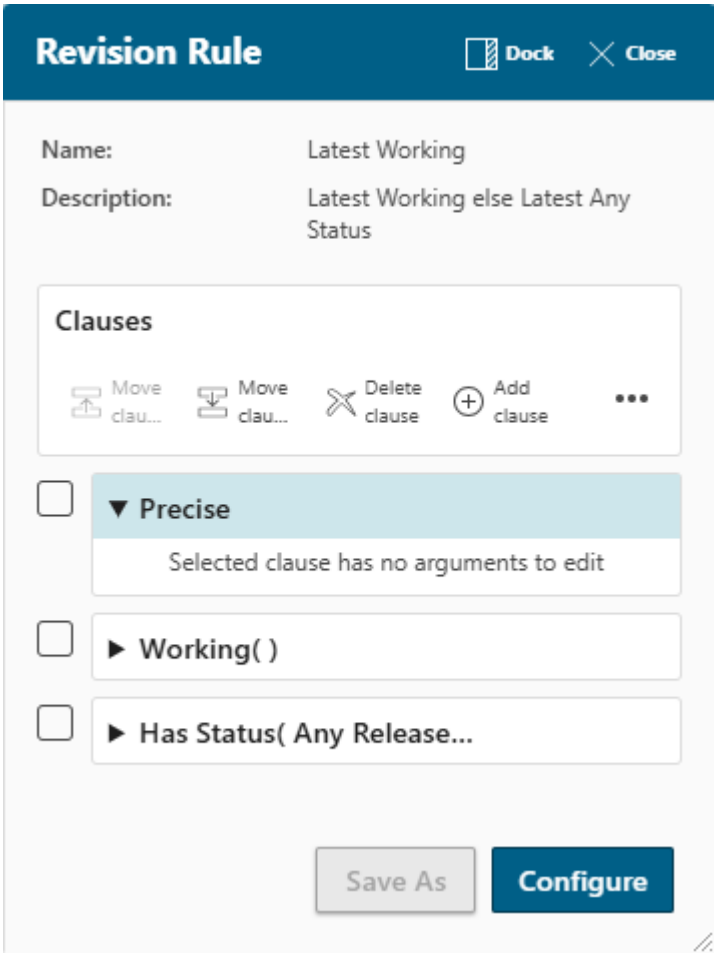
   - In Active Workspace, perform the following steps:

     a. Open a structure and click the **Revision** list.

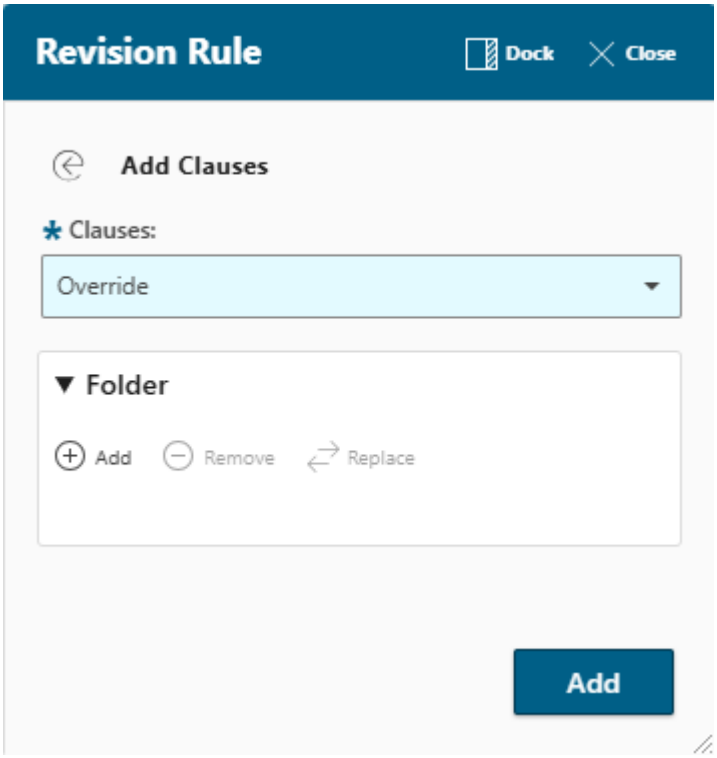
     b. Click ⓘ next to the revision rule that you want to modify.



     c. In the **Revision Rule** panel, click ⊕ **Add clause**.

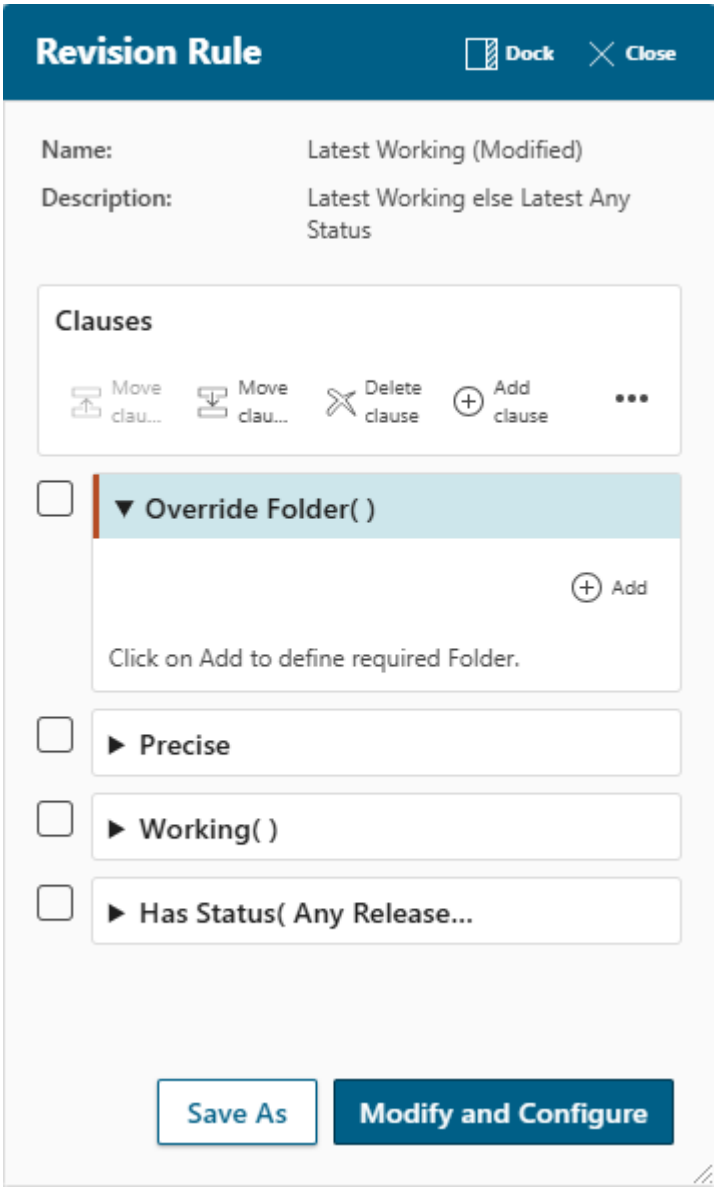


d. On the **Clauses** list, select **Override** and click **Add**.



e. Modify the clauses and set the precedence of the **Override** clause considering your requirement.

Considering your requirement, you can add multiple clauses with override folders to the revision rule.

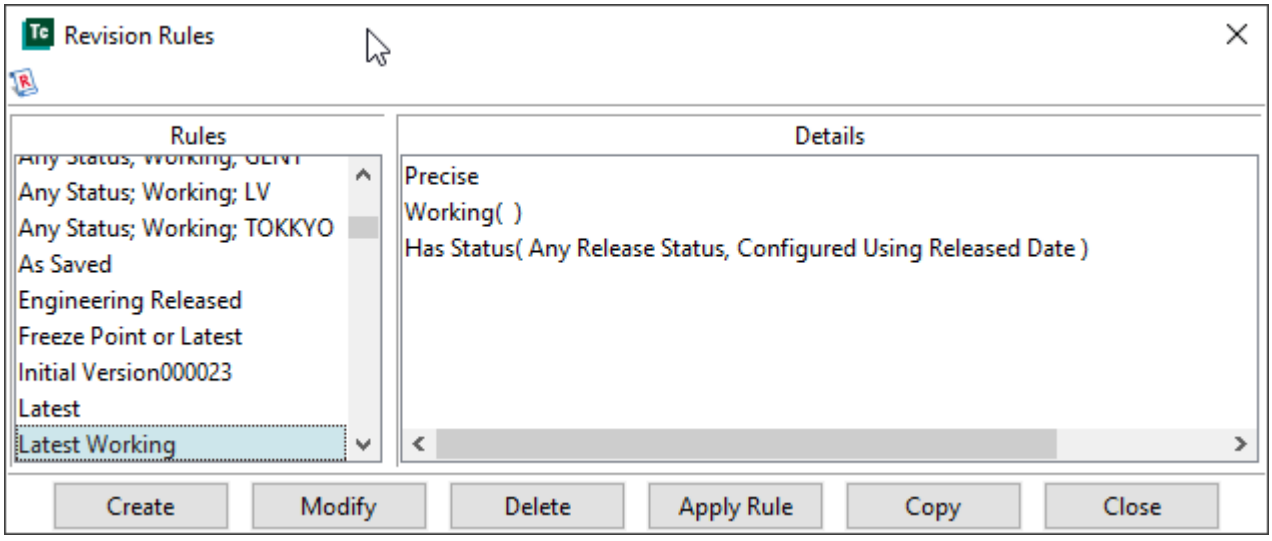


f. Click **Modify and Configure** to modify the selected revision rule.

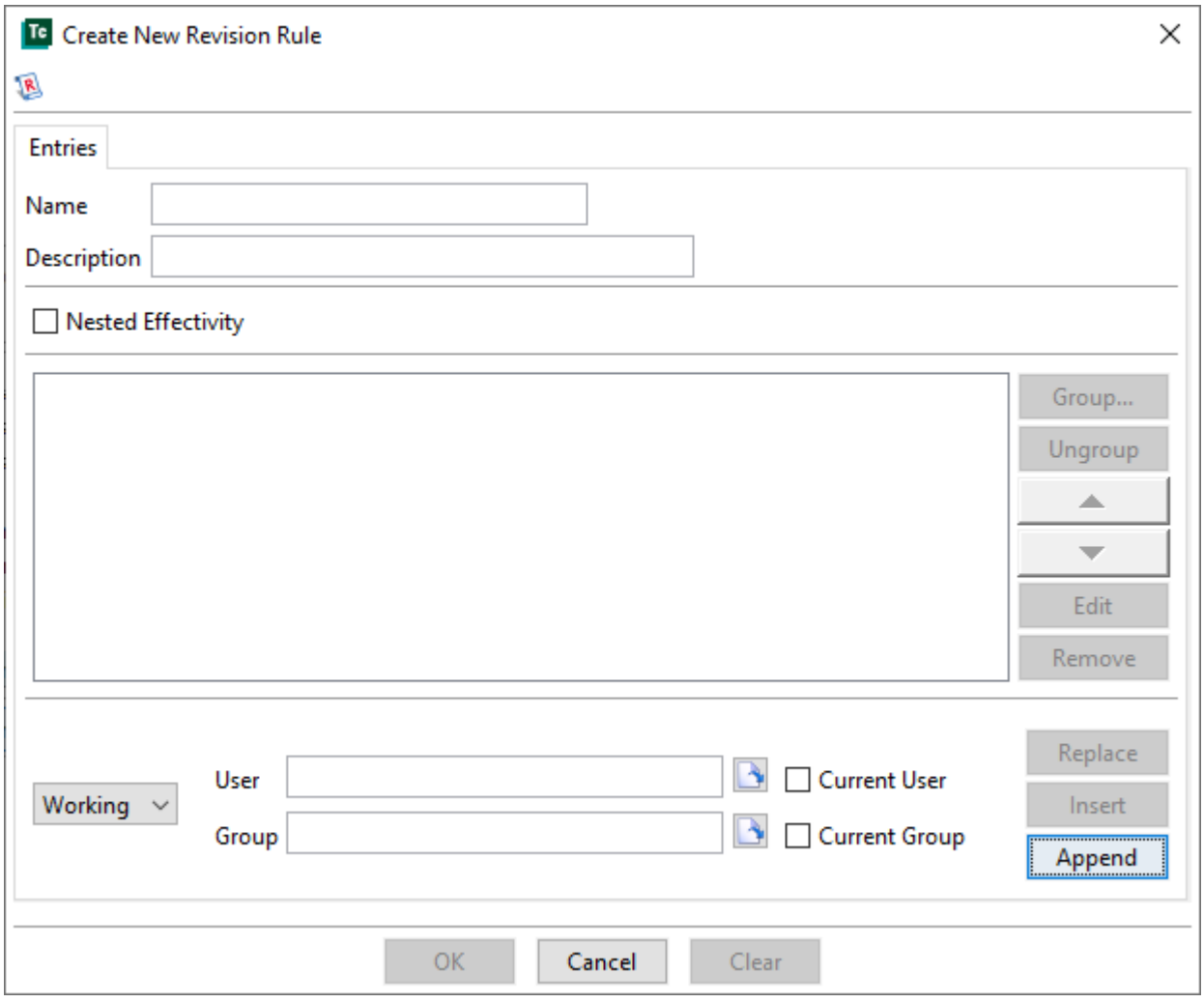
   Alternatively, click **Save As** and then enter a name and description for saving this revision rule with a new name.

- In the rich client, perform the following steps:

  a. In the Structure Manager application, click **Tools→Revision Rule→Create/Edit**.

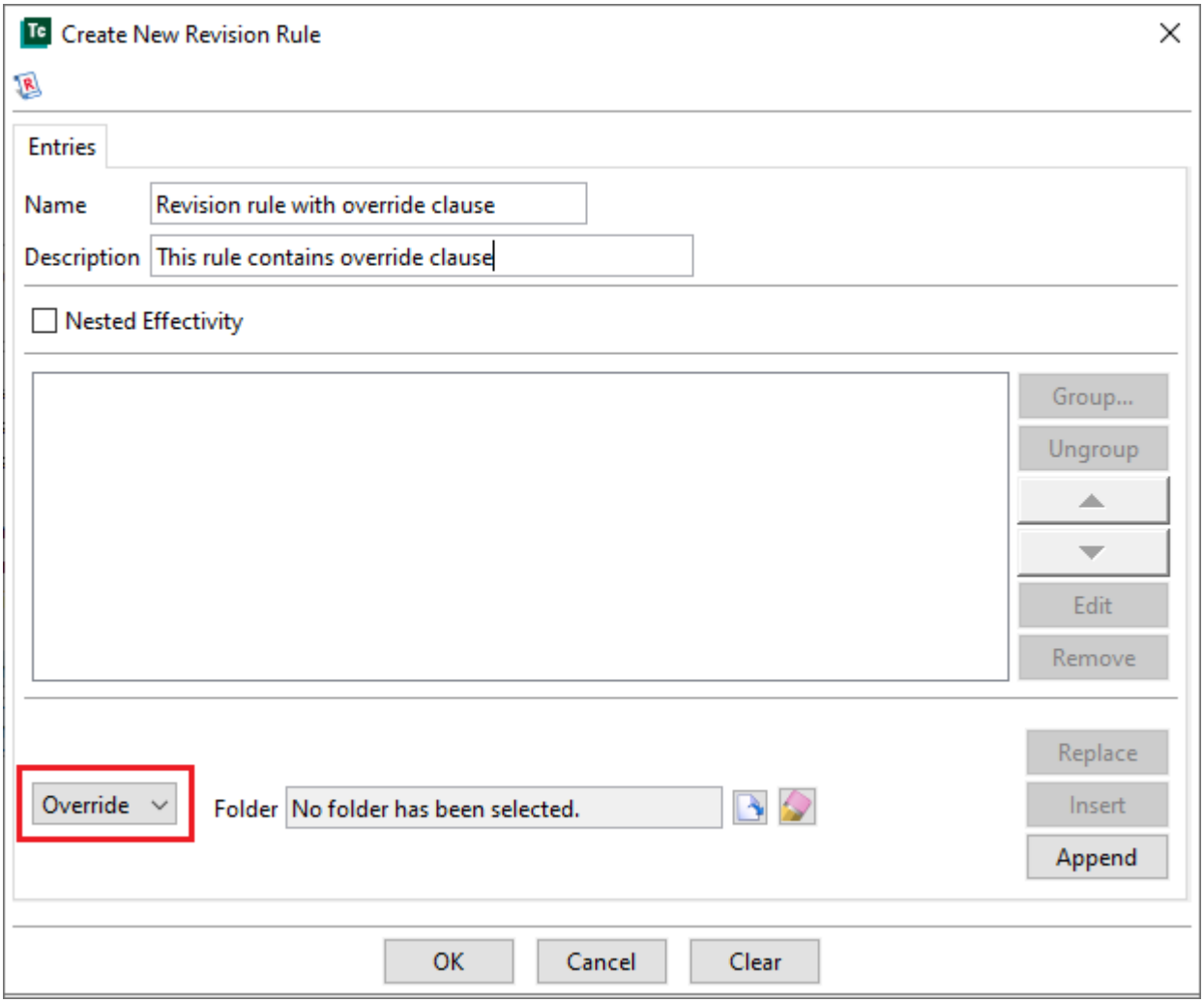


b. In the **Revision Rules** dialog box, select a rule that you want to use as a basis for creating the new revision rule in the **Rules** section, and click **Create**.



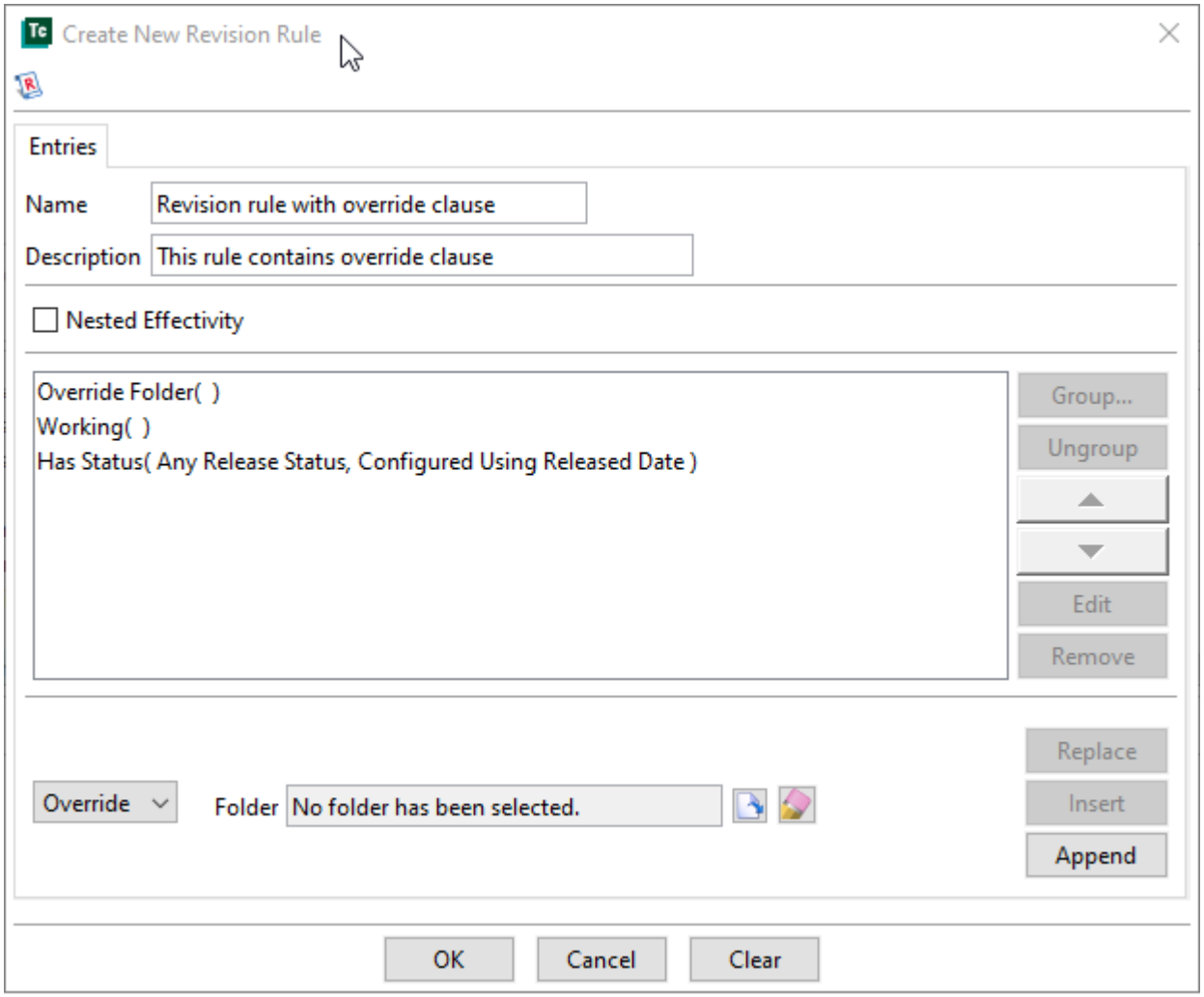
c. In the **Create New Revision Rule** dialog box, enter a name and description for the revision rule.

d. Select **Override**, and choose **the folder that you created in the previous step**.



e. Modify the clauses and set a precedence for the **Override** clause.

Considering your requirement, you can add multiple clauses with override folders to the revision rule.



f. Click **OK**.

The revision rule is created.

Set the default revision rule for a product structure

Teamcenter provides default revision rules that may be used at your site. **TC_Config_Rule_Name** specifies the default revision rule for structures. Set the **AWBUseDefaultRevisionRule** preference to `True` so that the configuration panel uses the existing **TC_Config_Rule_Name** as the default revision rule.

> Note:
>
> The **PSEShowUnconfigdEffPref** preference determines whether structure lines with effectivity that do not configure for the current revision rule or incremental changes are shown by default in a new Structure Manager window. If a structure line is shown even though its effectivity evaluates

to **false** for the current revision rule or incremental changes, you can use the following properties to identify the lines that do not configure for the current variant rule:

```
bl_has_date_effectivity=Y
bl_is_occ_configured=
```

Change the default revision rule for a workset

By default, when users open a workset, it uses the **Precise Only** revision rule. This is because the default value of the **TC_workset_config_rule_name** preference is **Precise Only**. You can change the default revision rule to another revision rule based on your site requirement. However, the new default revision rule must contain **Precise** as its first clause.

Provide access privilege for revision rules

Evaluate revision rules for read access

By default, Teamcenter applies a revision rule to configure a product structure before it applies an access rule. As a result, if a higher revision rule entry configures a particular revision of an item and the user has no read access to this revision, the structure line in Structure Manager shows an **UNREADABLE** stub entry.

You can optionally configure Teamcenter to continue evaluation when expanding the structure against the revision rule until it encounters a revision of the item to which the user has read access. However, if it finds no accessible item revision for any of the entries, it shows an **UNREADABLE** stub, as before. To allow Teamcenter to continue revision rule evaluation by making read access checks in this way, set the **PSE_check_read_access_for_Rev_Rule_entry** preference to **TRUE**. Structure Manager shows the next item revision to which you have access. If this preference is set to **FALSE**, evaluation stops when the first inaccessible configured item revision is encountered.

Unreadable lines may be shown or hidden, depending on the setting of the **BOM_hide_unreadable_lines** site preference. If the preference is set to **None**, unreadable lines are shown and labeled as **UNREADABLE**; this is the default setting. If the preference is set to **All**, unreadable lines are hidden. If the preference is set to a group name, unreadable lines are hidden from users in the specified group.

- If the user has no access to an item or GDE (generic design element) in an imprecise structure, the line is not displayed.
- If the user has access to the item in a precise structure but not the configured item revision, the line is displayed.
- If the BVR is precise and the user has no access to the item revision, the line is not displayed.
- If the BOM has only unreadable lines, the parent is shown as a leaf node.

Unreadable lines do not participate in roll ups, configuration, BOM compare, accountability checks, duplication, import/export, and searches. Hidden lines cannot be used as global alternates or substitutes.

You can use any of the following revision rule entries to show configured item revisions to which the user has read access:

- **Latest Working**
- **Status**
- **Latest Revision**
- **Override**

You can include more than one of these entries in the same revision rule to find accessible item revisions.

This behavior applies only to imprecise structures and does not affect the evaluation of precise structures. Similarly, there is no effect if you configure a structure by unit number or date effectivity.

Configure privileged and unprivileged users

User privileges determine if a user can create and modify revision rules, or only apply rules created by others. Specifically, privileged users have access to all the revision rule menu commands, while unprivileged users have access only to a subset of the commands. Consequently, unprivileged users cannot create or delete revision rules, and have only limited ability to modify revision rules without saving the changes.

Your Teamcenter administrator can restrict user access to these menu commands. Use the Command Suppression application to determine those roles that are privileged and so restrict access to the **Revision Rules** menu commands.

Control access to revision rules

Use Access Manager to control user access to revision rules. You can limit read access to control the users who can see and use a revision rule. You can use this technique to reduce the number of inapplicable revision rules that are presented to ordinary users, or to restrict rules to certain groups of users. You can use write access to control the users who can modify a revision rule.

You can apply an access rule globally to all rules using a class revision rule or other attribute, (for example, **OwningGroup**) if you created the revision rules appropriately. You can add object ACLs to specific revision rules for exception cases. A typical default access rule and rule tree ACL follow:

Access rule:

```
HasClass (RevisionRule) -> Private Rev Rule ACL
OwningGroup (dba) -> Public Rev Rule
```

Private revision rule ACL:

This ACL prevents Teamcenter displaying privately created revision rules to all users. Only the owning user and system administrator have access to the private rule. You can define an entry for **Owning User** that gives access to all users in the owning group. Alternatively, you could add it as an object ACL to the specific rule.

```
Owning User: Read, Write, Delete, Copy, Change
System Administrator: Read, Write, Delete, Change
World: No Read, No Write, No Delete, No Copy, No Change
```

Public revision rule ACL:

This ACL ensures that public revision rules are visible to all users. It also only allows users with a **configuration** role or members of a system administration group to modify public rules. You should control these permissions carefully, as unintended modification of revision rules can have significant consequences.

```
Role = Configurator: Write
System Administrator: Write, Delete, Change
World: Read, No Write, No Delete, No Copy, No Change
```

Improve the performance of BOM expansion during revision configuration

You can improve the performance of BOM expansion during revision configuration by using the **Teamcenter Revision Configuration Accelerator Service** service. Currently, this service supports the **Working** clause and the **Has Status** clause in a revision rule.

You must install the **Teamcenter Revision Configuration Accelerator Service** service and make sure that it is running. To install the service:

- In TEM, go to **Server Enhancements** > **Database Daemons** > **Teamcenter Revision Configuration Accelerator Service**.

  OR

- In Deployment Center, go to the **Components** tab and add the **Database Daemon** component. Under **Database Daemon Services**, select the **Teamcenter Revision Configuration Accelerator Service** check box.

The service runs in the background at regular intervals and updates the revision configuration results for the modified revision data.

20. Configure structure effectivity

Migrate legacy effectivity data to a new effectivity object

Earlier, the legacy effectivity data was stored in the **CFM_date_info** object. As this object is now obsolete, you must migrate the effectivity data to the new effectivity object called **Effectivity**.

To do this, in the Teamcenter command prompt, run the **effupgrade** utility in the **CFM date info occurrence effectivity upgrade** mode as follows:

```
effupgrade -cfm_date_info_occ
```

Along with the **-cfm_date_info_occ** mode, you must use the **Report**, **Upgrade**, and **Cleanup** submodes. You must use the **Report** submode first, then the **Upgrade** submode, and finally, the **Cleanup** submode. Optionally, you can use the **Upgrade** and **Cleanup** submodes together. As a system administrator or a business user, you must not modify any effectivity data while the utility is running in these submodes. If any data is modified, you must run the utility again, starting with the **Report** submode.

The **Report** submode creates a text report file for the legacy CFM date info data. The report file is used later by the **Upgrade** and **Cleanup** submodes. You must not modify this file. It is saved in a temporary folder. After this submode is run completely, the console displays the name of the text report file along with its file path. The following is the syntax for this submode:

```
effupgrade -cfm_date_info_occ -report
```

The **Upgrade** submode migrates the CFM date info data collected in the report file to the **Effectivity** object. After this submode is run completely, the migration status is updated to the report file. This same report file is then used by the **Cleanup** submode. The following is the syntax for this submode:

```
effupgrade -cfm_date_info_occ -upgrade -reportFile <report file name with
path>
```

The **Cleanup** submode cleans up the legacy **CFM_date_info** object after its data is migrated. If you provide the report file as an input, the cleanup is performed using the report file. After this submode is run completely, the cleanup status is updated to the report file. However, if you do not provide the report file, all the orphan **CFM_date_info** objects are cleaned up from the database. The following is the syntax for this submode:

```
effupgrade -cfm_date_info_occ -cleanup -reportFile <report file name with
path>
```

For more information about the **CFM date info occurrence effectivity upgrade** mode, in the Teamcenter command prompt, type the following command:

```
effupgrade -occ_eff_help
```

To view all the information available on the **effupgrade** utility, type the following command:

```
effupgrade -h
```

Display date effectivity without end item

If an end item is associated with a date effectivity, the end item is displayed along with the date effectivity. However, if an end item is not available, the text *(NONE)* is displayed after the date effectivity. To set how the date effectivity is displayed, set the following preference.

Preference	Value	Description
PSE_show_eff_empty_end_item	**true** This is the default value.	The date effectivity is displayed with the date followed by the end item (if an end item is available). For example, **2-Nov-2023 14:30 to 29-Apr-2023 14:30 to UP (037083)**
	false	Only date is displayed. For example, **3-Nov-2023 14:30 to UP (NONE)**

Allow users to view shared effectivity information

Users can view the shared effectivity associated to a structure in the **Table** view. The effectivity is shown when the **Table** view is selected in the **Overview** tab. To view effectivity in the **Table** view, add the following string in the **Awb0ContentTableItemRevUiConfigCots.xml** file:

```
<ColumnDef objectType="Awb0ConditionalElement"
propertyName="awb0ElementEffId" width="4000"/>
```

Enable multi-unit effectivities

Teamcenter allows you to configure product structure occurrences of an assembly based on specified multiple end items and the unit effectivity ranges for each of those end items. You can do impact analysis and eliminate the duplicate work required to maintain different product structures and complicated manual reconciliation.

A combination of multiple end items and range of units for each end item used to configure product structure occurrences is referred to as a *multi-unit configuration*. To enable the creation of multi-unit effectivities, the administrator must set the **Fnd0EnableMultiUnitConfiguration** global constant to **true** at each site with the Business Modeler IDE (BMIDE).

1. Open BMIDE with the foundation template.

2. Search for the **Fnd0EnableMultiUnitConfiguration** global constant and change the value to **true**. By default, it is set to **false**.

3. Create a custom template with the changes in BMIDE.

   For more information, see *Creating, deploying, and packaging templates* in *Configuring Your Business Data Model in BMIDE* on Support Center.

4. Deploy the template with the changes.

   For more information, see *Deploying templates* in *Configuring Your Business Data Model in BMIDE* on Support Center.

Control users who can set effectivity

To control the users who may set effectivity, add the following access rules at an appropriate place at the top level of the rule tree:

- **Has Type (Release Status)→Create Effectivity Users**
- **Has Type (Effectivity) - Edit Effectivity Users**

The release status rule controls who has write access to the release status and consequently can attach effectivity objects to it. This also determines who can initially create effectivity. Similarly, the effectivity rule controls who can edit an existing effectivity object.

21. Administer solution variants

Set up workflows to update solution variants

To allow users to update solution variants using a workflow, you must create a workflow process template by using the **SMC0-update-solution-variants** handler.

SMC0-update-solution-variants

Description

Updates solution variants for items or item revisions attached as target and reference respectively to the root task.

When a structure (item revision) is modified both within and outside of a change context, all solution variants associated with impacted item revisions are updated.

The default configuration for updating solution solutions is:

- Revision rule – Use the **Latest Working** revision rule.
- Effectivity – Use the effectivity of the active change. If an active change context is not set, effectivity is ignored.

The output is:

- Item revision attached as a target.
- Updated solution variants attached as a target.

Both are attached as target and added in that order. The source item revision is attached first and then its associated solution variants. Users of the workflow handler can check its type. The source item revision is of the item revision type and the solution variant is the item type.

Syntax

SMC0-create-solution-variants

Arguments

None

Placement

This handler can be placed on any action. It is typically placed at the **Complete** action of the root task so that the initial list is expanded at the start of the workflow process.

Restrictions

None

Examples

Use the following arguments and values to create a multilevel reused solution variant:

Argument	Values
Target Attachment	Item revision for which you want to update the associated reused solution variant.

Set up the column configuration for solution variants

You must perform the following steps so that the columns specific to solution variants are displayed on Active Workspace:

1. Verify if the **Active Content Structure**, **Product Configurator Support for Active Content Structure** (**smc1activeworkspacebom**), and **Solution Variant Support for Active Workspace** are installed.

2. Verify if the **smc1activeworkspacebom** template is set up in BMIDE.

3. (Optional) Add the custom properties that you want to be displayed as columns specific to solution variants on Active Workspace in the *Smc1SVPreviewUiConfigCots.xml* file. Additionally, add the custom properties that you want to be included in the **Solution Variants** table present on the **Overview** tab or in the **Solution Variants** section present on the **Solution Variants** tab of a configurable assembly (variable part) in the *SolutionVariantsProvider.xml* file. These files are located in the *TC_ROOT\smc1activeworkspacebom\data* folder.

   Use the **import_uiconfig** utility to import these files so that the custom properties are displayed on Active Workspace.

For detailed information on importing and merging column configurations, see the *Active Workspace Customization* help on Support Center.

Specify the source name and ID format for creating solution variants

In Teamcenter, if the **Enable Multi-Level Creation** option is enabled, you can use the **PSESolutionVariantItemProperties** preference to specify a format for the source name and the ID when creating a solution variant. This preference accepts a combination of values, which can be specified manually or can be system generated.

By default, this preference does not contain any values. If you do not specify any value or specify an invalid value, Teamcenter uses the name from the source structure and automatically generates the ID.

Procedure

1. Specify the internal names of the properties from the source Item Revision, separated by the plus (**+**) character. No spaces are allowed.

- To define the ID, use the following format: item_id_format=<itemrev_prop1>[+<itemrev_prop2>]*. For example, "item_id_format=custom_prop1+item_id+custom_prop2".

- To define the solution variant object name, use the following format: object_name_format=<itemrev_prop1>[+<itemrev_prop2>]*. For example, "object_name_format=custom_prop1+object_name+ custom_prop2".

The resulting object name or ID is a combination of the specified property values and the system generated ID if the specified property is **item_id**, separated by the minus (-) character, for example, "000260-custom_prop1_value". The source name or ID gets truncated if it exceeds the limit for the maximum allowed length.

2. Save your changes.

> Note:
>
> Teamcenter continues to support the earlier format for specifying values, for example, **item_id=bl_item_object_desc** and **object_name=bl_rev_object_desc**.

Define the reuse of a solution variant

By default, before creating a solution variant, Teamcenter compares the content of the configured source structure with solution variants that are already available. The content of the structure is based on the applied variant configuration. While comparing, if a matching solution variant is found, it is reused instead of creating a new one.

Teamcenter does the content comparison because the **SolutionVariantReuseBasedOnStructureMatch** preference is set to **true**, by default. For each unique saved variant rule (SVR) applied to the source configurable structure, if a matching solution variant is found, it is reused.

If you want new solution variants to be created for each each unique SVR, you must set this preference to **false**. On doing so, Teamcenter does not compare the content, and always creates new solution variants. However, for lower-level *reuse* assemblies where a subset variability scope is applied, solution variants are reused if the resulting features of the subset SVR matches an existing solution variant. A subset variability scope is a scope that belongs to the variability scope applied at the topmost level of the structure.

Disallow updating an existing solution variant

You can prevent updating an existing solution variant of type *Reuse* by configuring the statuses, so that a business user can obsolete it and create and then use a new solution variant.

To do that, you use the **SolutionVariantDisableUpdateStatusList** preference. You can add all valid Teamcenter statuses as values to this preference. The default value available in this preference is **Obsolete**. You can add more statuses to the list of statuses as per your business requirement.

When the user sets any of the statuses in the preference on a solution variant, automatic updates to the solution variant are disabled. The business user can replace this solution variant by either creating a new solution variant for its 150% configurable structure or by updating its parent solution variant.

Configure how a user creates, views, and updates solution variants

You can configure how a business user creates, views, and updates solution variants by using the **SolutionVariantProviderAvailable** preference.

Preference value	Solution variants table location	User action
False (default value)	**Solution Variants** tab > **Solution Variants** section	A user sets the **Solution Variant Category** to **Reuse** and updates a solution variant manually.
True	**Overview** tab > **Solution Variants** section	A user updates a solution variant using a workflow from the same location.

Specify the occurrence properties to be synced from the source structure to the solution variant while the solution variant is updated

You can specify the occurrence properties that are to be synced from the source structure to the solution variant while the solution variant is being updated. To do that, you use the **SolutionVariantSyncProperties** preference.

By default, this preference contains the following values:

- **bl_quantity**
- **bl_plmxml_occ_xform**
- **bl_occ_effectivity**
- **bl_ref_designator**
- **bl_sequence_no**
- **AIE_OCC_NAME**
- **AIE_OCC_ID**
- **AIE_Exported**

You can modify the preceding list to add any additional occurrence properties, such as Occurrence Note or any other custom occurrence properties. A property value must be an internal name of a BOM line property. The substitutes for a structure element are synced automatically. You need not include those as values in this preference.

In addition to carrying over the absolute data properties, if you want to carry over the **Position Designator**, **ID in Context**, and **Usage Address** properties, add the following values to the **SolutionVariantSyncProperties** preference:

- **bl_abs_occ_id**
- **bl_position_designator**
- **bl_usage_address**

22. Configure Product Configurator variants

Modify AND and OR operators in variant conditions

1. In your Teamcenter installation directory, search for the **tc_text_locale.xml** file, for example, *TC_ROOT***lang\textserver\en_US\tc_text_locale.xml**.

2. Create a backup of this file.

3. Open the file and search for the **k_variant_op_and** and **k_variant_op_or** values and modify them.

   Example:

   ```
   <key id="k_variant_op_and">AND</key>
   <key id="k_variant_op_or">OR</key>
   ```

   Change as follows:

   ```
   <key id="k_variant_op_and">&amp;</key>
   <key id="k_variant_op_or">|</key>
   ```

4. Start Teamcenter and confirm that formula representation is now changed from **AND, OR** to **&, |** successfully.

Enable the tracking of variant changes

To enable the tracking of variant changes, set the **CM_track_variant_condition_changes** preference to **ChangeNotice** types.

When users save changes to a structure containing variants, if this preference is **ON** for the current change type, Teamcenter tracks variant changes between the solution item and the impacted item.

23. Configure the packing of similar structure elements

The packing action groups multiple identical components in one level of an assembly. It groups the components that satisfy the packing criteria, which in turn is configured by setting the following preferences.

- Configure the packing action to exclude or include the sequence number as the packing criteria by specifying the **BOMExcludeFromPackCheck** preference.

  Syntax: **BOMExcludeFromPackCheck**:`seqno|none`

  - If this preference is set to `seqno`, find numbers are excluded from the BOM line pack check. It allows the BOM lines with distinct find numbers to be packed.
  - If the preference is set to `none`, no attributes are ignored during the default pack check. BOM lines must have the same part number and the find number to be packed.

- Specify additional packing criteria by setting the **BOM_Additional_Packing_Criteria** preference.

  Syntax: **BOM_Additional_Packing_Criteria**:`property name`

  - If only the property name is specified, then lines having the same value for the property are packed.
  - If the property name is followed by a colon, the string after the colon is the value when the pack is not allowed.
  - No string after a colon indicates an empty value.

- Configure the structures to load with packed elements by default.

  Syntax: **PSEAutoPackPref**:`0/1`

  - Set the **PSEAutoPackPref** to `1` to load the structures in the *packed* view.
  - Set it to `0` to retain the default view as *unpacked*.

- Configure the reference designator packing rules.

  Syntax: **BOM_Enable_Ref_Designator_Value_Packing**:`True/false`

  Set the **BOM_Enable_Ref_Designator_Value_Packing** preference to configure the reference designator packing rules. You can pack or unpack product structure lines that include reference designators. For example, if **BOM_Enable_Ref_Designator_Value_Packing** is set to `True`, eight BOM

lines with the reference designators **C1, C5, C6, C7, C10, C14, C15, C16** will be packed to one BOM line with the reference designator property **C1, C5-7, C10, C14-16**.

The reference designators are shown in the International Organization for Standardization (ISO) and in the American Society of Mechanical Engineers (ASME) formats. In the ASME format, the reference designator is shown in the following format: uppercase letter followed by a numeral and a letter as a suffix, for example, **S1A**.

24. Configure structure comparison

About configuring structure comparison

Whenever two structures are compared, equivalent occurrences between the two structures are identified first. The system uses the **ID In Context** property value to identify equivalent occurrences. Properties other than **ID In Context** can also be used to identify equivalent occurrences in the structures.

What happens when Dynamic Equivalence check box is not selected?

The system performs an accountability check to compare the two structures. *Accountability check* is the verification mechanism to check if all elements in one structure have equivalent elements in other structure. For users to perform an accountability check, ensure that the Multi-Structure Foundation license is obtained and installed in your Teamcenter setup.

While performing an accountability check, the system first compares the **Id In Context** property.

If **Matched** option is selected in comparison results, those properties that have same **Id In Context** property will be shown in the comparison results as matching.

If **Different** option is selected, then the properties that have matching **ID In Context** property are further compared based on the properties specified in the **AWBPropertiesToCompare.BVR** and **AWBPropertiesToCompare.EBOM** preferences. The elements that have different values for the compared properties are highlighted as partial matches.

What happens when Dynamic Equivalence check box is selected?

The system compares the structure elements based on the properties specified in the **MEAccountabilityCheckDynamicIDICProperties** preference. By default, this preference contains the **Item ID** property. Any other property added to this preference will be included in the comparison.

> Example:
>
> If the default **Item ID** property is used as the equivalence criteria, the lines with the same **Item ID** are treated as equivalent.

When the **Dynamic Equivalence** check box is selected, the **ID In Context** property is ignored unless it is added to the **MEAccountabilityCheckDynamicIDICProperties** preference.

After the dynamic equivalence check, the equivalent lines are further compared to find if it is a partial match or a full match. Properties for this comparison are specified in the **AWBPropertiesToCompare.BVR** and **AWBPropertiesToCompare.EBOM** preferences. By default, this preference contains **Find No.** and **Item revision ID**.

MEAccountabilityCheckDynamicIDICProperties

The following example shows the **MEAccountabilityCheckDynamicIDICProperties** preference syntax and description.

Example:

```
<preference name="MEAccountabilityCheckDynamicIDICProperties"
type="String" array="true" disabled="false" protectionScope="Site"
envEnabled="false">

<preference_description>
Defines the list of BOM Line properties used by accountability check
dynamic equivalence criteria when searching for equivalent BOM Lines.
By default the BOM Line property "bl_item_item_id" is always included
in dynamic equivalence criteria, unless certain structure search
friendly properties are provided:
- internal BOM Line properties pointing to Occurrence Note
- compound property on a Form
- property on an Item type, description or name
(e.g. "bl_item_object_type")
- property on an Item Revision type, description or name
(e.g. "bl_rev_object_name")
If none of these is available, the property "bl_item_item_id" will
be used in conjunction with any other specified properties. The
first cacheless(structure) search property in the preference is
used as an initial property to collect the BOM Lines on which the
other properties are evaluated. Empty property values are not
considered as equal during compare. As an example for a value to
search for absolute transformations, the entry "bl_plmxml_abs_xform"
can be added.
 </preference_description>
      <context name="Teamcenter">
           <value>bl_item_item_id</value>
      </context>
    </preference>
```

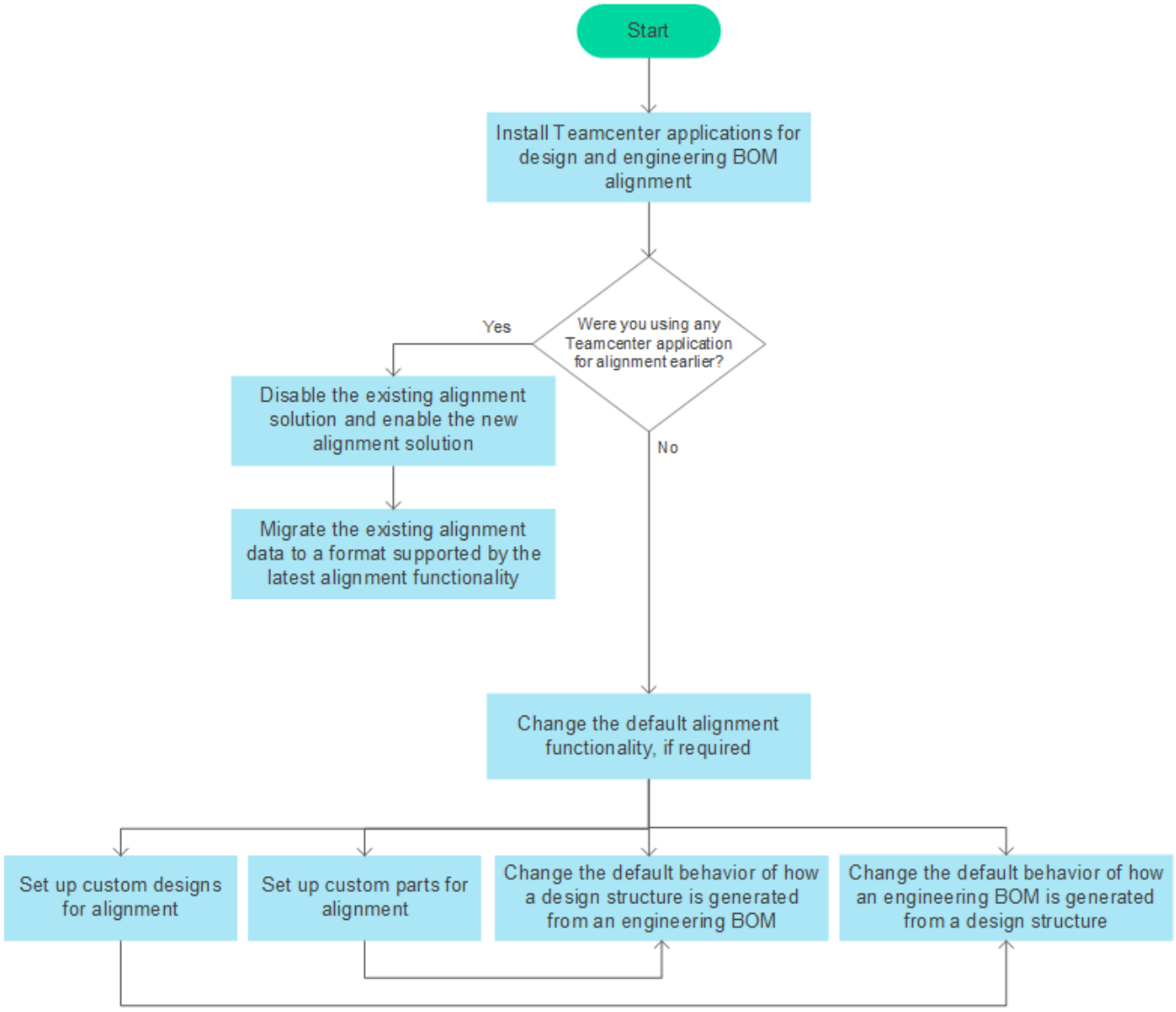
If you specify a different value (other than the default one, **bl_item_item_id**), an accountability check is performed to compare two structures. Accountability check requires the Multi-Structure Foundation license.

Specify comparison properties and comparison result retention period

For comparing two part structures and for comparing two parts within a part structure, you must specify the **BOMLine** properties to be compared in the **AWBPropertiesToCompare.BVR** preference.

Additionally, you can define for how long the comparison results can be retained in the **Awb0CompareResultCleanupDays** preference. After this time period, the comparison results are deleted.

25. Set up design BOM and engineering BOM alignment



For detailed information on how to set up design and engineernig BOM alignment, see *Design and BOM Alignment — Deployment and Administration*.

26. Validate the engineering BOM setup

After deploying engineering BOM, and making the required customization and administration changes, you must validate if engineering BOM is set up correctly.

Procedure

1. In the Teamcenter command prompt, open the *TC_ROOT***TR\tc_menu** directory. Here, *TC_ROOT* is the folder where Teamcenter is installed.

2. Run the **validate_ebom_configurations** utility. For detailed information about the utility, see its help:

   validate_ebom_configurations -h

   > Example:
   >
   > To validate the engineering BOM setup for the assembly BOM mode for the **Part** and **Design** business objects, run the utility as follows:
   >
   > **validate_ebom_configurations -u=***tc_admin_username* **-p=***tc_admin_password* **-g=dba -conf=AsmBOM -partTypeName=Part -designTypeName=Design**

   This utility generates a report in the log file location shown on the command prompt. The report lists the issues if the setup fails.

   Each issue is listed as an **ERROR**. The action to be taken to rectify the issue is listed next to it.