TEAMCENTER

Basic Classification — 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

Deploying and administering basic classification 1-1

What is the difference between basic and advanced classification? 2-1

Planning the deployment of basic classification 3-1

Installation options 4-1

Install classification using Deployment Center 5-1

Install classification using Teamcenter Environment Manager 6-1

Configuring and administering basic classification on Active Workspace

Configuring basic classification on Active Workspace 7-1
- Understanding the presentation layer and its required configurations 7-1
- Enable the presentation layer 7-1
- Configuring search for basic classification 7-2
- Configuring the classification interface 7-16
- Configuring interchangeable attributes 7-28
- Creating graphics for an object based on class templates 7-29
- Configuring classification enforcement 7-30
- Configuring optimized unit values 7-31
- Import pre-packaged classification hierarchy for ECAD, MCAD, and routing data for basic classification 7-32
- About working with advanced and basic classification simultaneously 7-34
- Troubleshooting the installation and configuration 7-41
- Give access to users for classifying objects 7-43
- Granting access to non-dba users for Classification Manager tasks 7-43

Administering basic classification on Active Workspace 7-44
- Viewing your hierarchy definitions 7-44
- Authoring hierarchy definitions 7-45
- Indexing classification data 7-56
- Administering classification libraries 7-58
- Sharing classification data 7-65
- Classification preferences and utilities 7-75

Configuring and administering basic classification on Rich Client

Getting started with Classification Admin 8-1
- Getting started with Classification Admin 8-1

Classification Admin interface 8-2
Basic concepts for using Classification Admin 8-4
Basic tasks using Classification Admin 8-5
Customizing Classification menu commands 8-6
Specifying classifiable item types 8-7
Localizing Classification 8-7
Administering Classification 8-8
Displaying the classification hierarchy 8-8
Displaying the classification hierarchy 8-8
Displaying classes in the hierarchy 8-9
View the tree graphically 8-10
Displaying a subset of the hierarchy 8-11
Refresh the hierarchy tree 8-12
Configuring the search by type 8-13
Creating and managing groups 8-13
Create a group 8-13
Modify a group definition 8-15
Delete a group 8-16
Add an image to a group 8-16
Display information in the Properties pane 8-17
Viewing attribute values 8-17
Creating and managing classes 8-19
Creating and managing classes overview 8-19
Classifying a single workspace object multiple times 8-19
Create a class 8-20
Assign attributes to a class 8-22
Apply properties to attributes 8-24
Specify the applicability of attributes 8-28
Remove an attribute from a class 8-30
Setting a default value for a class attribute 8-30
List attribute values 8-31
Save the class definition 8-31
Modify a class 8-32
Copy a class 8-32
Moving a class 8-34
Delete a class 8-34
About deep copy rules 8-35
Managing units of measurement 8-35
Create or modify a unit definition 8-37
Protecting unit definitions from modification 8-38
Set a measurement system for a class 8-38
Mapping classification attributes to NX part attributes 8-39
Add images to classes 8-39
Mapping a class to another class 8-39
Display the autofilter 8-39
Creating and managing views 8-40
Working with Classification views 8-40
Create a view 8-41
Assign attributes to the view 8-42

Apply properties to attributes 8-43
Setting a default value for a view attribute 8-45
Customizing the layout of attributes and attribute fields 8-46
Creating a user-defined button 8-50
Specify applicable attributes in a view 8-67
Modify a view 8-67
Copy a view 8-68
Delete a view 8-68
Define a mapping view 8-69
Using the SUBSTR operator when mapping attributes 8-70
Allowable types for mapping classes 8-71
Create mapping views for NX 8-72
Creating and managing key-LOVs 8-73
Create a key-LOV 8-73
Add or insert key-LOV entries 8-74
Add or insert a key-LOV submenu 8-75
Create an interdependent key-LOV 8-76
Preview a key-LOV entry 8-79
Save a key-LOV entry 8-79
Remove a key-LOV entry 8-79
Modify a key-LOV entry 8-80
Deprecate a key-LOV entry 8-80
Undeprecate a key-LOV entry 8-81
Creating and managing the attribute dictionary 8-82
Attribute dictionary overview 8-82
Create attribute dictionary definitions 8-82
Defining units and formats 8-84
Assigning reference attributes 8-90
Optimizing attribute values 8-95
Modify attribute dictionary definitions 8-96
Delete attribute dictionary definitions 8-97
Add user-defined data to an attribute 8-97
Search the dictionary for an attribute 8-98
Using custom logic to automatically compute attribute values 8-101
Controlling access to classification objects 8-103
Classification access control overview 8-103
ICO protection 8-105
Classification access privileges 8-107
Granting access to administrative objects to non-dba users 8-109
Create a classification access rule 8-110
Example: controlling the display of the hierarchy tree for Classification users 8-111
Example: controlling access to hierarchy definitions 8-112
Example: controlling access to ICOs 8-114
Create a named access control list (ACL) 8-114
Modify access control list entries 8-115
Delete access rules 8-116
Searching the classification hierarchy 8-117
Classification hierarchy search overview 8-117
Search using the quick search feature 8-117

Search using the Search Class dialog box 8-119
Viewing files associated with to groups, classes, and views 8-120
Adding images to groups, classes, and views 8-120
Add an image to a group, class, or view 8-120
Remove an image from a group, class, or view 8-121
View GIF images 8-121
Viewing files associated with an ICO in the viewer 8-122
Generating graphics for classification objects 8-122
Graphics generation overview 8-122
Configure the graphics builder 8-123
Understanding part family templates 8-125
Understanding template parts 8-126
Associating part family templates or template parts with class definitions 8-127
Attach a part family template or template part to a class 8-127
Mapping part family template attributes or template part expressions to class attributes 8-130
Automatically map attributes 8-131
Selectively map attributes 8-131
Delete a template from a class 8-132
Create classification instances (ICOs) for part family members 8-132
Create graphics using legacy Genius4000 Tcl scripts 8-133
Create graphics when template parts or part family templates have a multifield key identifier 8-135
Setting template or script priority 8-135
Sharing template data using Multi-Site Collaboration 8-138
Sharing classification hierarchy data 8-139
Overview of sharing with Multi-Site Collaboration 8-139
Multi-Site Collaboration sites 8-139
Object ownership 8-140
Object dependencies 8-140
Understanding access rights 8-141
Example: configuring access rules for Multi-Site use cases 8-141
Synchronizing classification data 8-143
Share classification hierarchy objects 8-143
Remove sharing of classification hierarchy objects 8-144
Delete a class at the master site 8-144
Transfer object ownership 8-144
Add or remove attributes in shared classes 8-145
Transitioning from text file sharing to Multi-Site Collaboration 8-145
Customizing the hierarchy tree display 8-147
Hierarchy tree customization overview 8-147
Associate a custom symbol with a group or class 8-148
Remove a custom symbol from a group or class 8-148
Configuring Resource Manager features 8-149
Configuring the guided component search 8-149
Importing a tool vendor catalog 8-165
Setting up site-specific properties to search resources by site 8-171
Working with the data dictionary 8-175
Data dictionary overview 8-175

Set up a data dictionary 8-175
View library information 8-176
Deleting a data dictionary from the hierarchy 8-176
Importing and exporting hierarchy data 8-176
Overview of importing and exporting hierarchy data 8-176
Export data using PLM XML 8-177
Using transfer modes 8-178
Modify the ConfiguredDataExportDefault transfer mode 8-179
PLM XML restrictions 8-180
Migrating SML subclasses to storage classes 8-180
Legacy SML subclass migration 8-180
Migrate classes and subclasses 8-181
Administering classification libraries 8-181
Library management overview 8-181
About specifications 8-183
Types of objects you work with 8-185
Install classification libraries 8-186
Using the clsutility and the lbrmanager utility 8-187
Install and configure library management 8-188
Extend classes to the presentation layer using clsutility 8-189
Synchronize the presentation layer with the classification hierarchy 8-189
Create library data using the lbrmanager utility 8-191

1. Deploying and administering basic classification

New product development and product improvement commonly involve the reuse of existing elements. This increases efficiency and savings. Digital libraries contain massive quantities of objects that are unrelated to each other except for how they are used or reused. *Classifying* these objects using descriptive attributes that you can search ensures they can be easily found. Teamcenter Classification helps business users to manage and classify product data efficiently. Teamcenter contains classification hierarchies, classes, and attributes organized into class definitions so that business users can quickly find objects for reuse.

As an installer, you can install and setup classification for the business users and classification administrator. As an administrator, you can create a classification hierarchy containing classes and class attributes that helps you describe the objects that you want to reuse in your organization. The classification administrator sets up such features as the sharing of hierarchy and classified data, adding images to classes, or enabling graphics creation for part family templates. You can deploy and administer basic classification on the Rich Client or on Active Workspace.

Where do I go from here?

 Business User	Using classification for reuse and standardization on Active Workspace Using classification for reuse and standardization on Rich Client
You want to learn more about classification standard taxonomy.	What is classification standard taxonomy?
 Installer	
You want to deploy and administer advanced classification.	Advanced Classification — Deployment and Administration
How to deploy and set up basic classification?	On **Active Workspace** On **Rich Client**
Learn which components must be installed for various basic classification scenarios	**Install classification**
See a quick overview of installation options	**Installation options**
 Classification administrator	
You use classification in the Teamcenter rich client and now want to search for and author classification data in Active Workspace.	**Configure full-feature classification for traditional classification data**

How do different groups of users see only the branches of the classification hierarchy that are relevant to their work?	**Refer to the information on how to configure classification libraries.**
How do you create classification hierarchy for basic and advanced classification data?	**Refer to authoring classification hierarchy objects in Classification Manager**

2. What is the difference between basic and advanced classification?

When using the classification feature, there are two types of data that can exist in your classification hierarchy — Basic and Advanced. These are some of the differences between these two types of data:

	Basic classification	Advanced classification
Availability	Available on the rich client and Active Workspace.	Available on Active Workspace.
Effectiveness	Effective in defining objects uniquely with properties for reuse.	Effective in capturing overall product information for PIM and MDM systems with support for ECLASS standard class hierarchy and definitions.
Standard Features	Some of the basic classification features: • Classification hierarchical representation • Flat list of properties • List of values • Unit of measure • Support for views • Limited to 200 Properties in a class	Some of the advanced classification features: • Class definition versioning • Flexible data modeling with attribute blocks, aspects, cardinality, and polymorphism. • Complex data type attributes • Native data storage • Support for namespaces • Unlimited number of properties • No Array limitations
Data model standards	Supports DIN-4000 standard and confirms to underlying specifications of ISO/TS-13399 standard.	Supports ECLASS standard and conforms to DIN 4002, ISO 13584-32, ISO 13584-42, IEC 61360 standards.
Hierarchy and class definitions	Define custom hierarchy and class definitions.	Define custom hierarchy and class definitions. Additionally supports standard based ECLASS hierarchy spanning more than 48 domains.
Data exchange support	Supports PLMXML, TCXML, and Multi-Site.	Supports JSON, OntoML, BMEcat.

There is very little difference between these two types of data in the user interface. These data types are, however, installed and configured differently. If you have questions, contact your classification administrator.

3. Planning the deployment of basic classification

Before deploying classification, plan your deployment considering the following scenarios:

❒	You are already using the basic classification functionality using the Teamcenter rich client and you want to view this data in Active Workspace.
❒	You are currently using the basic classification functionality and want to migrate to the advanced classification functionality.
❒	You want to use both basic and advanced classification functionality.
❒	You want to view classification data related to your project (library management).

Once you have identified the scenarios that are applicable in your case, you can **install Teamcenter Classification** and then set up classification by creating the classification hierarchy and performing additional configurations.

4. Installation options

	Install advanced classification	Create search indexing views	Extend data to presentation layer using clsutility	Set preference CLS_auto_sync_node_hierarchy to true	Set CLS_is_presentation_hierarchy_active to true	Index Data
Use traditional basic classification	X	✓	X	X	X	✓
Use additional features with traditional basic classification[1]	X	✓	✓	✓	✓	✓
Use both advanced and traditional basic classification simultaneously	✓	✓	✓	✓	✓	✓

Legend	
✓	Required
X	Not required

The following additional features are available for basic classification after installing the presentation layer and extending the data:

- CLASSIFICATION tile on the Home page
- **Data sharing with other sites using Multi-Site** or collaborating using PLM XML or briefcase files
- Search similar to search for an object based on the properties of another object

5. Install classification using Deployment Center

Add the Classification application to your existing Teamcenter environment.

Procedure

1. Log on to Deployment Center and select the environment to which you want to add Classification.

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 depending on your scenario:

Scenario	Applications to select for Active Workspace based solution	Applications to select for Rich Client based solution
Traditional basic classification	**Classification Active Workspace**	Automatically installed
Traditional basic classification and advanced classification together	**Advanced Classification** **Classification Active Workspace**	Not supported
Library Management	**Classification Active Workspace** **Teamcenter Classification Collector (earlier called Library Management)**	**Teamcenter Classification Collector (earlier called Library Management)**
Classification localization support Supports internalization and localization of classification	**Classification L10N**	**Classification L10N**

   Select the application, and then 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 each script to its target machine, and then run each script on its target machine.

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

Postrequisites

If you want to use classification with Active Workspace, ensure that you **enable the presentation layer**.

6. Install classification using Teamcenter Environment Manager

You can install the classification functionality through Teamcenter Environment Manager.

Procedure

1. Launch Teamcenter Environment Manager.

2. Proceed to the **Features** panel and select the features based on your requirements.

Functionality required	Features to select
View traditional basic classification classes and data in Active Workspace	• **Active Workspace** > **Client** > **Reuse and Standardization** > **Classification Client** • **Active Workspace** > **Client** > **Reuse and Standardization** > **Classification Server**
Use full feature classification with traditional data	• **Active Workspace** > **Client** > **Reuse and Standardization** > **Classification Client** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Classification Server** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Presentation Layer - Next Generation Classification Server**
Use traditional basic classification and advanced classification with ECLASS data simultaneously	• **Active Workspace** > **Client** > **Reuse and Standardization** > **Classification Client** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Classification Server** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Presentation Layer - Next Generation Classification Server** • **Extensions** > **Reuse and Standardization** > **Advanced Classification**
Install Library management	• **Active Workspace** > **Client** > **Reuse and Standardization** > **Classification Client** • **Active Workspace** > **Client** > **Reuse and Standardization** > **Teamcenter Classification Collector**

Functionality required	Features to select
	• **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Classification Server** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Presentation Layer - Next Generation Classification Server** • **Active Workspace** > **Server Extensions** > **Reuse and Standardization** > **Teamcenter Classification Collector Server** • **Extensions** > **Reuse and Standardization** > **Teamcenter Classification Collector**

3. After selecting the features perform the next steps to complete the installation.

7. Configuring and administering basic classification on Active Workspace

Configuring basic classification on Active Workspace

Understanding the presentation layer and its required configurations

The presentation layer provides full featured classification Active Workspace functionality. It provides more flexibility than the traditional basic classification storage hierarchy. It serves as the basis for implementing other classification features such as advanced classification (classification standard taxonomy or CST) and classification libraries. It provides the ability to deal with traditional basic classification features (such as creating a class hierarchy, adding classification objects to a classification hierarchy, classifying workspace objects, searching for classification objects, and modifying and deleting objects from a classification hierarchy) and advanced CST and classification library management features simultaneously.

Additionally, the presentation layer adds the **Classification** tile to the **home page** and allows you to configure visual navigation cards for searching.

Ensure that you do the following in order to work with presentation layer classification:

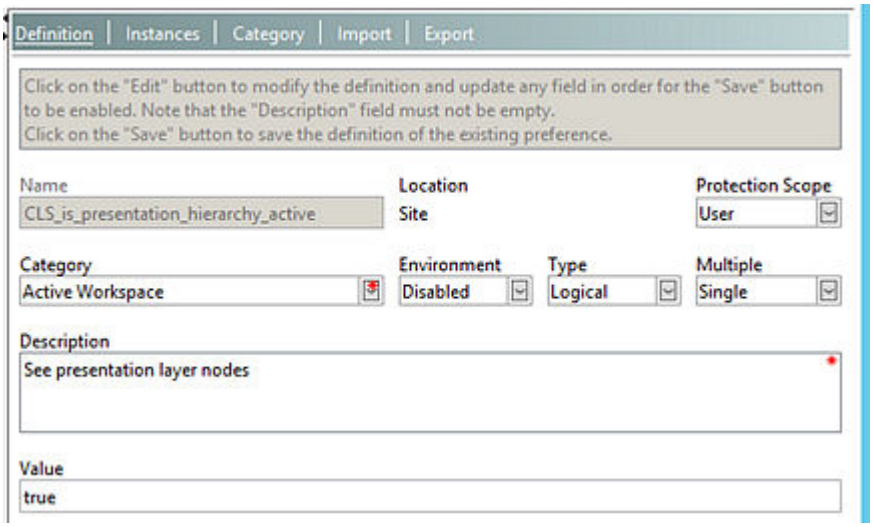
- Turn on the visibility of the presentation hierarchy using the **CLS_is_presentation_hierarchy_active** preference to **true**.
- **Extend the traditional basic classification classes to the presentation layer.**
- **Turn on automatic synchronization of the presentation layer with the classification hierarchy by setting the CLS_auto_sync_node_hierarchy to true.**

Enable the presentation layer

The presentation layer provides the ability to deal with traditional basic classification features (such as creating a class hierarchy, adding classification objects to a classification hierarchy, classifying workspace objects, searching for classification objects, and modifying and deleting objects from a classification hierarchy) and advanced CST and classification library management features simultaneously.

Additionally, the presentation layer adds the **Classification** tile to the **home page** and allows you to configure visual navigation cards for searching.

1. Make the node hierarchy (presentation layer) visible in the Active Workspace user interface by updating the value of the **CLS_is_presentation_hierarchy_active** user preference to **true**.



Being able to turn the visibility of the node hierarchy on and off per user allows users of both traditional classification and CST to classify objects in the same environment. Depending on the setting, a user can see either the traditional classification classes or the CST classes.

> Note:
>
> CST classes are not subject to access control. Therefore, if your environment contains both CST and traditional classification classes, if the presentation hierarchy is not active (**CLS_is_presentation_hierarchy_active=false**), CST classes are displayed in the search results for traditional classification users.

Configuring search for basic classification

Configure search for traditional basic classification

After installing classification or after adding data (classes, views, properties), you must enable searching for the data.

1. *Create search index views and choose attributes by which to search and filter*.

   Each property for which you want to search must be contained in a search index view.

2. *Make properties searchable in the global search*.

   Because you added properties to search index views, you must run the **bmide_modeltool** utility to update the facets.

3. **Index classification data**.

Any time you create new classification data, such as classifying new objects, you must re-index to be able to search for the new object in the global search or see the property displayed in the list of filters when searching for data. There are two ways to re-index:

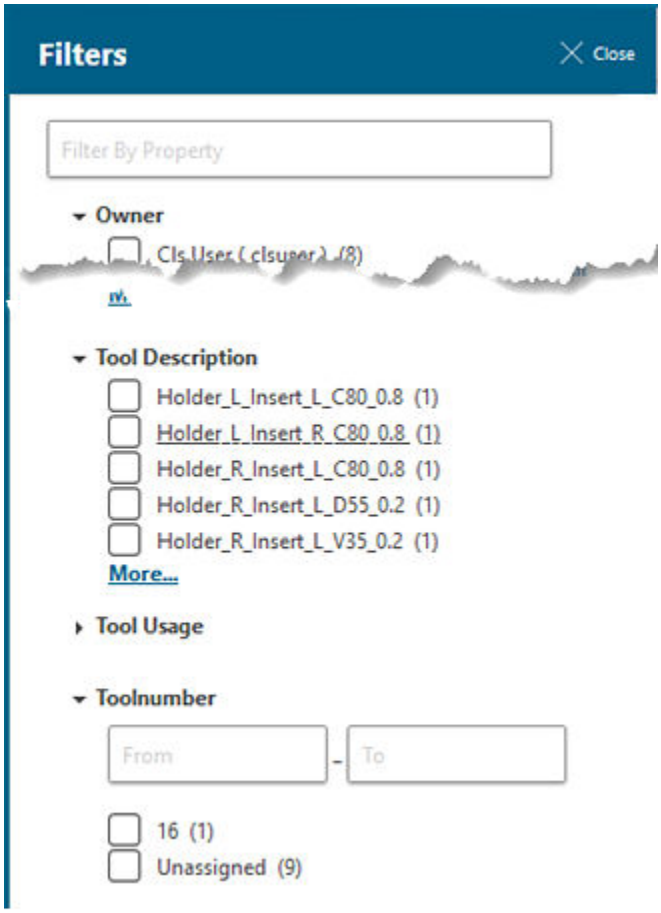
- Index all data.

  This can be a time-consuming process depending on the amount of data in your database.

- **Index only the workspace objects whose classification data has changed**.

Create search index views and choose attributes by which to search and filter

For traditional basic classification, the attributes shown in the facets in Active Workspace are governed by search index views created in rich client. Only the attributes marked for filtering in the search index view are available for searching in the facets.



1. In Teamcenter rich client Classification, verify that classified data is available. Classification should contain a structure of classes with attributes and classified objects.

   For more information see the *Teamcenter Classification* documentation.

2. Choose one of the following to define the classification attributes (properties) you want to index.

> Note:
>
> It is highly recommended that you add only those attributes that you truly require for searching to the search index views. Adding all attributes creates a very long list of filters in Active Workspace that can impede efficiency.

- To select all classes, attributes, and filters, run the **smlutility** utility. For example, from a Teamcenter command window, type:

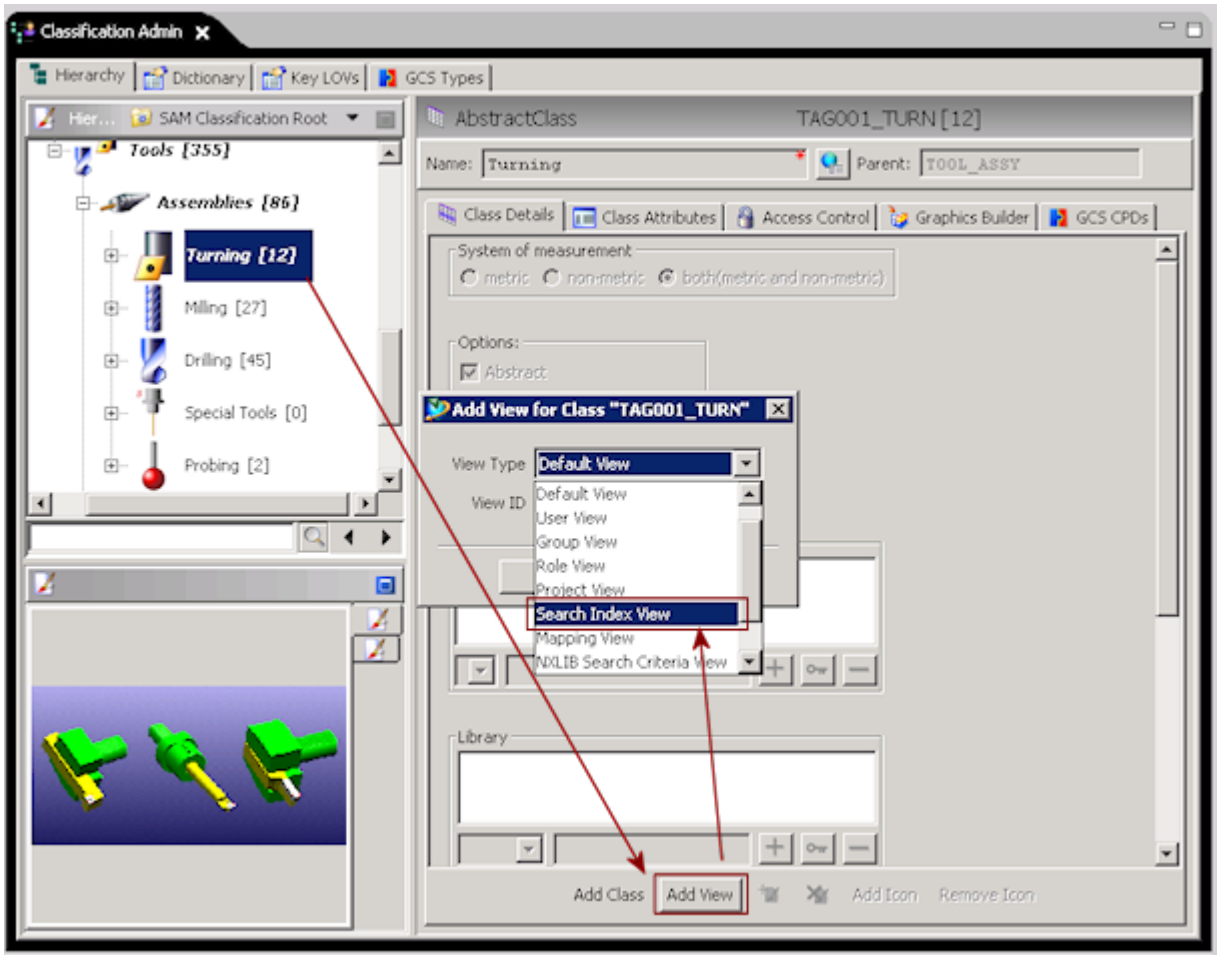
  ```
  smlutility -create_indexing_views -u=username -p=password -g=dba
  ```

  A search index view is created for each class in Classification Admin. All attributes of the class are assigned to the view. You can only run this command once for a class.

- Specify individual classification attributes:

  a. Ensure that the **ICS_searchindex_view_visible** preference to **TRUE**.

  b. In the Teamcenter rich client **Classification Admin**, the **Add View** operation now includes the **Search Index View** option.



Class attributes for these classes are not yet configured for indexing or filtering.

For a class to be available for indexing and filtering, the class must contain a search index view, at least one attribute, and at least one classified object.

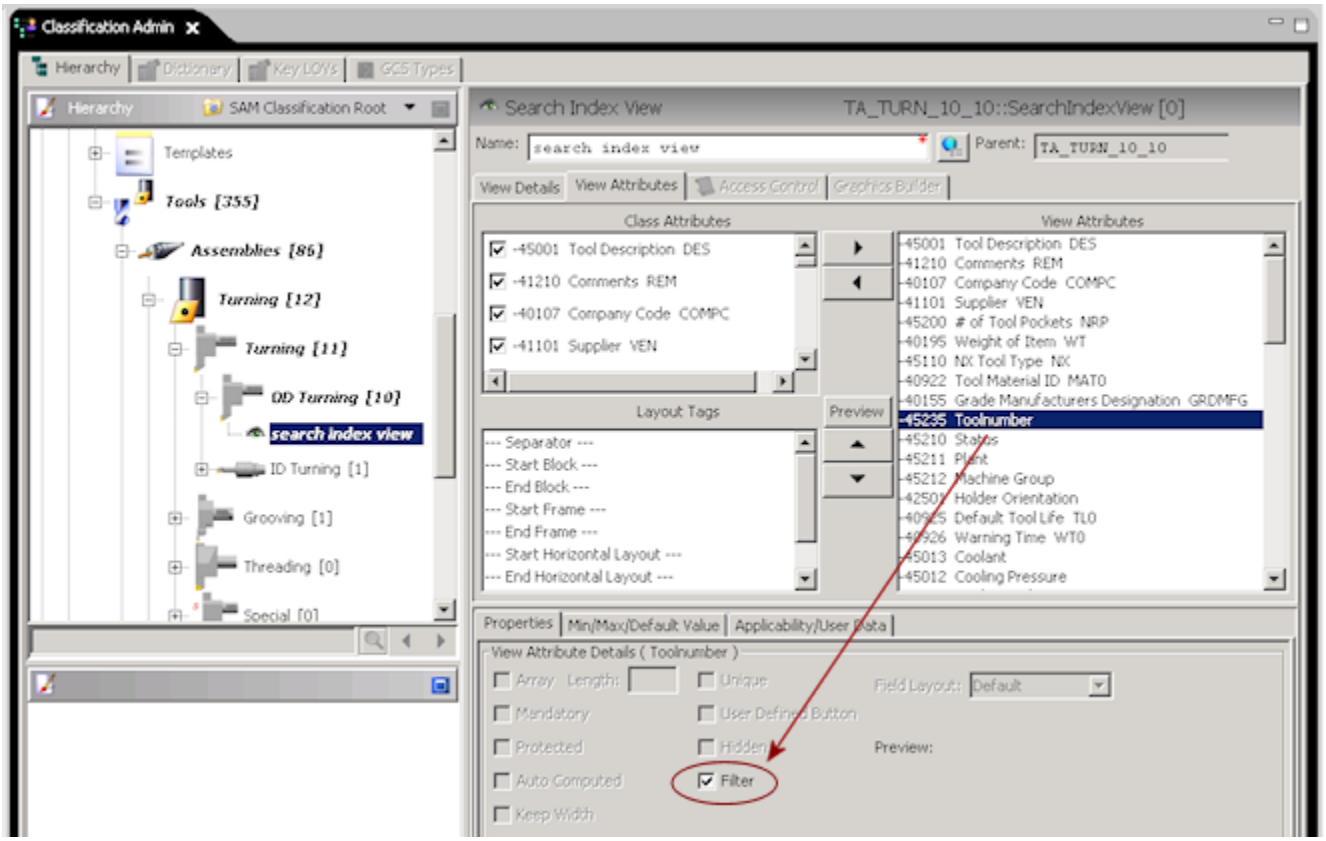
c. In Teamcenter rich client Classification Admin, select the search index view for a class and select the **View Attributes** tab.

d. Add all attributes that you want included in the indexing to the **View Attributes** list.

You cannot add reference attributes to search index views, since these attributes can be directly searched via the object attributes they are linked to.

e. Individually, in the **View Attributes** list: select an attribute, choose **Edit Current Instance** , select the **Filter** check box, and click **Save Current Instance**.

The **Filter** property indicates that the attribute is to be available in the **Filter** list in Active Workspace.



3. *Make properties searchable in the global search*

4. *About indexing classification data*

How views limit visible search filters

Search filters are available for all the attributes that are included in a search index view of a class and for which the Filter option is selected. These are the properties that are indexed for a workspace object classified in that class. However, it is possible to add another layer of restrictions. Setting up views allows a classification administrator to limit the visible attributes to a subset of the available attributes in a class based on user, group, or role. These views influence which filters are displayed in the **Filter** pane in the following fashion.

The classification hierarchy contains the following classes and views:

Class A:

Properties	Contained in Search Index view and Filter option selected	Contained in user, group, or role view
Prop1	✓	✓
Prop2	✓	✓
Prop3	✓	X

Class B:

Properties	Contained in Search Index view and Filter option selected	Contained in user, group, or role view
Prop3	✓	✓
Prop4	✓	✓
Prop5	✓	X
Prop6	X	✓

The following scenarios describe the behavior when searching the classification hierarchy.

Scenario 1 - Navigate to Class A and search only in that class:

When you search in **Class A**, the following properties are displayed in the search filters:

Prop1

Prop2

Prop 3 is not available as a filter because it is not included in a view created for the class.

Scenario 2 - Navigate to Class B and search only in that class:

When you search in **Class B**, the following properties are displayed in the search filters:

Prop3

Prop4

Prop 5 is not available as a filter because it is not included in a view created for the class. **Prop6** is not displayed as it is not indexed.

Scenario 3 - Perform a search of the entire hierarchy using the * wildcard character:

When you search the entire hierarchy, the following properties are displayed in the search filters:

Prop1

Prop2

Prop4

Prop3 and **Prop5** are not available as filters because they are restricted in views. **Prop6** is not displayed as it is not indexed.

If you search the entire class hierarchy, any property that is restricted in any class view is not available for filtering. If you navigate to a class where the property is not restricted by a view, the property becomes available in the list of filters.

Note:

- For traditional basic classification classes, if the **Filter** box is not selected for a particular attribute in a class, then it is not displayed in the **Filters** pane even if it is included in the group, user, or role view for a class.
- If you use advanced classes, all properties are indexed and their visibility when searching is determined solely by their inclusion in user, group, or role views.

Make properties searchable in the global search

By default when you add properties to a class, they are displayed as searchable filters. If, additionally, to search for properties in the global search, or when indexing classification data to mark the changed classification objects, you must do the following:

1. Run the **bmide_modeltool** utility.

   For example, in a Teamcenter command window, change to the *TC_ROOT***bin** directory and type:

   ```
   bmide_modeltool -u=username -p=password -g=dba -tool=all
   -mode=upgrade
   -target_dir="TC_DATA"
   ```

2. Verify the schema file was correctly updated.

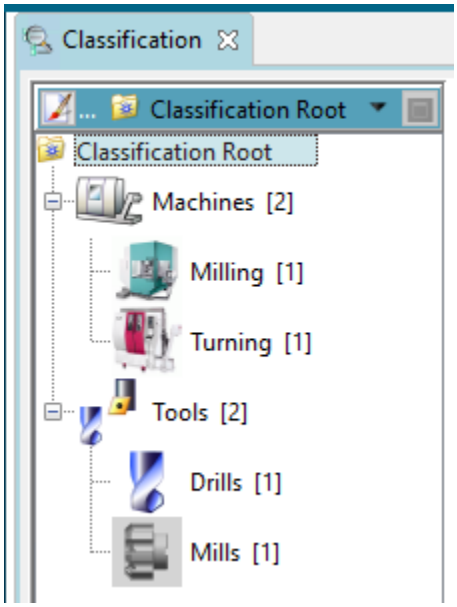
   a. Verify that the **tc_solr_schema.xml** file in the *TC_DATA***ftsi\solr_schema_files** folder has the current date and time.

   b. Open the **tc_solr_schema.xml** file in a text editor and search for **TC_0Y0_CLS_0Y0**. You should find several lines containing this string. Close the text editor without saving.

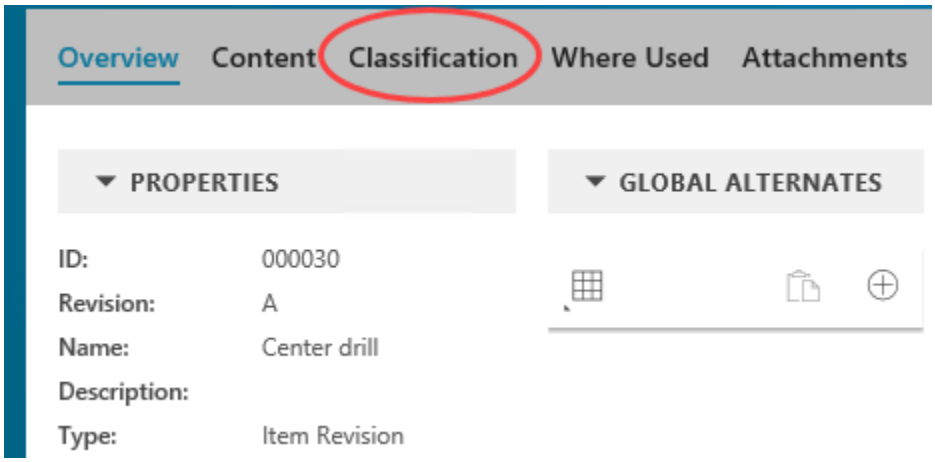
Configuring search in Active Workspace when using traditional basic classification—Example

This example demonstrates how to configure the search in Active Workspace when using traditional basic classification.

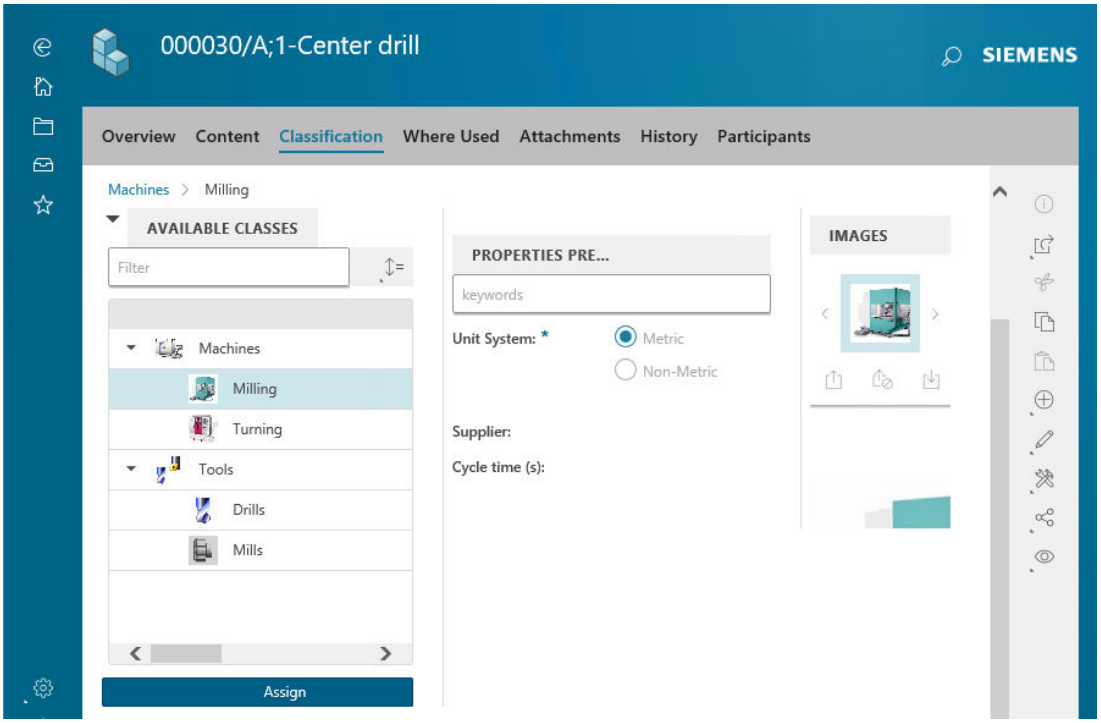
In rich client, there is the following classification hierarchy containing two abstract classes (**Machines** and **Tools**) and each of those classes contains two storage classes. Each storage class contains one classified object.



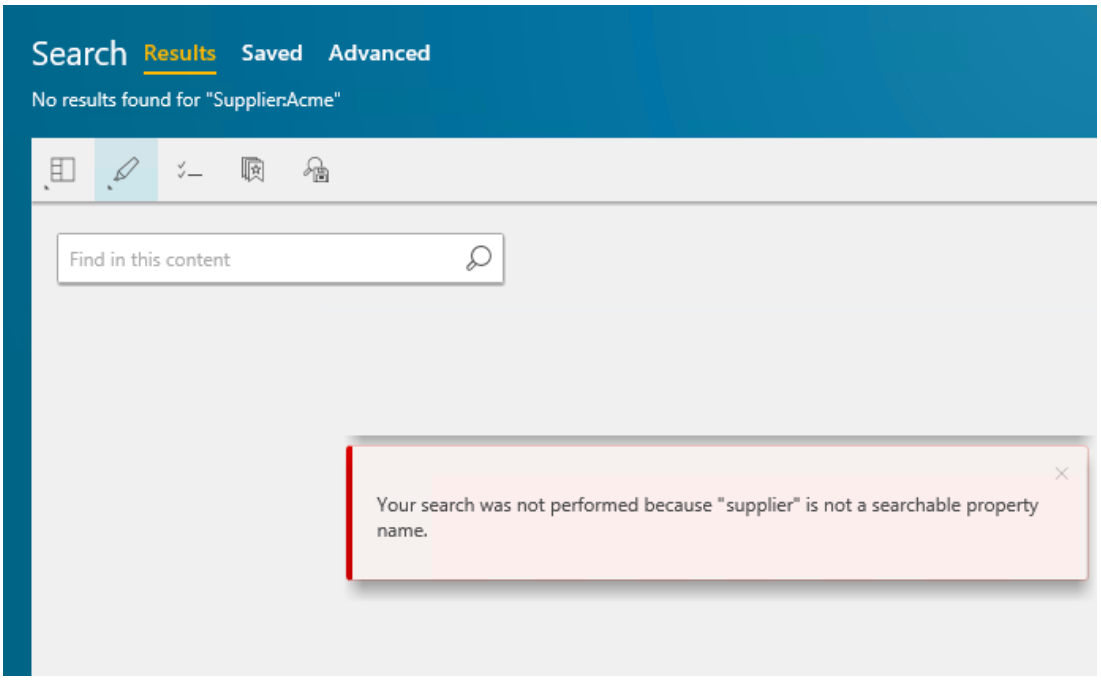
After installing the client and server components of Active Workspace, the **Classification** tab is now visible:



The storage classes are available for classification:

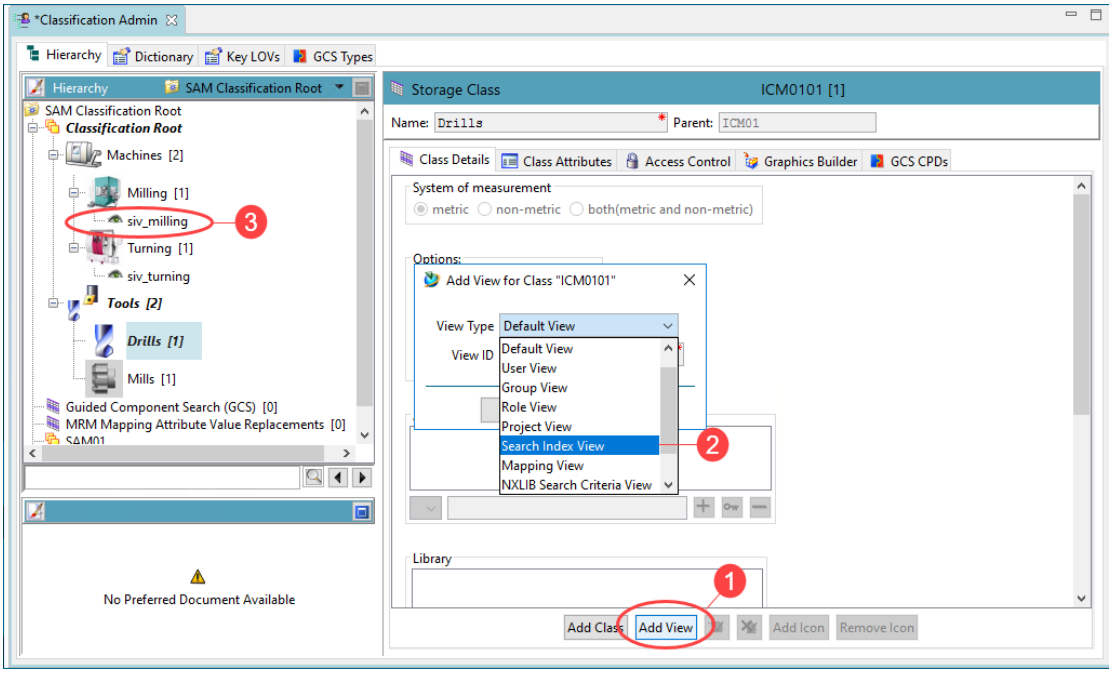


However, trying to search for a machine that is classified using one of its properties returns no search results:

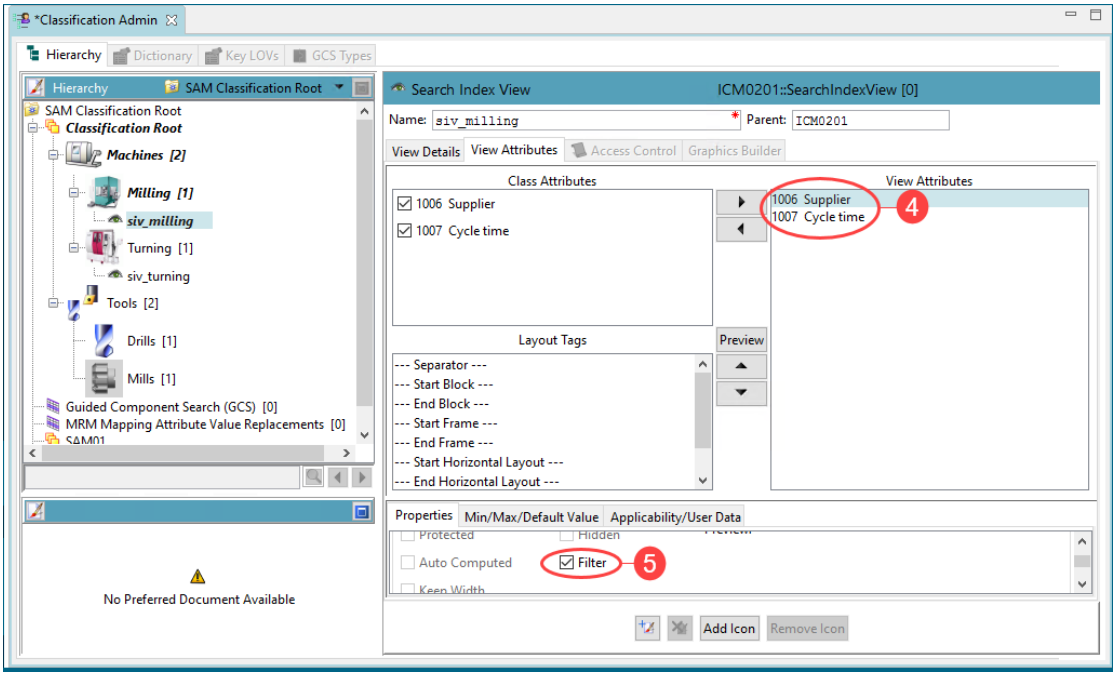


To enable searching:

1. Add **Search index** views to the classes in rich client:



2. For each property (4), select the **Filter** attribute (5) and save the view.



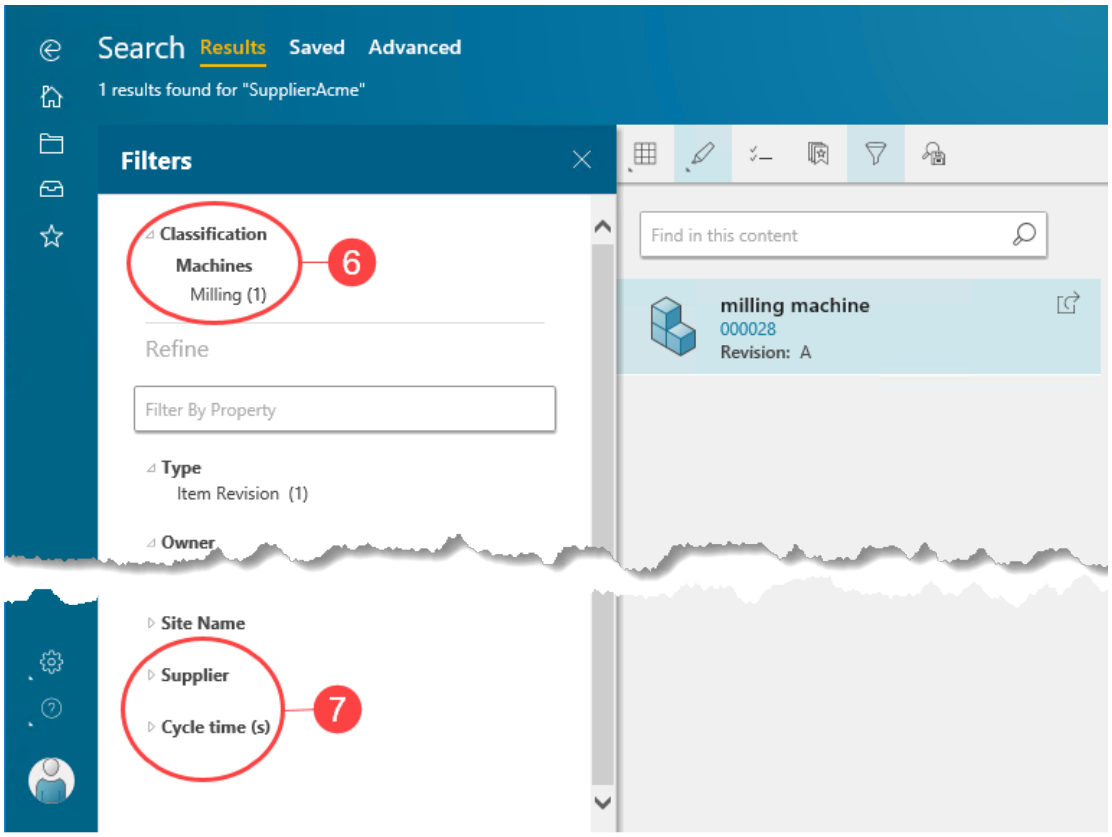
Note:

This example works through a simple use case. When performing these activities for production data, use the **smlutility** utility to create the **Search index** views.

Because the hierarchy has changed, the facets must be updated.

3. **Run the bmide_modeltool utility to update the facets**, and **re-index the data**.

Now, when you search for the **Supplier** property in Active Workspace, the data is found. The classification class 6 and the properties 7 are displayed in the **Filter** list in Active Workspace:



Configuring the search similar feature

The *search similar* finds the filter criteria for objects and presents them. By modifying these facets, you can find other objects with similar properties.

There are two types of properties that you can enable for the search similar feature:

- Workspace object properties

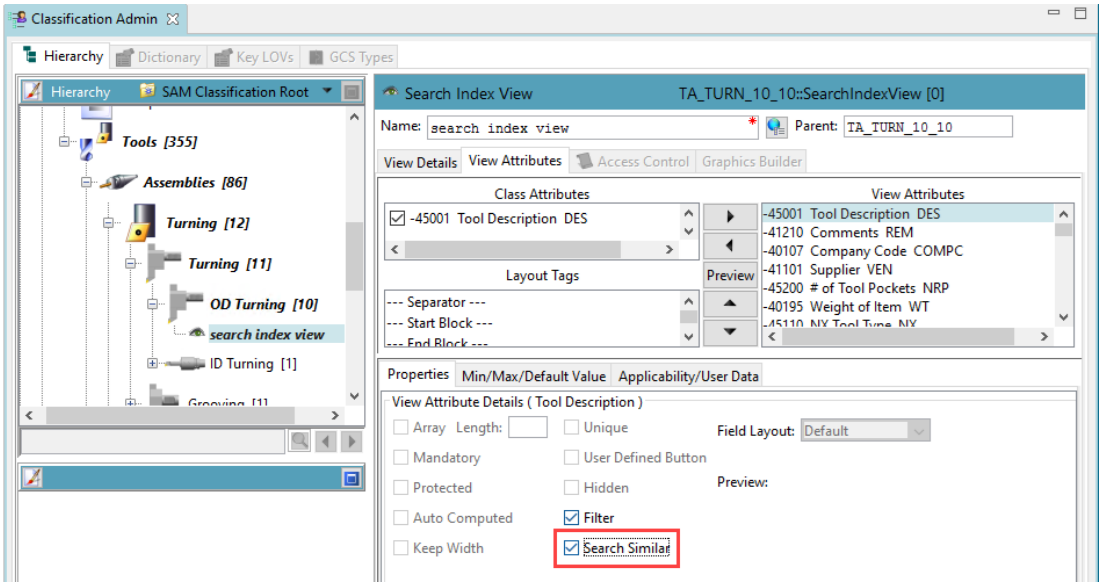
  The **CLS_search_similar_wso_props_enabled** preference contains a list of workspace object properties that are used to find similar objects.

  When adding new properties to this preference, only properties that are listed in the **AWS_FullTextSearch_FacetFilters_ObjectData** preference are valid entries. Use the values in this preference to obtain the correct syntax of the property entries.

- Classification properties

  Classification properties for the search similar feature are enabled using any of the following methods:

  - In rich client, in the **Search index view** of a class, select the **Search Similar** option for each desired attribute.



  - Run the **smlutility** utility:

    smlutility -update_indexing_views -searchSimilar *attribute-IDs class-IDs*

    For example:

    - To enable the search similar feature for all properties in all classes, type the following:

      smlutility -update_indexing_views -searchSimilar

      This enables the **Search Similar** option on all properties, in every **Search Index** view, that have the **Filter** option checked.

    - To enable or disable the search similar feature for selected attributes in every class containing a **Search Index** view, add the **-attrids=** argument. To disable an attribute, add an exclamation mark before the attribute ID:

      smlutility -update_indexing_views -searchSimilar -attrids= attr1, !attr2,attr3

    - To enable or disable the search similar feature for selected classes, add the **-classids=** argument.

smlutility -update_indexing_views -searchSimilar -attrids= attr1, !attr2,attr3 classids=class1,class2

- Import the configuration using a JSON file:

  smlutility -update_indexing_views -searchSimilar -configFile=*JSON-file-name*

  The JSON file must have the following format:

```
{
  "SchemaVersion": "1.0.0",
  "Classes": [
        {
            "ID": "classA",
            "Enabled": [ "1" ],
            "Disabled": [ "2", "3" ]
        },
        {
            "ID": "classB",
            "Enabled": [ "*" ],
            "Disabled": [ "2" ]
        },

    ],
  "Attributes": {
    "Enabled": [ "4", "5" ],
    "Disabled": [ "6" ]
  }
}
```

  This code does the following:

  - Enables attribute **1** in **classA** and all attributes in **classB**.
  - Disables attribute **2** in both **classA** and **classB** and attribute **3** in **classA**.
  - Enables attributes **4** and **5** in all classes containing a **Search Index** view and having the **Filter** option set on both of these attributes.
  - Disables attribute **6** in all classes.

  For more information about the syntax of the JSON file, see the following schema files

  ...*TC_DATA***\classification\json\schema\basic\SearchSimilar_basic.schema.json**

Searching using localized classification values

Classification values are localized along with other objects using the same localization process.

The following features are fully supported for localization:

- Keyword search
- Property specific search
- Category name in filter panel
- Filter value
- Filtering using localized value

Re-index classification data after upgrading Teamcenter

After upgrading Teamcenter, or after installing any classification features, you must **re-index classification data**.

If the new release to which you are upgrading includes new classification features, it generally also includes schema changes. During this upgrade procedure all classification data is marked for re-indexing and included during your next indexing synchronization. This can cause the synchronization to take a longer than usual.

Classification business constants for search

Business object constant	Object type on which constant can be set	Default setting	Description
Awp0SearchIsIndexed	**Item** **ItemRevision** **Cls0ClassBase** **Lbr0LibraryElement**	Set to **true** on the **ItemRevision**	Enables a business object for indexing. This business object is configured in the **AWS2** template owned by the search. Although it is possible to set this constant to true on **Item** objects, it is recommended to index item revisions only. To enable classification presentation layer classification objects (**Cls0ClassBase**) and library elements (**Lbs0LibraryElement**) for indexing, you must install the **Cls1ClassificationAW** and **Lbr0LibraryManagementAW** templates which set this constant to **true** on **Cls0ClassBase** and **Lbr0LibraryElement** respectively.
Awp0SearchIsClassifyDataIndexed	Specific object type, for example: **Item**	Is set automatically to **true** when you install the **ics1icsaw** template.	This business object type constant is specific to classification and is set on a business object to enable its associated classification information for indexing.

Business object constant	Object type on which constant can be set	Default setting	Description
	ItemRevision **Cls0Object** **Lbr0LibraryElement**		If in an environment item revisions are classified and are also marked for indexing and if this business object type constant value is set to **true**, classification information associated to those item revisions are indexed. Administrators needs to install the **ics1icsaw** template to enable classification indexing which sets this BO constant to **true** on **workspaceObject**. All subtypes of **WorkspaceObject** (**Item**, **ItemRevision**, and so on) inherit this configuration.
Awp0SearchClassifySearchEnabled	**WorkspaceObject**	Installing the **ics1icsaw** template to enable classification data indexing, sets this business object type constant to **true** on **WorkspaceObject** and all its subtypes inherit this configuration.	This business object type constant is set to enable indexed classification properties for search. If this business object type constant is set to **false** and if the classification information is already indexed, it is not used for searching. Consequently, classification-specific filter categories are not displayed in the Active Workspace client.

Configuring the classification interface

Adding images and icons to a classification class or hierarchy

A visual representation of classes helps to quickly navigate the hierarchy or provides additional information about a class. In classification, you can add images and icons to a class.

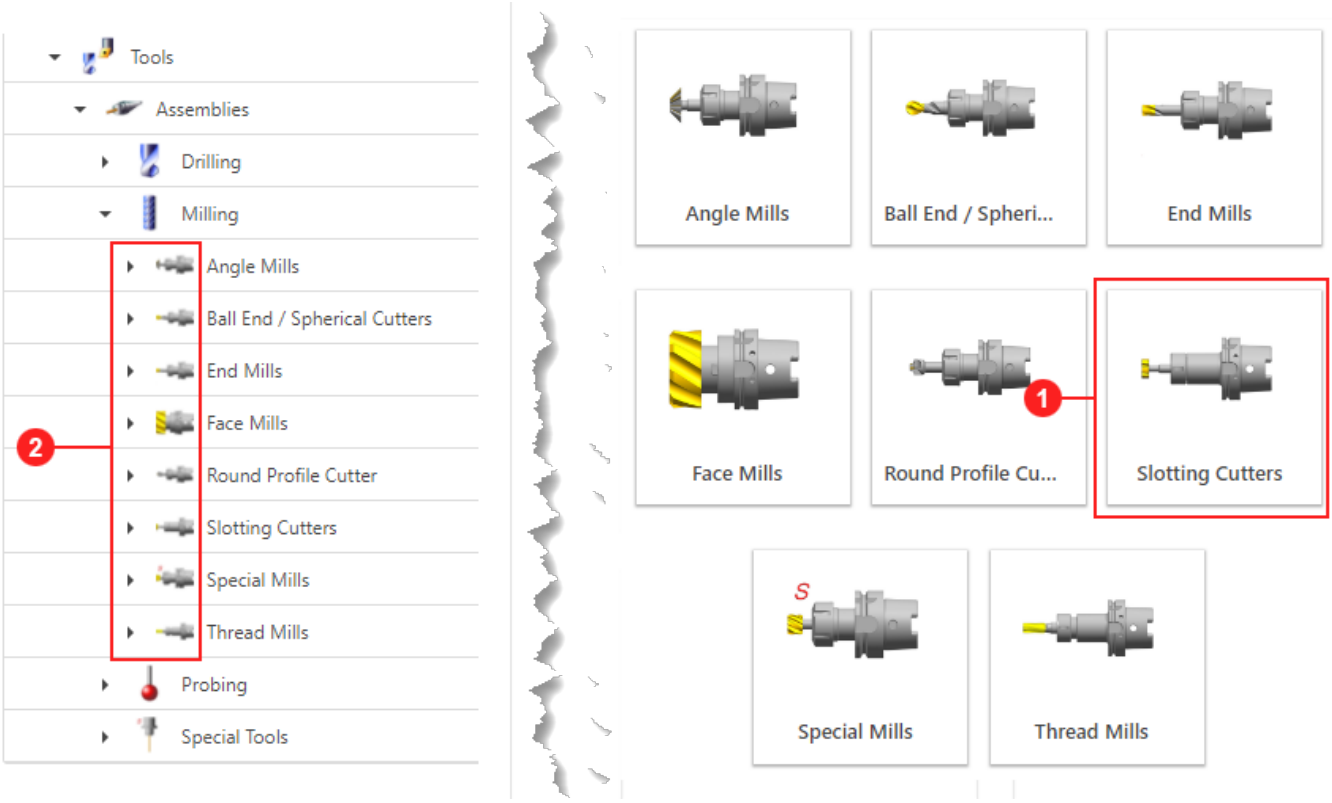
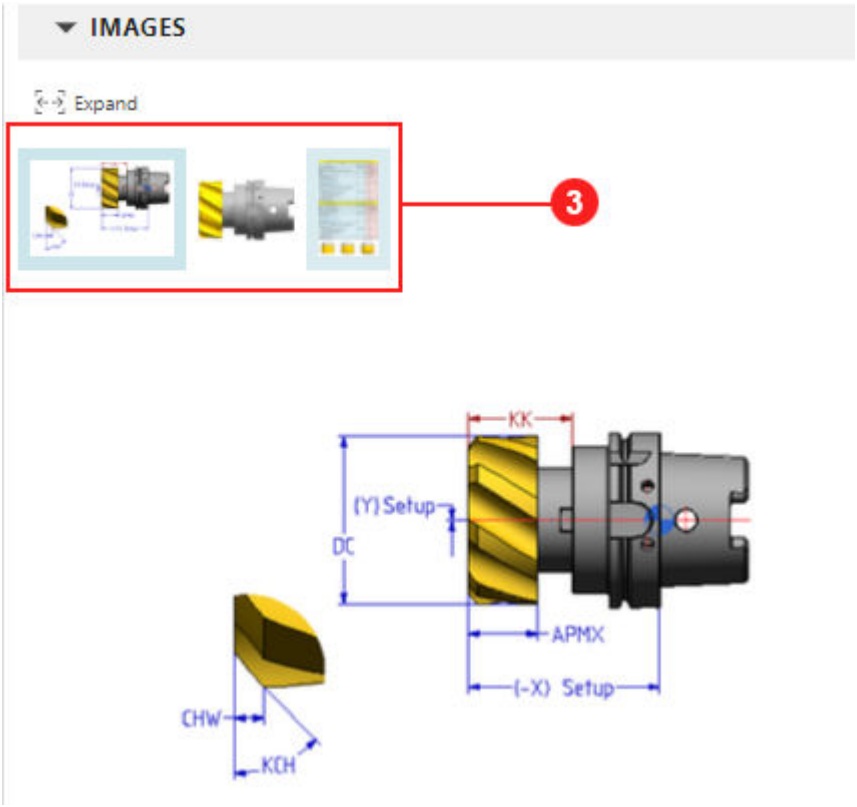


Image 1 Refers to an image associated with a class. A class can have multiple images associated. These are displayed in the **IMAGES** pane where you can switch between them using the carousel 3.



One image is designated as the primary image. This is the image that is displayed in the visual navigation card.

Icon 2 Refers to the small graphic displayed in the classification hierarchy when you navigate the hierarchy.

The behavior of images and icons varies depending on what type of classification you use.

- Traditional basic classification:

  Import images and icons using **smlutility** to automatically display them in the Active Workspace client. Alternatively, you can add images and icons manually in the rich client.

- Presentation layer:

  - Advanced classification (classification standard taxonomy) is not installed:

    Extend the data to the presentation layer to automatically display icons and images.

    If the presentation hierarchy is installed, you can import icons and images with **clsutility**. If you import a node icon, it overwrites an icon that is specified in traditional basic classification. If you import an image, it is displayed along with the class images that are configured with traditional classification.

Add images and an icon to a node

Adding images or an icon to a node provides visual points of reference within the classification hierarchy. Images help to identify the contents of a node. You can add multiple images to a node which then appear in the **Preview** section within the **Overview**.

Procedure

1. On the **Classification Manager** dashboard, click **Nodes**.
2. Select a node to which you want to add an image or an icon and then select **Attachments**.
3. In the **Files** section, click **Add** ⊕ and click any one of the following:
   - **Add Icon** ⊕ to add an icon in the selected node.
   - **Add Image** ⊕ to add an image in the selected node.
4. In the **Add Icon** or **Add Image** panel, click **Choose File** to select the required file, and click **Add** to add an icon or an image to the node.

   When you add an image as an attachment to a node, a preview of the image appears in the **Preview** section of the **Overview**. For a given node, you can include multiple images by adding them one at a time.

   When you add an icon as an attachment to a node, it is displayed in the visual navigation card of the node tree. You can add one icon at a time. To replace an icon, you must first delete the previously added icon.
5. Click **List** or **Images** to view all the images and icon added to the node.

Delete an image or an icon from a node

You can delete an image or icon from a node.

Procedure

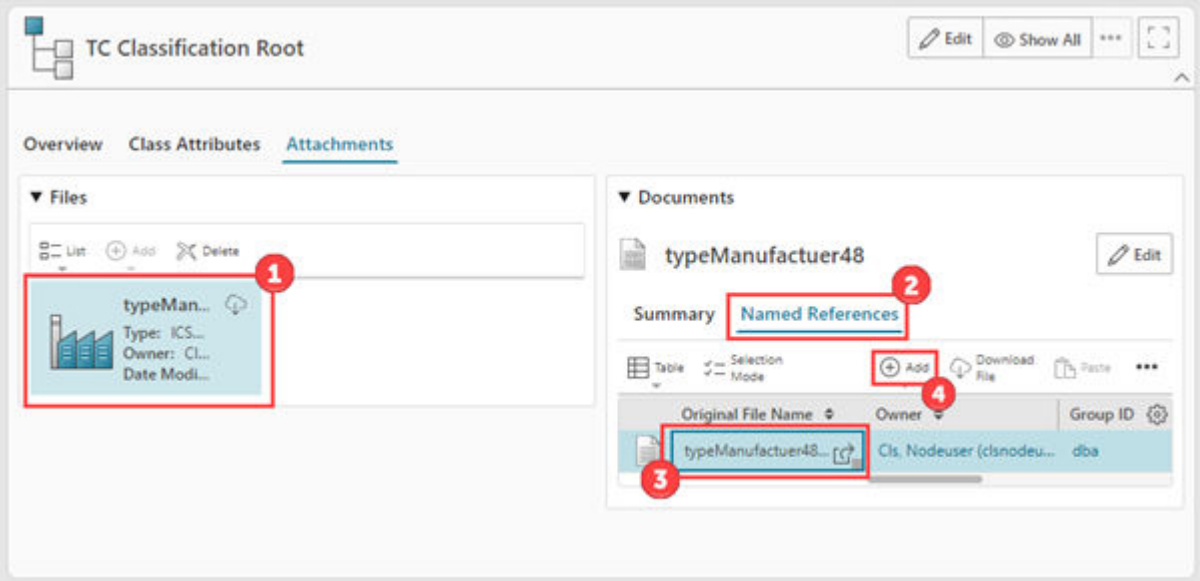
1. On the **Classification Manager** dashboard, click **Nodes**.
2. Select a node to from which you want to delete the image or icon.
3. Under **Files**, select the image or icon and click **Delete** ✕.
4. In the confirmation message, click **Delete**.

Add images and an icon to a class

Adding images or an icon to a class provides visual points of reference within the classification hierarchy. Images help to identify the contents of a class. You can add multiple images to a class which then appear in the **Preview** section within the **Overview**. You can add images or an icon to a basic class only.

Procedure

1. On the **Classification Manager** dashboard, click **Classes**.
2. Select a class to which you want to add an image or an icon and select **Attachments**.
3. In the **Files** section, click **Add** ⊕ and click any one of the following:
   - **Add Icon** ⊕ to add an icon dataset in the selected class.
   - **Add Image** ⊕ to add an image dataset in the selected class.
4. In the **Add Icon** or **Add Image** panel, click **Choose File** to select the required file.
5. Specify the **Name** for the dataset, and click **Add** to add an icon or an image dataset to the node.
6. Select the dataset, and in **Documents**, click**Named References**.
7. Select the dataset, click **Add** ⊕ and click any one of the following:
   - **Add Icon** ⊕ to add an icon in the selected dataset.
   - **Add Image** ⊕ to add an image in the selected d.

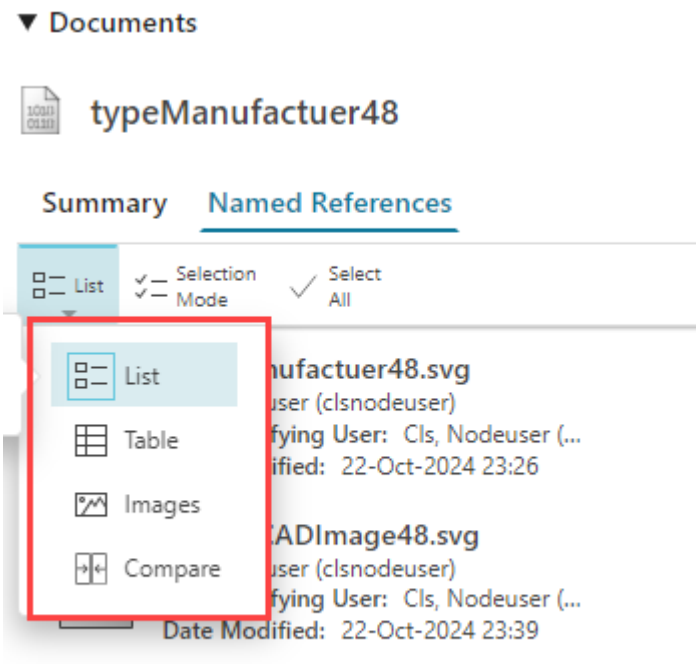


8. In the **Add Icon** or **Add Image** panel, click **Choose File** to select the required file and click the **Add** button to add an icon or image to the class.

   When you add an image as an attachment to a class, a preview of the image appears in the **Preview** section of the **Overview**. For a given class, you can include multiple images by adding them one at a time.

   When you add an icon as an attachment to a class, a preview of the image appears in the **Preview** section of the **Overview** and it is also displayed in the visual navigation card of the class tree. You can add one icon at a time. To replace an icon, you must first delete the previously added icon.

9. Click **List**, **Table**,**Images**, or **Compare** to view all the images and icon added to the class in the selected view.



Delete an image or an icon from a class

You can delete an image or icon dataset as well as an image or and icon file from a class.

Procedure

1. On the **Classification Manager** dashboard, click **Classes**.
2. Select a class from which you want to delete an image or an icon and select **Attachments**.
3. To delete an icon or an image, do the following:
    a. In the **Documents** section, under **Named References**, select the image or icon to be deleted and click **Delete** ✕.
    b. In the confirmation message, click **Delete**.
4. To delete an icon or an image dataset, do the following:
    a. In the **Files** section, select the image or an icon dataset to be deleted and click **Delete** ✕.
    b. In the confirmation message, click **Delete**.

The images and icon associated with the dataset will also be deleted from the **Documents** section.

Changing the default icon of visual navigation cards

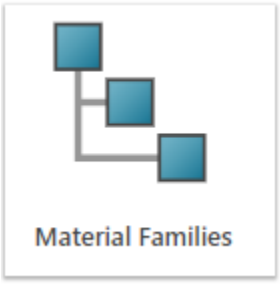
In the Class Navigator you can view the classification hierarchy by image. These tile images are referred to as *visual navigation cards* (VNCs) and allow the display of larger images while viewing the selected level in the classification node hierarchy.

VNCs are visible in the following areas:

- The Classification location when you click on the CLASSIFICATION tile on the home page
- The **Classification** tab where you classify objects

Configuring the image displayed on the VNC

If you have not yet imported node icons, a generic VNC is displayed listing the children of the node.

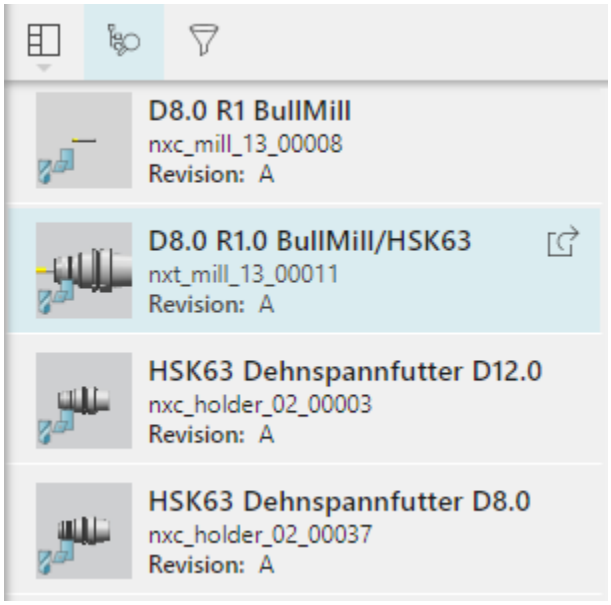


The images are based on the node icons that you import using **clsutility -import -icon -id=***hierarchy-node-ID*. If a node icon does not exist for a class, there is a fallback mechanism that searches for the first image of a node, if none exists, then the first class image is used.

> Tip:
>
> To make optimal use of the image space on the VNC, create a square image. Rectangular images are cropped.

Configuring what information appears in the classification tiles

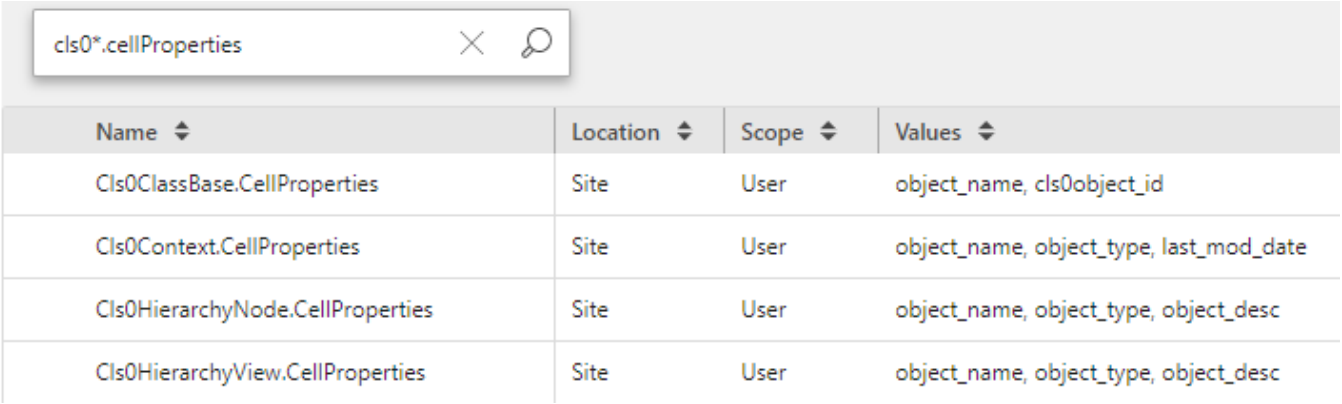
Tiles are displayed in the primary work area, for example, in search results:



Configuration of the object tiles respects the type hierarchy.

Mdl0ModelElement		
	Cls0ClassBase	
		Cls0CstObject
		Cls0Object

Tile properties are configurable on the parent class or on the child classes themselves. Site preferences determine what is displayed on the tiles for **Cls0Object** and **Cls0CstObject**, for example:



Name	Location	Scope	Values
Cls0ClassBase.CellProperties	Site	User	object_name, cls0object_id
Cls0Context.CellProperties	Site	User	object_name, object_type, last_mod_date
Cls0HierarchyNode.CellProperties	Site	User	object_name, object_type, object_desc
Cls0HierarchyView.CellProperties	Site	User	object_name, object_type, object_desc

Enable the display of DWG and CGM class images in the Classification tab

To use CGM or DWG file formats as class graphics, you must first convert them to PDF using the **prepare** utility delivered with the Lifecycle Visualization installation.

1. Run the Lifecycle Visualization installation, and select the **Convert and Print** option.

2. Copy the CGM or DWG file to the **VVCP** directory in the installation location.

3. Double click the **prepare.exe** utility.

4. Run the utility to convert the **CGM** or **DWG** files to PDF. For example, for a file named **my_drawing.dwg**, enter the following:

   ```
   prepare -pdf my_drawing.dwg
   ```

   This converts the **my_drawing.dwg** file and creates the **my_drawing.pdf** file in the **VVCP** directory.

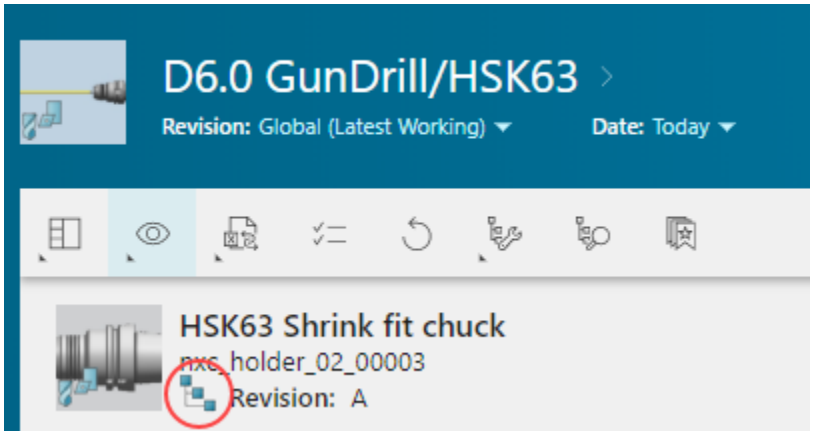
   You can add path names to change the input and output directories.

   Run `prepare -h` to obtain information about the utility.

5. Associate the **PDF** to the required class in Classification Admin (rich client).

Configure a visual indicator for classification on an object

You can set a visual indicator on an object to display that it is classified:



Add the following to your configuration code:

```
"classiBOM": {
            "iconName": "Classified",
            "tooltip": {
                "showPropDisplayName": false,
                "propNames": ["ics_classified"]
            },
            "modelTypes": ["Awb0DesignElement"],
            "prop": {
```

```
            "names": ["awb0UnderlyingObject"],
            "type": {
                "names": ["ItemRevision"],
                "prop": {
                    "names": ["ics_classified"],
                    "conditions": {
                        "ics_classified": {
                            "$eq": "YES"
                        }
                    }
                }
            }
        }
    }
```

Configuring views in traditional basic classification

In the rich client, traditional basic classification classes often have views that provide layout options for class attributes. These views are visible in Active Workspace. Any blocks created in the rich client are displayed as property groups in Active Workspace. You can use these property groups to quickly navigate large numbers of properties in a class.

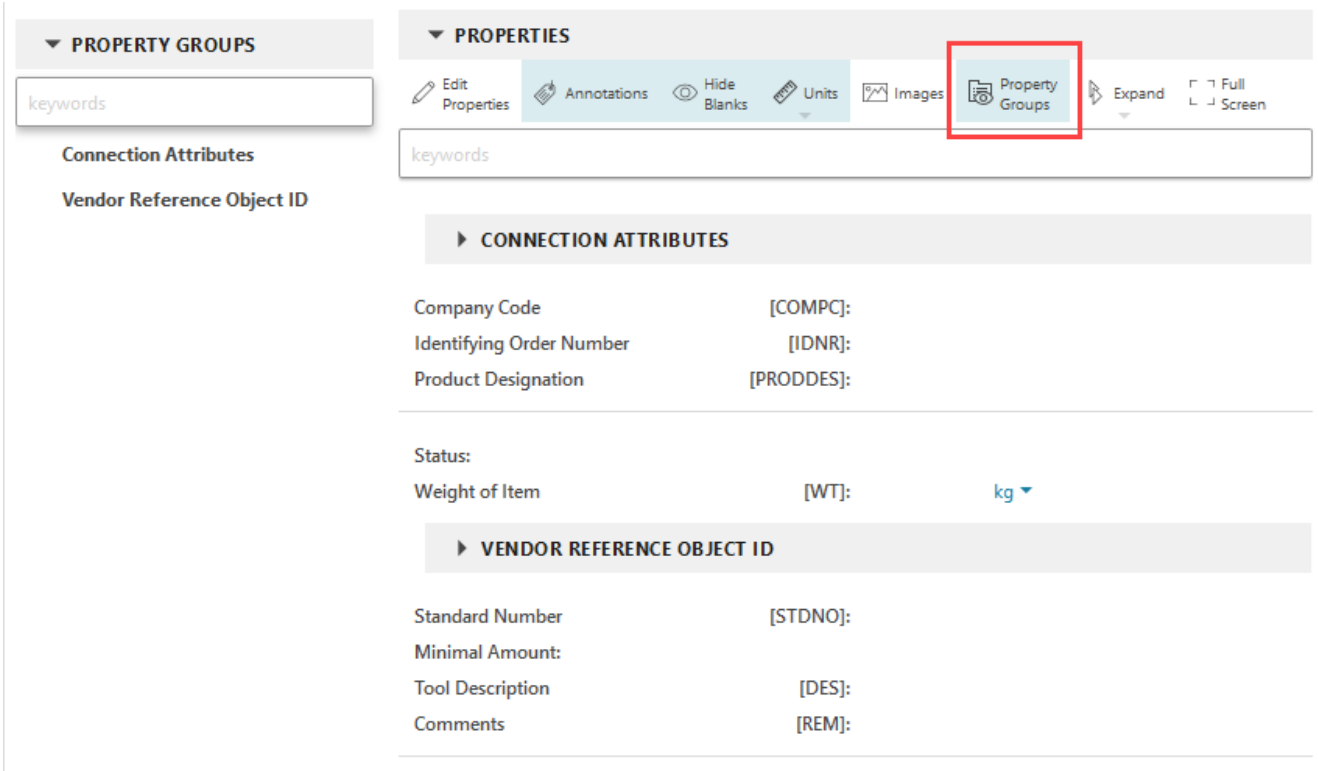
The following compares a view created in the rich client with its display in Active Workspace.

Rich client:

View Attributes

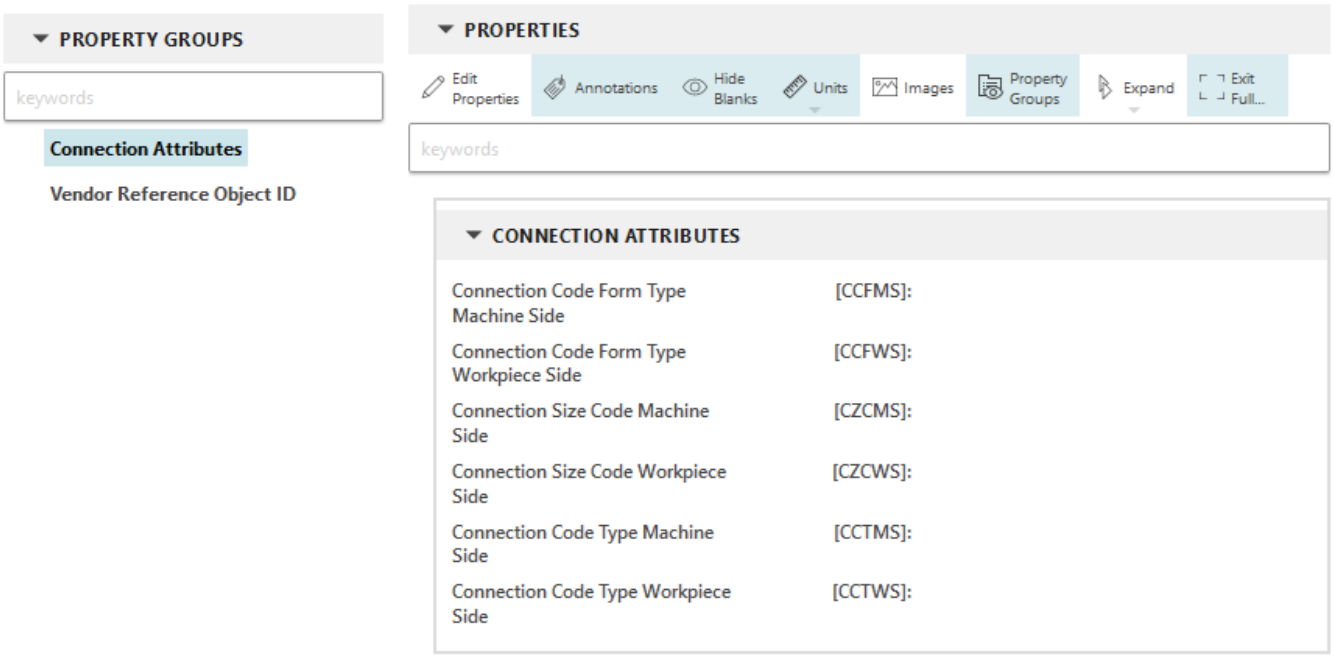
--- **Start Frame (Connection Attributes)** ---
-40036 Connection Code Form Type Machine Side CCFMS
-40037 Connection Code Form Type Workpiece Side CCFWS
-40039 Connection Size Code Machine Side CZCMS
-40040 Connection Size Code Workpiece Side CZCWS
-40041 Connection Code Type Machine Side CCTMS
-40042 Connection Code Type Workpiece Side CCTWS
--- **End Frame** ---
-40107 Company Code COMPC
-40166 Identifying Order Number IDNR
-40219 Product Designation PRODDES
--- **Separator** ---
-45210 Status
-40195 Weight of Item WT
--- **Start Block** ---
-40930 Vendor Reference Object ID VEN0
-40931 Vendor Reference Class ID VEN1
-40932 Vendor Reference Date VEN2
--- **End Block** ---
-40209 Standard Number STDNO
-43150 Minimal Amount
-45001 Tool Description DES
-41210 Comments REM
--- **Separator** ---

Active Workspace:



Attributes not contained within blocks or frames are displayed outside of the property groups. As a block does not have a name associated with it in the rich client, the name of the first attribute within the block is used as the name of the property group.

With a property group selected:



The following rich client layout tags are not supported in Active Workspace:

Start Horizontal Layout

End Horizontal Layout

Start Column Layout

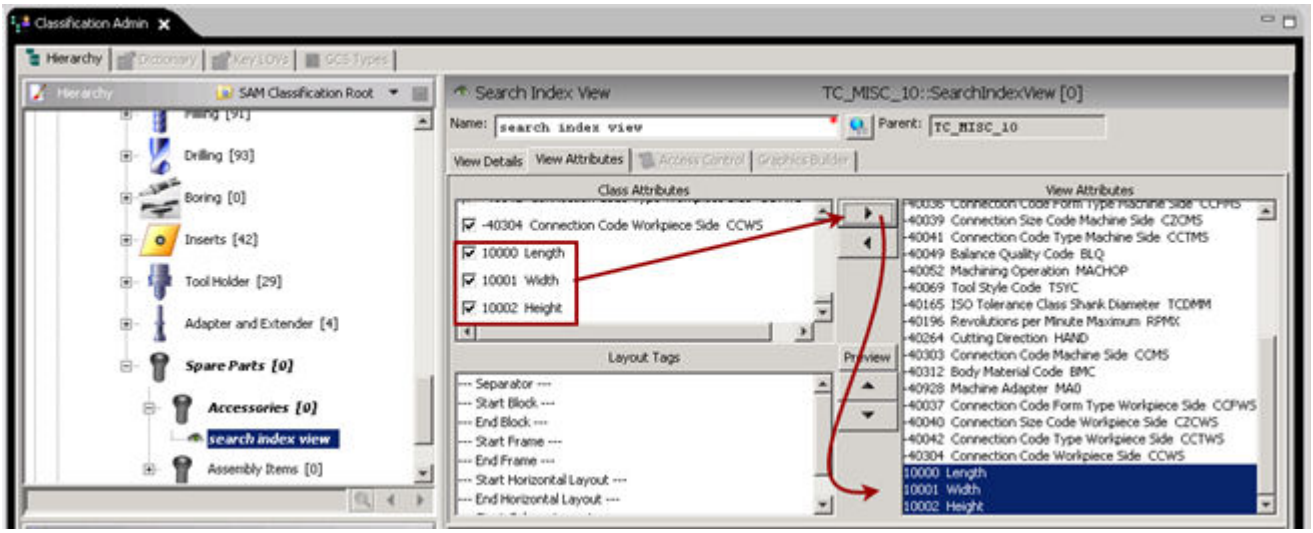
End Column Layout

Configuring interchangeable attributes

Attributes are referred to as *interchangeable* if you can search for the value of one and the results list other attributes in the group. For example, if **length**, **width**, and **height** are set to **Interchangeable**, you can search for **length**=10 and the results return all objects with **length**=10, **width**=10, and **height**=10.

To specify that attributes are interchangeable:

1. Add the desired attributes to the **search_index_view** in Classification Admin.



2. Select the **Filter** option for each attribute.

3. Add the attribute IDs to the **AWS_cls_attribute_interchangeability_definition** preference.

   For example, to define two attributes whose IDs are **10000** and **10001** as interchangeable, set the preference value as follows:

   10000:10001

   10001:10000

   To define three attributes as interchangeable whose IDs are **10000**, **10001**, and **10002**, set the following value:

   10000:10001, 10002

   10001:10000, 10002

   10002:10000, 10001

4. Re-index the search data.

Creating graphics for an object based on class templates

There are two ways in which Active Workspace can automatically create part files and JT graphics for a classification object:

- Based on part family templates

Part family templates are used in NX to define a set or *family* of parts that share similar form, fit, and function but differ based on parameter values (for example, length, width, or diameter) that typically control the physical characteristics of the part (or tool). The part families are created with the help of a Microsoft Excel file that holds a list of all *part family members*.

- Based on template parts

  Any NX part can be used as a template part.

Both part family templates and template parts contain expressions that describe a part parametrically. For example, **L1** represents the length of a drill. If you change the value of **L1**, you can quickly create many drills (*part family members* or *member parts*) of different lengths. Although the behavior in Active Workspace when using part family templates or template parts is very similar, the mechanics of how graphics are created for the members in the background varies.

> Note:
>
> Revisioning is supported with the template part method only.

An administrator must perform the following steps in rich client to configure graphics building for classification objects:

1. Associate the part family template or template part with a specific class.

2. Map the template expressions to class attributes.

A user creates a new classification object by entering attribute values. When the user starts the process to create graphics, Active Workspace starts the graphics builder executable that communicates with NX in the background and generates a new part family member or member part using the new attribute values. The graphics builder executable also creates a 3D model and, optionally, a JT file. These are stored in the database, and the JT file is displayed in the viewer

To create graphics based on a part family template, a user must have write privileges to the template. This is not the case for creating graphics based on template parts.

> Note:
>
> To create graphics based on a standalone classification object, you must **install the presentation layer** and **extend the data**.

Configuring classification enforcement

If your business process requires that all objects of a specific type, for example **Part** or **PartRevision**, must be classified, you can enforce classification.

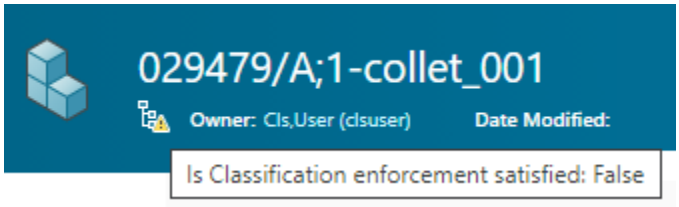
To enforce classification:

- List the internal type name in the **CLS_enforce_classification_on_types** preference.

  Example:

  Part or **PartRevision**.

When an object of this type is created, it is marked with a warning sign indicating that it has not yet been classified and a tooltip notifying that the enforcement requirement is not fulfilled.



To ensure that classification of these objects occurs, a user can submit the object to a workflow that uses the **Review Classifications** template and assign it to the responsible subject matter expert. The object to be classified is displayed as a target of this workflow and can be opened from within the workflow. After classification, the subject matter expert completes the workflow affirming that the object is correctly classified and the warning sign is removed from the classified object.

Configuring optimized unit values

You can display *optimized* attribute values in Active Workspace. An optimized attribute value is one that provides the most readable value with the least number of leading or trailing zeros. For example, 1 km is easier to read than 1000 m.

Attribute value entered	Optimized attribute value
1723000 µF	1.723 F
9640000 pF	9.64 µF
4372.3 Ω	4.3723 kΩ
3756794 mm	3.756794 km

Note:

The number of decimal places actually shown is dependent on the number specified in the **Number of decimal places** attribute in the unit definition.

The configuration of optimized values is applicable to traditional basic classification data only. Set the optimization at the class level in the **Class Attributes** pane in the rich client—the value of this attribute is optimized in this class only.

Note:

Active Workspace does not respect attribute optimization set in the attribute dictionary in the Teamcenter rich client.

Import pre-packaged classification hierarchy for ECAD, MCAD, and routing data for basic classification

The pre-packaged classification hierarchy for ECAD, MCAD, and routing data is available. The first storage class in this hierarchy has sample images. You can also use this classification hierarchy as a starting point or as a template when creating the classification hierarchy for other data. You can download this data from the following locations or search for the following ZIP files:

- *ECAD.zip*

  https://docs.sw.siemens.com/en-US/doc/282219420/PL20240523460057788.basic_class_deploy/xid2252446

  This file contains ECAD data and is derived from the Xpedition library to capture classification data for the ECAD domain.

  You can integrate this hierarchy with the Xpedition library following the regular integration best practices, using EDA Library Gateway.

- *MCAD.zip*

  https://docs.sw.siemens.com/en-US/doc/282219420/PL20240523460057788.basic_class_deploy/xid2252446

  This file contains MCAD data and is derived from the NX Machine Library to capture classification data for the MCAD domain.

  You can integrate this hierarchy with the NX Machine Library following the regular integration best practices, using the NX-Classification integration.

- *Routing_parts.zip*

  https://docs.sw.siemens.com/en-US/doc/282219420/PL20240523460057788.basic_class_deploy/xid2252446

  This file contains Routing Parts data and is derived from the NX Routing Parts Library to capture classification data for piping and tubing standard parts.

  You can integrate this hierarchy with the NX Routing Parts library following the regular integration best practices, using the NX-Classification integration.

Procedure

1. In the Active Admin workspace, click **Classification Manager**.
2. Click **Import**.
3. In the **Import** panel, select the **PLMXML** option.
4. Click **Choose File** and browse to the file that you want to import.

   > Note:
   > The file you are importing must be in the ZIP format.

5. From the **Transfer Mode** list, select **incremental_import**.
6. Click **Import**.

Results

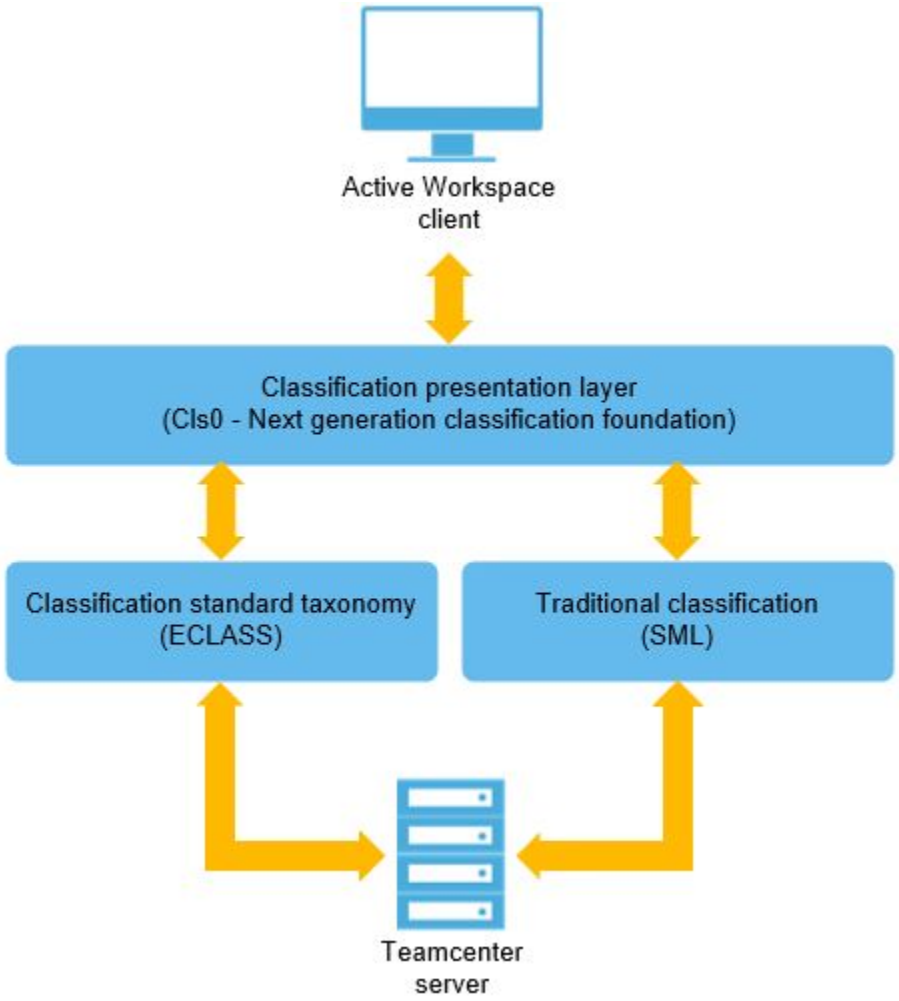
The files are imported and listed under the **Siemens samples** node.

About working with advanced and basic classification simultaneously

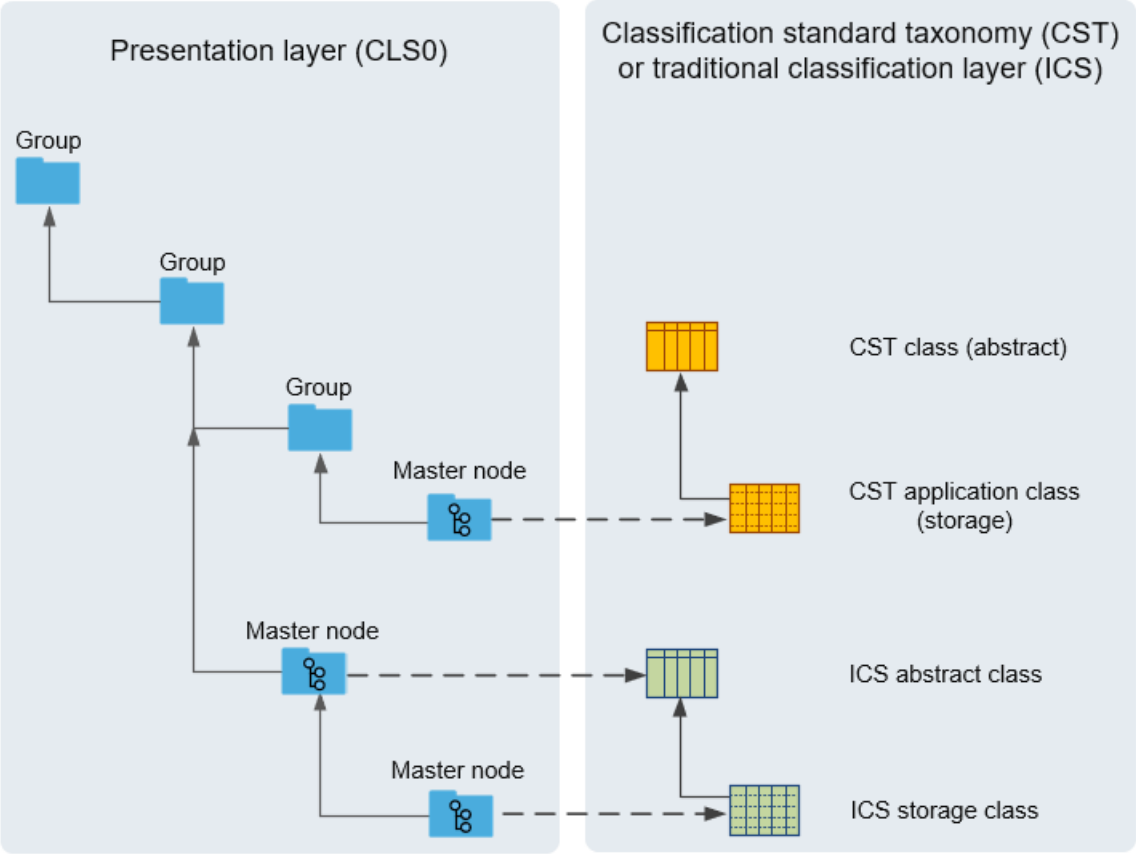
Classify with advanced and basic classification simultaneously

Advanced classification standard taxonomy (CST) offers some features that traditional basic classification does not. Additionally, in the future, CST will be regularly enhanced. For this reason, you may want to consider **migrating your traditional basic classification classes to the advanced CST format**.

Both advanced and basic classification classes can co-exist in a single environment.



In this case, the nodes reference the appropriate class type: For basic classification, a node references ICS classes and for advanced, it references CST classes.



The **CLS_is_presentation_hierarchy_active** user preference determines whether nodes or classes are displayed in the hierarchy. If this preference is set to **true**, only nodes are displayed in the hierarchy. If it is set to **false**, traditional classes are displayed.

> Note:
>
> The **CLS_is_presentation_hierarchy_active** preference is not delivered by default. You must add it manually.

Basic classification (ICS) classes must be extended to the presentation hierarchy as follows:

1. **Extend classes to the presentation layer using clsutility**.

2. **Synchronize the presentation layer with the classification hierarchy**.

Extend classes to the presentation layer using clsutility

1. Extend classification data to the presentation layer by running the following command line utility:

   clsutility **-import -hierarchy -cid=***group-or-class-ID*

This command extends the classification subhierarchy under the specified group or class.

- For classification groups, running this utility creates presentation layer group nodes with the same ID as the associated groups.
- For classification classes, running this utility creates presentation layer master nodes with the **storage_class_type** property corresponding to their associated classes.

Additional optional arguments:

- **-include_instances**

  Includes ICOs in the subhierarchy.

- **-exclude_children**

  Excludes the subhierarchy. It extends the specified group or class only.

2. (Optional) **Synchronize the presentation layer with the classification hierarchy.**

Extend classes to the presentation layer when using only a subset of the classification hierarchy

When using only a subset of the classification hierarchy, you must run the **clsutility** on the entire hierarchy. In this case, the argument **-include_instances** is not required since you only need to extend the classes to the nodes that are required.

Example:

clsutility -import -hierarchy -cid=ICM

Synchronize the presentation layer with the classification hierarchy

Use one of the following methods to synchronize the presentation layer with the underlying classification hierarchy:

- **Manually run clsutility.**
- Enable automatic synchronization that shadows specific operations by setting the **CLS_auto_sync_node_hierarchy** preference to **true**.

  When at least the top-level node of the storage and presentation hierarchies are **linked**, the synchronization mechanism is triggered by a change to groups, classes, or ICOs in the storage hierarchy and results in the following changes to the presentation hierarchy.

Operation on class hierarchy	Operation required on node hierarchy
Add/delete group	Add/delete group node
Add/delete class	Add/delete master node
Add/delete class ICO	Add/delete classification element
Copy/paste or cut/paste of a group	Copy/paste or cut/paste of a group node
Copy/paste or cut/paste of a class	Copy/paste or cut/paste of a master node

The synchronization mechanism functions as follows:

- If a target is not found in the presentation hierarchy, it is created (applies to node or element).
  - To create a node in the presentation hierarchy, Teamcenter matches the parent node with the parent class.
  - To create an element in the presentation node, Teamcenter matches the node with the storage class.
- To find targets, Teamcenter searches for the following:
  - A group node with the same ID as the classification group
  - A master node with the storage class type property pointing to the associated classification class
  - A classification element referencing the associated ICO

  The objects in the hierarchies are mapped as follows.

Object types from Classification (class hierarchy)	Corresponding object types from the presentation layer (node hierarchy)
Group	Group node
Abstract class	Master node
Storage class	Master node
Classification object (ICO)	Classification element (ICO)

Migrating traditional basic classification to advanced classification standard taxonomy

Advanced classification standard taxonomy (CST) offers some features that traditional basic classification does not. Additionally, in the future, CST will be regularly enhanced. For this reason, you may want to consider migrating your traditional basic classification classes to the advanced CST format. To do this, you can run the following utility:

clsutility -migrate -classification2cst -cid=*class-ID* **[-recursive | -confirm| -include_icos]**

> Caution:
>
> It is strongly recommended that you test the migration of your classes in a staging environment and evaluate the results. Deploy the migration in production only when you are completely satisfied with the test results.

This utility migrates traditional classification classes and all the objects (properties, dictionary attributes, key-LOVs, and associated images) referenced by that class, as well as all the parent classes, to the CST format. Optionally, the utility can also convert all child classes (**-recursive**), and any ICOs already existing in the class (**-include_icos**).

Running the utility creates a class in the database with an automatically generated IRDI. This IRDI is displayed in the utility output. Additionally, node definitions are created for application classes so that the migrated CST classes are visible and can be used for classification.

```
Migration summary for traditional objects
-------------------------------------------------------------------------------------------------------------
               TYPE |          SML ID |                      CST ID |       Status | Note
-------------------------------------------------------------------------------------------------------------
     NodeDefinition |              AD |                 SMPL1#AD#001 |  UNPROCESSED |
  PropertyDefinition |          -80001 |          SMPL1#02-f6f89w#001 |  UNPROCESSED |
  PropertyDefinition |          -80002 |          SMPL1#02-fsf89w#001 |  UNPROCESSED |
  PropertyDefinition |          -80005 |          SMPL1#02-e6f89w#001 |  UNPROCESSED |
  PropertyDefinition |          -80015 |          SMPL1#02-ccf89w#001 |  UNPROCESSED |
  PropertyDefinition |          -80018 |          SMPL1#02-bsf89w#001 |  UNPROCESSED |
    KeyLOVDefinition |          -80051 |          SMPL1#09-Tcf89w#001 |  UNPROCESSED |
  PropertyDefinition |          -80020 |          SMPL1#02-bMf89w#001 |  UNPROCESSED |
     ClassDefinition |  ADT-CONNECTION |  SMPL1#01-ADT-CONNECTION#001 |  UNPROCESSED |
      NodeDefinition |  ADT-CONNECTION |     SMPL1#ADT-CONNECTION#001 |  UNPROCESSED |
  PropertyDefinition |          -82005 |          SMPL1#02-q5989w#001 |  UNPROCESSED |
    KeyLOVDefinition |          -80040 |          SMPL1#09-WMf89w#001 |  UNPROCESSED |
  PropertyDefinition |          -86123 |          SMPL1#02-la989w#001 |  UNPROCESSED |
     ClassDefinition |   ADT-CONN-BUND |   SMPL1#01-ADT-CONN-BUND#001 |  UNPROCESSED |
      NodeDefinition |   ADT-CONN-BUND |      SMPL1#ADT-CONN-BUND#001 |  UNPROCESSED |

JSON input for KeyLOVDefinitions: C:\Users\ny6wyo\AppData\Local\Temp\KeyLOVDefinitions.json
JSON input for PropertyDefinitions: C:\Users\ny6wyo\AppData\Local\Temp\PropertyDefinitions.json
JSON input for ClassDefinitions: C:\Users\ny6wyo\AppData\Local\Temp\ClassDefinitions.json
JSON input for NodeDefinitions: C:\Users\ny6wyo\AppData\Local\Temp\NodeDefinitions.json

Operation completed successfully.
```

The **-confirm** argument creates the objects in the database. It is recommended that you run the utility without the **-confirm** argument and review the resulting JSON files. These are created in your *temp* directory. The exact path is included in the utility output. You can modify the JSON files if necessary and

import them manually or, if you are satisfied with them, rerun the migration utility using the **-confirm** argument.

You must set the following two preferences to run the migration:

- **CST_default_namespace**

  This preference specifies the namespace of all CST objects created by the migration utility.

- **CST_default_migration_status**

  This preference specifies the status of the CST objects created by the migration utility. Valid values are **Develop**, **Released**, or **Retired**. Objects set to **Released** can no longer be modified.

The output is displayed in tabular form with a column for status. This status can be one of:

- **UNPROCESSED**

  The migration utility was run in dryrun mode (default). You can review the JSON files created for the new object but it does not yet exist in the database.

- **SKIPPED**

  The reason that the object was skipped is displayed in the **Note** column.

- **SUCCESS**

  The migration utility was run with the **-confirm** argument. The object was successfully migrated to advanced classification.

- **FAILURE**

  The object was not successfully migrated. See the **Note** column for more information.

Limitations

- Class attribute, class detail, and view attribute options that are not supported in advanced classification are migrated as property values only. Your data is not lost, but is not relevant to advanced classification. Only the **Mandatory** and **Protected** options are supported in advanced classification.

- As advanced classification does not support localization for the **UserData1**, **UserData2**, and **Annotation** properties, localized values for these properties are not migrated.

- Default or minimum and maximum values set for properties are stored as individual property values. Advanced classification does not support inheritance, so if a default or minimum and maximum value

is inherited in rich client, the value that would be displayed through that inheritance mechanism is displayed as the property value in the migrated class or view. If one of the supported views contains a value in rich client that differs from the value displayed in the default view, the property is assigned the value displayed in the default view.

- Reference attributes are not supported in CST.
- Interdependent key-LOVs are migrated as individual property values.
- Only views that are supported in advanced classification (**Default**, **User**, **Group**, and **Role** views) can be migrated.

Troubleshooting the installation and configuration

- **All classification options are installed and indexing is configured but new objects are still not included in search results**

  To use traditional basic classification in the Active Workspace client, it is not necessary to install the presentation layer (**Next Generation Classification**). If, however, you inadvertently do install this option, you *must* also extend the data using **clsutility** to make use of full functionality.

- **Filter categories are localized but facet values are not**

  If you mark any classification property as localizable, you must run the **bmide_modeltool** utility.

- **Search is incorrectly configured**

  The following situations may arise if the search is not correctly configured:

  - You cannot search by classification properties.
  - The classification search works but facets are not visible (or there are unassigned facets).
  - You cannot search the library elements by classification property.

  Correctly configure the search.

- **The incorrect filters are displayed for library elements**

  1. Set the dynamic prioritization preferences to display all the configured filters. This allows you to investigate further.

     - Set **AWC_dynamicPriority_hideSingleValueFacetCategories** to **true**.
     - Set **AWC_dynamicPriority_hideUnassignedValueFacetCategories** to **true**.

2. Make properties searchable in the global search.

- **The classification search works, but there only a limited number of properties available in the search filters.**

  Increase the number of search facets displayed using the **AWC_classification_facets_threshold** preference. The default value is **200** and the facets are sorted with the most common facets at the top.

  > Caution:
  >
  > Setting this value to a high number can cause a decrease in performance or even cause problems with Solr.

- **Performing a bulk import of classification data**

  Ensure that the **CLS_auto_sync_node_hierarchy** preference is set to **false** before performing a PLM XML import of hierarchy data.

- **What business constants are involved in classification installation?**

  The following business object constants are set in Business Modeler IDE:

  - The **Awp0SearchClassifySearchEnabled** business object type constant is set to **true** at the workspace object level. This turns on the classification search.
  - The **Awp0SearchIsClassifyDataIndexed** business object type constant is set to **true** at the item and item revision level.

    This turns on classification indexing. The object type on which this constant is set is indexed for searching. All subtypes inherit the setting.
  - The **Awp0SearchIsIndexed** business object type constant is set to **true** at the item and item revision level.

    The object type on which it is set is indexed for searching. All subtypes inherit the setting.

    > Note:
    >
    > Do not set this constant at the workspace level since this prevents your database from being indexable.
    >
    > Instead, set it true for each object that you want to index. For example, item and item revision.

  Verify that these constants are correctly set using the **gettypeconstantvalue** utility, for example:

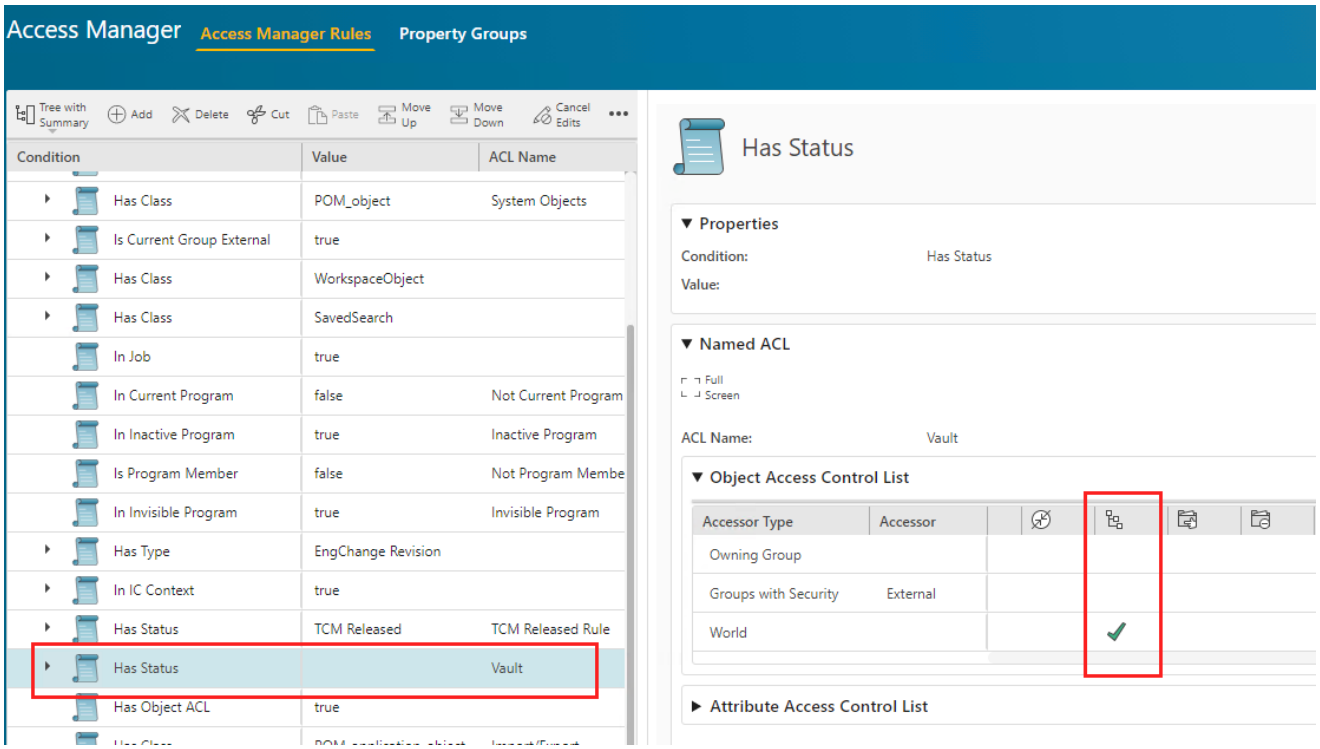
```
gettypeconstantvalue -u=username -p=password -g=dba
-typename=WorkspaceObject
-constantname=Awp0SearchClassifySearchEnabled
```

Give access to users for classifying objects

The following points determine whether you can classify an object (that is, whether the **Classify** button is visible for a selected object):

- The workspace object's type must be listed in the ICS_classifiable_types preference. If your business use case requires classifying items instead of item revisions, you must remove the **ItemRevision** entry from this preference.

- You must have write access to the workspace object. If this object is already released, you must have write access to the classification object (**write_ico** access). The workspace object access is set in the **Access Manager** application in rich client. The classification entries in the **Has Status() → Vault** rule must be set to write access.



- You must set additional classification preferences depending on the classification features you use.

Granting access to non-dba users for Classification Manager tasks

While configuring Access Manager rule conditions for a classification user, any user can be a classification admin user. It is not necessary for the user to be defined as a dba user for performing classification administrator tasks.

You must create an ACL to grant the **Read**, **Write**, **Delete**, **Change**, and **Copy** access for any non-dba user to access the Classification Manager application and perform classification administrator tasks.

Administering basic classification on Active Workspace

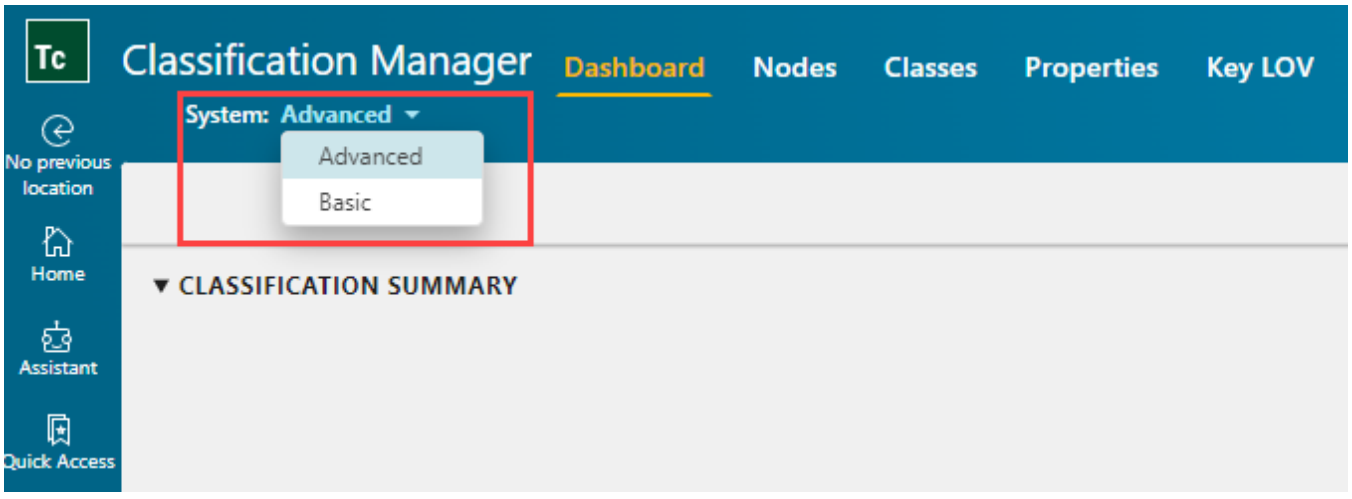
Viewing your hierarchy definitions

The basic and advanced classification hierarchy is created of nodes, classes, properties, and key-LOV administrative objects. These definitions are displayed in **Classification Manager**. The **Classification Manager** provides an overview of every classification definition in the database on the **Dashboard** tab, as well as details about each object on the **Nodes**, **Classes**, **Properties**, and **Key LOV** tabs.

The **CLASSIFICATION MANAGER** tile is found in the **Active Admin** workspace.

By default, the **Active Admin** workspace is mapped to the **dba** group and role. You can add other groups and roles as needed. Alternatively, you can create your own workspace that includes the **CLASSIFICATION MANAGER** tile.

You can view the classification definitions for basic or advanced classification by selecting either the **Basic** or the **Advanced** system in **Classification Manager**.



Authoring hierarchy definitions

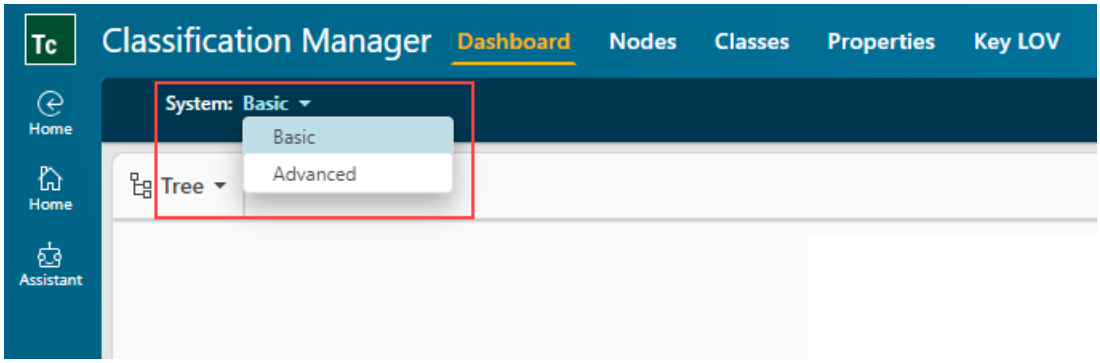
Creating classification hierarchy definitions

You use Classification Manager to define the nodes, classes, properties, and Key LOVs that form the classification hierarchy for basic and advanced classification. You also define and format the attributes that, when associated with a class, determine the type of information that is stored. The order when creating a classification hierarchy is to first create the Key LOV definitions, then the property definitions, then the class definitions, and finally the node definition so that the node is visible in the classification hierarchy for end users.

Use Classification Manager to do the following basic tasks:

- Create a Key LOV definition
- Create a property
- Create a class
- Create a node

You can create the classification definitions for basic and advanced classification. Select either **Basic** or **Advanced** from the **System** list available on the **Classification Manager** dashboard before creating the classification definitions.



Create a Key LOV definition

You use Key LOVs to define one or more values that can be set for specific classification attributes. Once created, these lists are stored in the data dictionary and can be used and reused as required.

Procedure

1. On the **Classification Manager** dashboard, click the **Key LOV** tab.
2. Click **New** ⊕.

3. In the **Add Key LOV** panel, specify the following details:

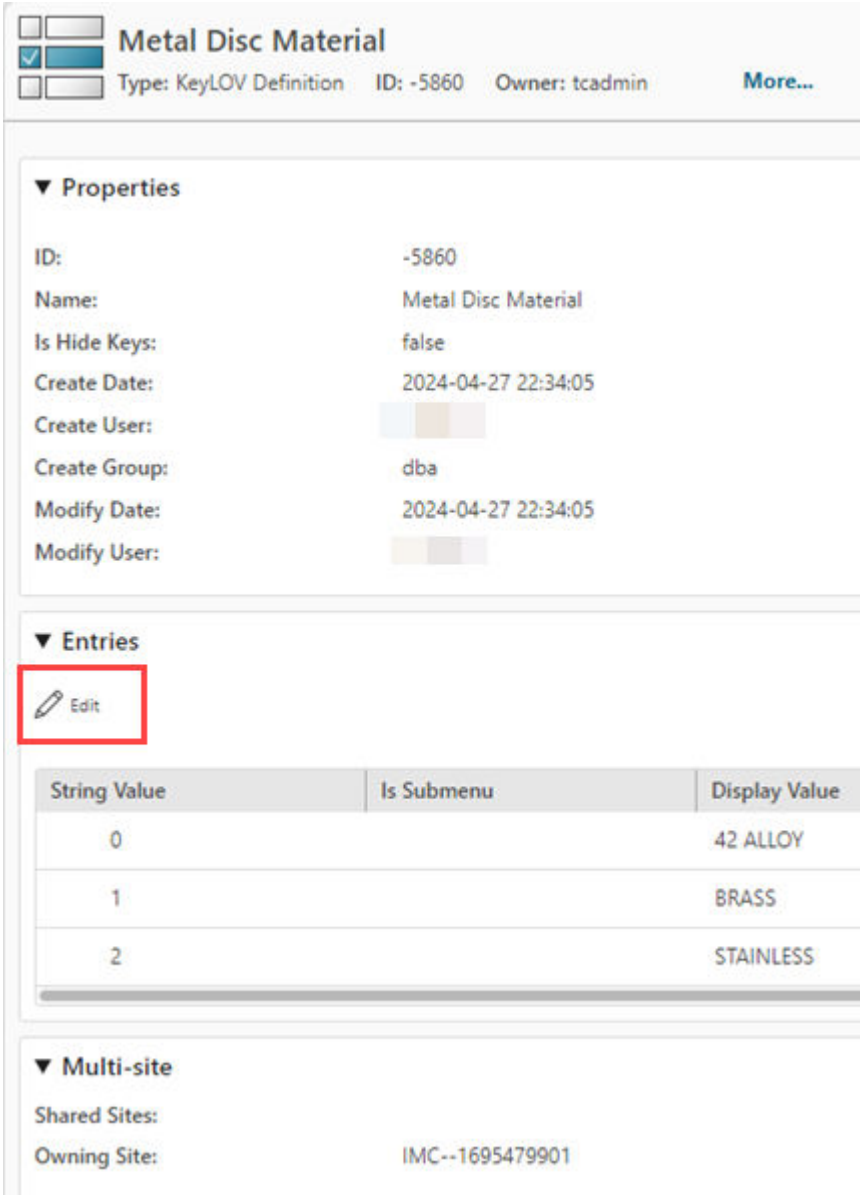
   - **ID**
   - **Name**

4. Click **Add**.

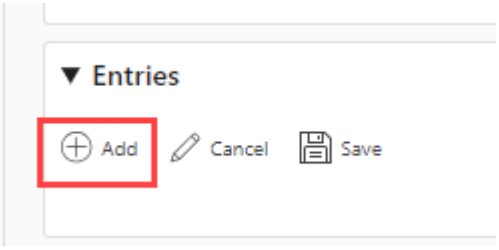
   The Key LOV definition is created.

5. After creating a Key LOV, you can add the entry for the Key LOV definition.

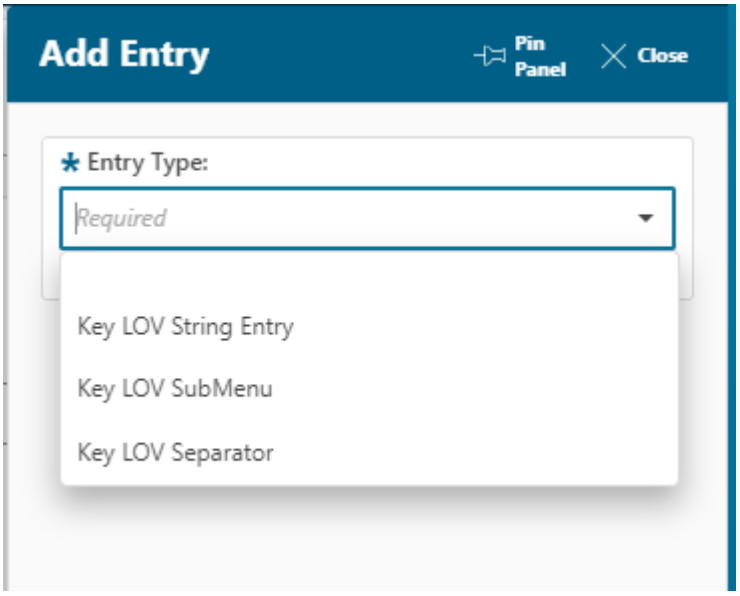
6. Open the Key LOV definition. Under **Entries**, click **Edit** in order to add entries to the Key LOV or to edit previously added entries.



7. Click **Add** ⊕.



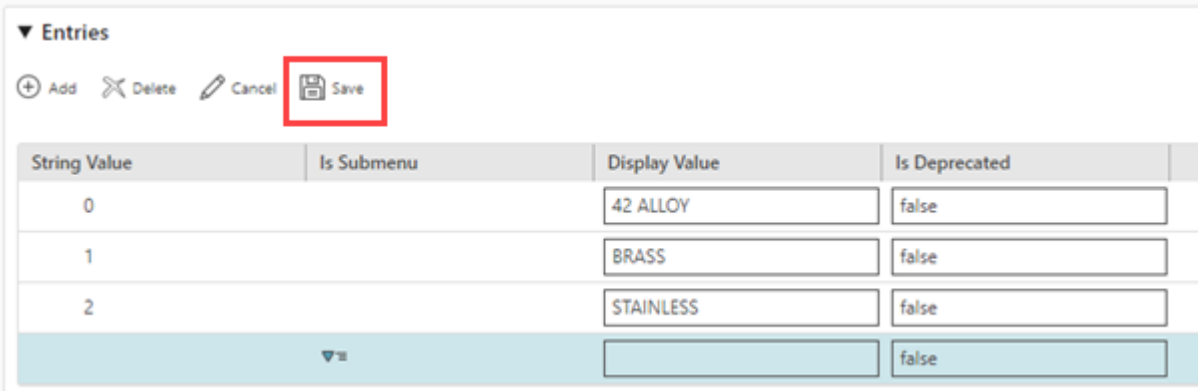
8. In the **Add Entry** panel, select the type of entry.



9. Specify the details as required and click **Add**

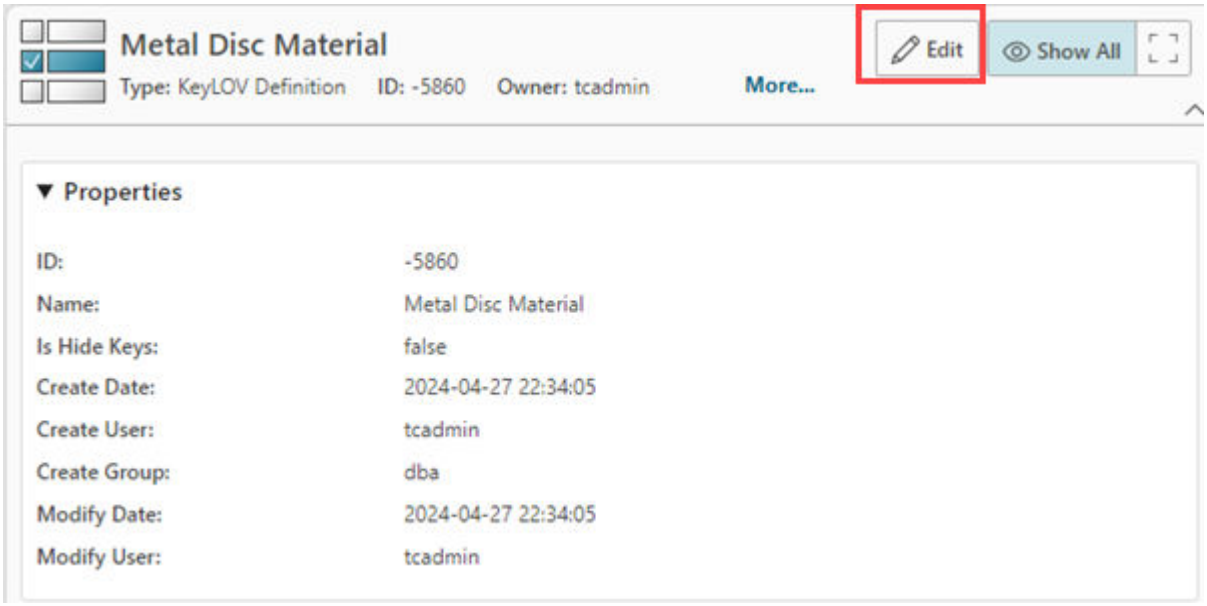
   You can add more entries to the Key LOV as required.

10. After all the entries are added, click **Save**.



11. To edit Key LOV definition, click **Edit**.

    The editable cells are activated.



12. Enter or update the Key LOV definition as required.

13. Save your changes for the Key LOV definition, by clicking **Save**. To cancel your edits, click **Cancel**.

Create a property

A *property* is used to describe the attributes of a class. Attributes can be of any type, including string, integer, double, boolean, and reference.

Procedure

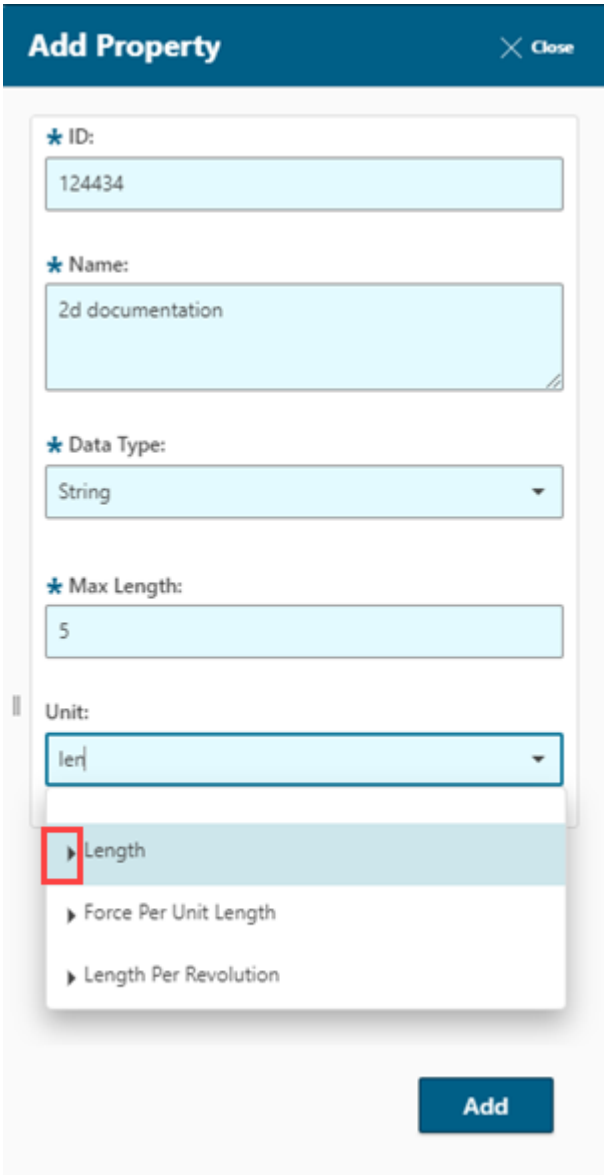
1. On the **Classification Manager** dashboard, click the **Properties** tab.

2. Click **New** ⊕.

3. In the **Add Property** panel, specify the **ID** and **Name** for the property.

4. Select the **Data Type** from the list and then specify the required values for the selected data type, if available.

5. Select the **Unit** applicable to the property from the list.



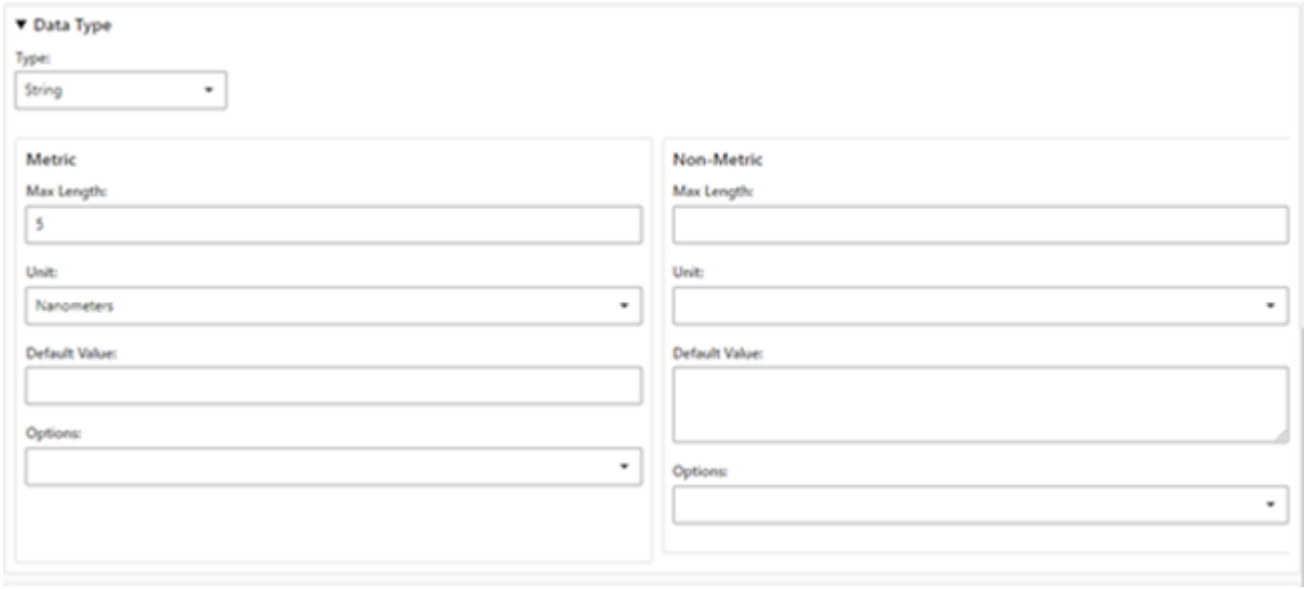
You can expand the arrow to select a specific unit type. For example, for **Length**, you can select from the different units of length such as, **Meters**, **Centimeters**, **Nanometers**, and so on.

6. Click **Add**.

7. To edit any property, click **Edit** .

   The editable cells are activated.

8. Enter or update the property values as required.



9. Save your changes by clicking **Save**. To cancel your edits, click **Cancel**.

Create a class

Create a class

Classes define the attributes that are stored for classified objects and determine which attributes are inherited by other classes. They are the primary building blocks of the Classification system. Workspace objects are classified by choosing a class from the hierarchy and providing values for the attributes of the class.

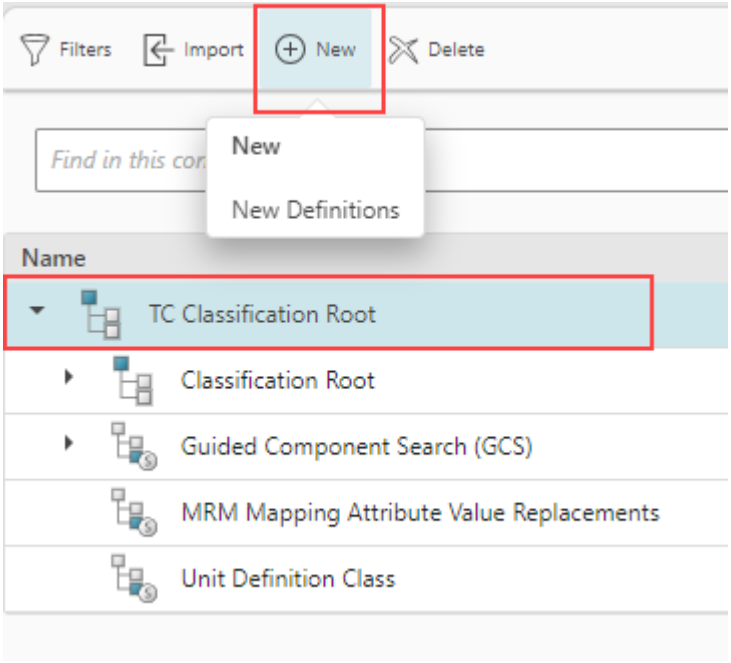
Procedure

1. On the **Classification Manager** dashboard, click the **Classes** tab.

2. Select a hierarchy under which you want to add a new class. You can do anyone of the following:

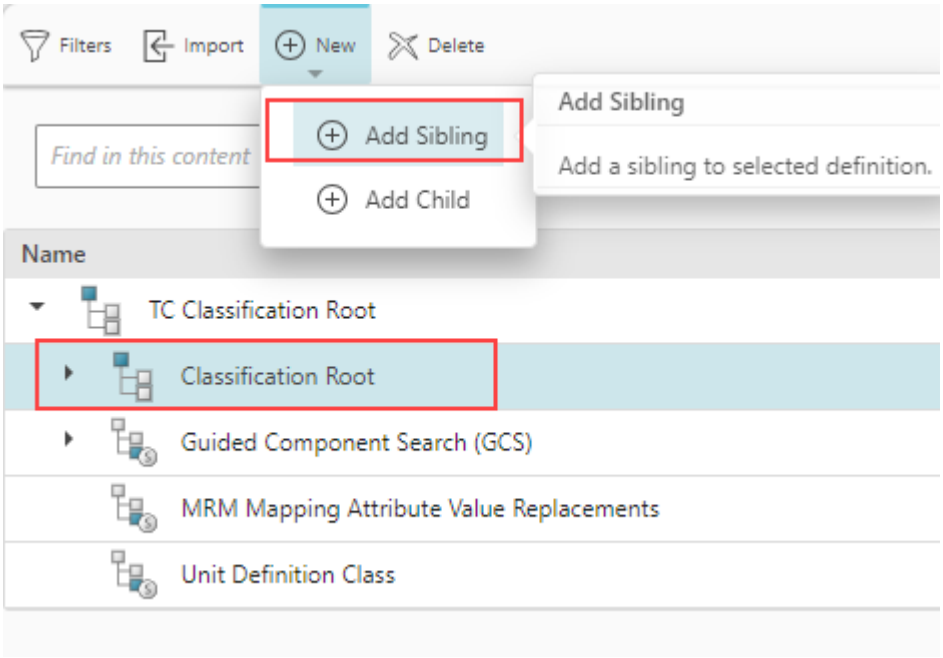
   - Add a class definition directly under the top level root class.

     To do this, select the top root class and click **New** ⊕.



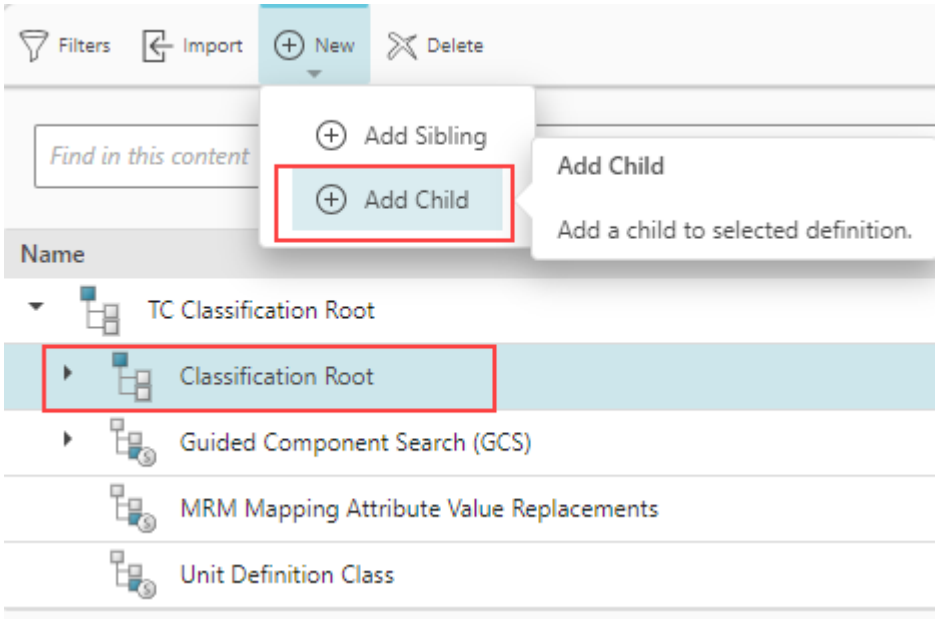
- Add a class definition as a sibling class at the same level as an existing class definition.

  To do this, select an existing class definition, click **New** ⊕ > **Add Sibling** ⊕.



- Add a class definition as a child class under an existing class definition.

  To do this, select an existing class definition, click **New** ⊕ > **Add Child** ⊕.



3. Select **Class Type** from the list.

   - **Group**
   - **AbstractClass**
   - **StorageClass**

4. Specify the **ID**, **Name**, and other details as required.

5. Click **Add**.

   A new class is added within the selected hierarchy. You can click **Show All** in the **Overview** tab to view all the properties associated with the class.

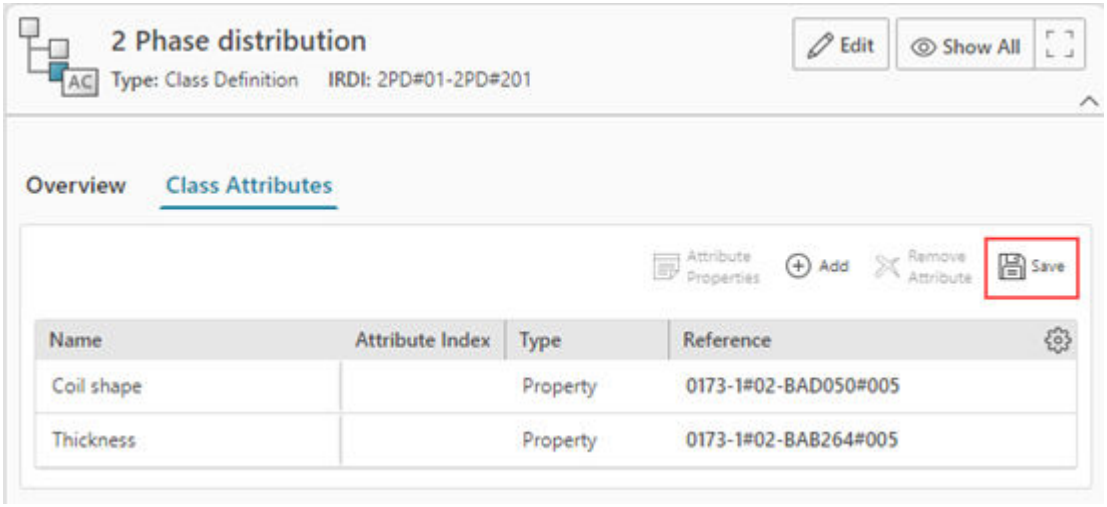
   After creating a class, you must **add attributes to the definition of the class**. Attributes added to the class become inherited attributes of any child classes.

Create a class attribute

Attributes are either inherited or directly associated with the class. The **Inherited Attributes** list displays all the attributes that are inherited from any and all parent classes. Inherited attributes cannot be deleted or modified. You can, however, add attributes to the definition of the class. Attributes added to the class become inherited attributes of any child classes. A class can contain a maximum of 200 attributes, both inherited and local.

Procedure

1. Open the class definition to which you want to add the class attribute. Click **Class Attributes**.
2. Click **Add** ⊕.
3. In the **Add Class Attribute** panel, search for the attributes that you want to add to the class. Available search filter options are:
   - **Name**
   - **ID**
4. Specify the **Value** of the selected attribute.
5. Click **Search**.
6. Select the class attribute that you want to add from the search results and click **Add.** You can also select multiple class attributes to add at a time.
7. After all the attributes are added, click **Save**.



The new class attributes are displayed automatically in the **Class Attributes** table.

8. To view additional attribute properties for each class attribute, select the class attribute and click **Attribute Properties**.

   If you select multiple class attributes and click **Attribute Properties**, you can compare attribute properties for all the selected class attributes.

9. To remove a class attribute, select the attribute and click **Remove Attribute** ✕.

10. To edit any class attribute, click **Edit** .

    The editable cells are activated.

11. Enter or update the class attributes as required.

12. To save your changes, click **Save** . To cancel your edits, click **Cancel** .

Create a node

You can create a node with details such as node properties, application class, parent, and the details about the class and its properties. You can use the property groups navigator to get the property details within each property group.

> Note:
>
> For basic classification, nodes are created automatically as soon as a class is saved provided the preferences **CLS_is_presentation_hierarchy_active** and **CLS_auto_sync_node_hierarchy** are set to **True**. Additionally, If you cannot find the newly created classes, verify if the following tasks are complete:
>
> - **Extend classes to the presentation layer using clsutility**
> - Create search indexing views
> - **Index classification data**

However, if the nodes are not created automatically, you can do so for the basic data as can be done for the advanced classification as follows.

Procedure

1. On the **Classification Manager** dashboard, click the **Nodes** tab.

2. Click **New** > **Add Sibling** .

3. In the **Add Node** panel, select the **Node Type** from the list.

4. Specify the details in the remaining fields based on the selected node type.

5. Click **Add**.

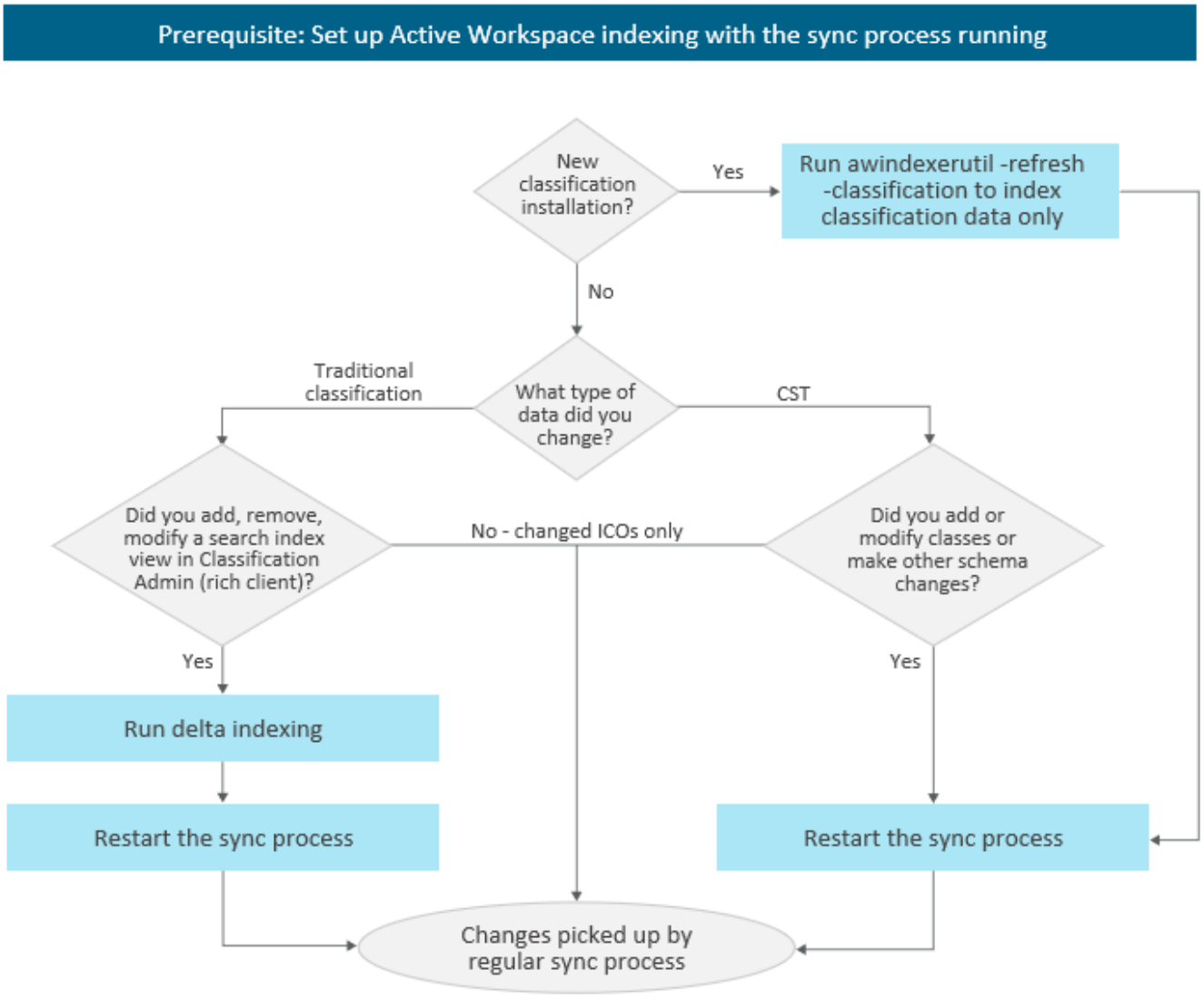
   A node is created, and the information about the node type and the application class is displayed under the **Overview** section.

After you create or update the classification definitions for advanced classification in Classification Manager, you must release these definitions for the classification hierarchy to be available in the user interface for classifying the data.

Indexing classification data

About indexing classification data

Many changes to classification classes and data require indexing. There are several classification indexing options available to you. Classification indexing is based on the general Active Workspace indexing mechanism. As a prerequisite, classification assumes that general Active Workspace indexing is fully functional and synchronizing on a regular basis. The following workflow assists you with your indexing decisions:



Index classification data

If indexing is set up and running, use classification indexing to discover only the objects whose classification data has changed. This process marks these objects for indexing and they are picked up by the next scheduled round of regular indexing.

1. Ensure that regular indexing is running by entering the following in a command line:

   ```
   runTcFTSIndexer -task=objdata:test
   ```

2. Do one of the following:

   - Mark all classification data for indexing (or re-indexing):

     Run the following command:

     ```
     awindexerutil -u=user -p=password -g=group -classification -refresh
     ```

     Use this command if you have performed an initial installation of a classification feature or made substantial changes to your classification data and want to refresh all data. Running this command marks the following objects for indexing:

     - Workspace objects classified in either traditional basic or advanced CST hierarchies
     - Standalone classification objects (**cls0Object** and **cls0cstObject**)
     - Library elements

     If the release you are upgrading to includes new classification features, it generally also includes schema changes. During this upgrade procedure, all classification data is marked for re-indexing and included during your next indexing synchronization. This can cause the synchronization to take longer than usual.

   - Index only changes made to the search index views in traditional basic classification:

     The search index views in traditional basic classification define which classes and properties are searchable. If you make any changes to these views, you can index only the changes made in the search index views. Changes in the search index view that require indexing are:

     - Adding or removing a search index view from a class
     - Adding or removing properties from a search index view

     a. Make properties searchable in the global search.

     b. (Optional) Run the **awindexerutil** utility:

```
awindexerutil -u=user -p=password -g=group -delta
-classification -dryrun
```

Running this utility using the **dryrun** option displays the objects that will be marked for indexing in the next step. Scanning this output can assist you in troubleshooting the indexing of your data.

c. Run the **awindexerutil** utility:

```
awindexerutil -u=user -p=password -g=group -delta
-classification
```

This step marks changed data for indexing. Once marked, these objects are picked up in the next regular indexing synchronization flow.

Starting and stopping the indexing synchronization process

Before you begin classification indexing activities, you must stop the index synchronization process. The classification indexing mechanisms then mark objects for indexing. These objects are indexed the next time the synchronization process occurs.

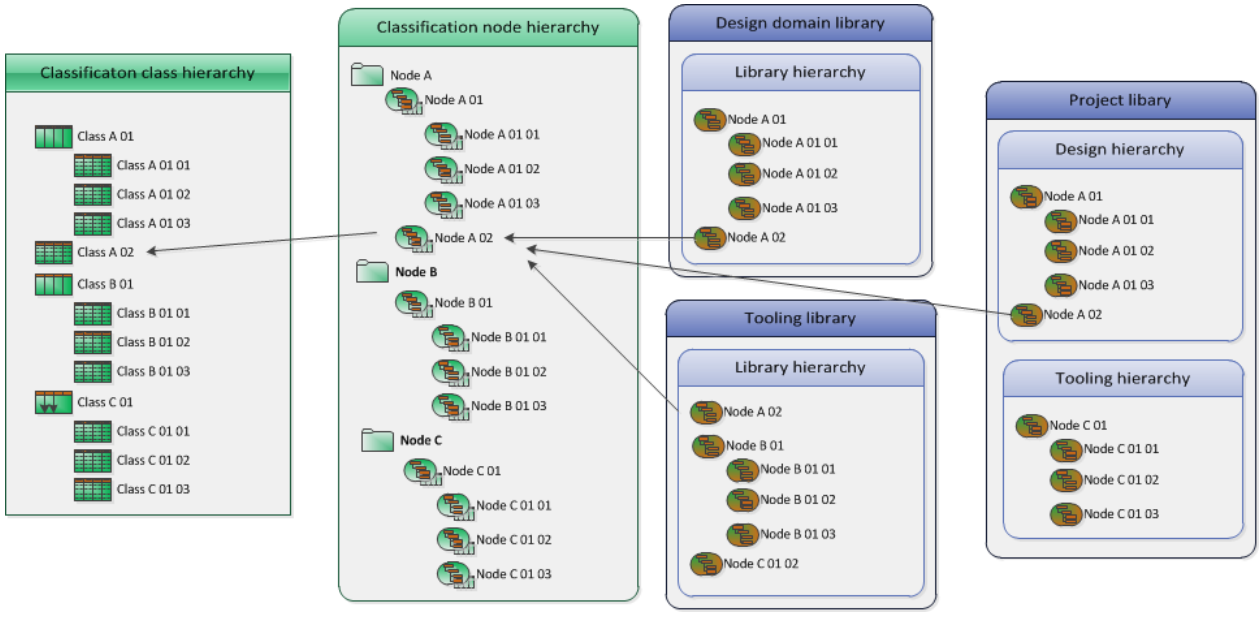
After performing any classification indexing activities, is important to restart the synchronization process so the marked objects are picked up and indexed. Do this by scheduling regular syncing of the database with the **runTcFTSIndexer** with the **-task=objdata:sync** argument.

Administering classification libraries

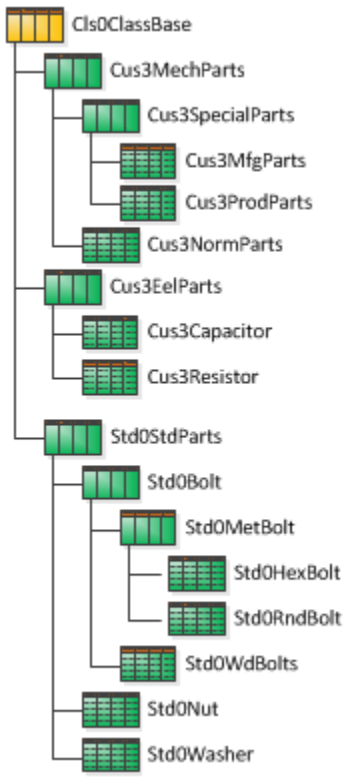
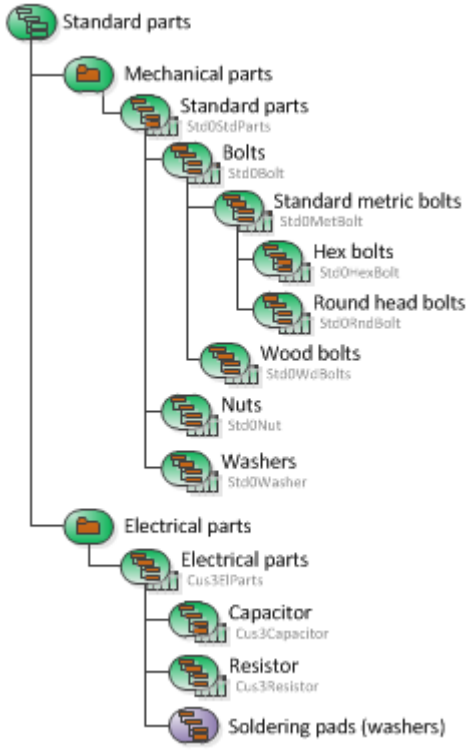
Library management overview

Libraries are used to organize classified objects with similar characteristics or application in a hierarchy. They reflect the business use of the objects they contain. For example, if you were looking for a bolt, you would start by searching for the bolt library. Once you find the bolt library, you can perform additional searches, use filters, or browse through the hierarchy to find the object that meets your criteria.

The class hierarchy is built based upon attribute inheritance. This does not always reflect the business use of the object for which you search. Libraries are based on a *presentation hierarchy* and provide you with a different view on the data. The presentation hierarchy is built on top of the traditional classification hierarchy and mirrors it. However, because it is built from workspace objects, it supports the flexibility of libraries. A library contains library elements that are used to manage library objects and, ultimately, reference classification classes. A classifying node links to a storage class. Using an optional synchronization mechanism, you can ensure that any actions performed on the traditional classification hierarchy are also performed on the presentation hierarchy.



A presentation hierarchy can have its own symbols and images but its structure is a one-to-one reflection of the classification storage hierarchy. Classification nodes in the presentation hierarchy refer back to classification classes in the storage hierarchy.

Data that is stored in a storage hierarchy...	**... is displayed in a presentation hierarchy**
	

There is no user interface for classification libraries. You can display presentation layer nodes in My Teamcenter and search for data filtered by libraries in Active Workspace. Libraries are administered using the **clsutility** and **lbrmanger** utilities.

Membership rules are used to populate a library on demand. A membership rule tells Teamcenter to search through a part of a presentation hierarchy and find all elements that have a particular characteristic, and then create a library element for the node element. For example, select all classified bolts from the **Bolts** class that are made of titanium and whose diameter is greater than 20. You can create multiple rules for a particular library node, and you can search either presentation hierarchies or other libraries to populate a new library.

Types of objects you work with

When creating libraries, you work with the following objects:

- Library: container
  - Hierarchies
    - Hierarchy nodes
      - ◊ Library elements
      - ◊ Membership rules
- Specifications
  - Specification rules

Library	Classification data that is filtered to suit a particular business need.
Hierarchy	Structures within a library that organize their constituent library elements. A library can contain multiple hierarchies. A hierarchy (also referred to as a presentation hierarchy) refers back to classes in the classification storage hierarchy.
Node	A representation of a classification storage class within a library hierarchy.
Membership rule	Rule that provides an automated way to populate a library with library elements. Rules identify the objects that should become members of a library and are specific to a particular library node. They can be evaluated at any time to update a library. For example: Rule A: Select all classified bolts from where material is titanium. Rule B: Select all bolts from the vendor catalog where material is titanium and diameter greater than 20.

Specification Design or component selection guidelines set up by expert users for a particular domain or discipline to assist you in finding components suitable to a particular design purpose.

Using the clsutility and the lbrmanager utility

Libraries are administered using two command line utilities:

- **clsutility**

  This utility is used to create and update the presentation layer based on the classification hierarchy.

- **lbrmanager**

  This utility is used to create, display, publish, and retract library objects.

These utilities contain many arguments and subarguments. The instructions about how to use the utilities is embedded in the utilities themselves. To obtain help for the utilities, enter:

utility-name **-h**

For example:

lbrmanager -h

To obtain help about a subargument, enter:

utility-name -subargument **-h**

For example:

clsutility -include_instances -h

These utilities are installed when you **install the library feature**.

Create library data using the lbrmanager utility

The following examples demonstrate how to create, display, publish, and retract library objects.

> Note:
>
> To obtain more information about any of these commands, type:

command-name -sub-command **-h**

For example:

lbrmanager -create -library -h

Create a standalone library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -library -id=***MechanicalPartsLibrary* **-name=***MechanicalPartsLibrary* **-descr="***MechanicalParts Library***" -type=***Domain* **-disciplines="***Mechanical,Electrical***"**

To display a library:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -libraries -id=***MechanicalPartsLibrary*

Create a hierarchy

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -hierarchy -id=***FixturesHierarchy* **-name=***FixturesHierarchy* **-descr="***Fixtures Hierarchy***" -libraryId=***MechanicalPartsLibrary*

To display a hierarchy:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -hierarchies -libraryId=***MechanicalPartsLibrary* **-id=***FixturesHierarchy*

Create a library node

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -node -id=***Assemblies* **-name=***Assemblies* **-descr="***All Assemblies***" -libraryId=***MechanicalPartsLibrary* **-hierarchyId=***FixturesHierarchy*

To display a node:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* *-id=Assemblies*

Create a child library node

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -node -id=***BasePlate* **-name=***BasePlate* **-descr="***Base Plate***" -libraryId=***MechanicalPartsLibrary* **-hierarchyId=***FixturesHierarchy* **-parentNodeId=***Assemblies*

To display a node:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* **-id=***BasePlate*

Display the previously created library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -libraries -id=***MechanicalPartsLibrary*

Output	Description
`[MechanicalPartsLibrary] MechanicalPartsLibrary`	Library created in *Create a standalone library*.
`|-[FixturesHierarchy] FixturesHierarchy`	Hierarchy created in *Create a hierarchy*
`|-[Assemblies] Assemblies`	Node created in *Create a library node*
`|-[BasePlate] BasePlate`	Child node created in *Create a child library node*

Publish an item to a library

Prerequisite data:

- Two items: **BasePlateItem1**, **BasePlateItem2**
- A text file, **BasePlate.txt**, with the following content:

  -type=Item -mfkPropNames=item_id -item_id=BasePlateItem1

  -elemPropNames=lbr0ElementId -lbr0ElementId=BasePlateLE01

  -type=Item -mfkPropNames=item_id -item_id=BasePlateItem2

  -elemPropNames=lbr0ElementId -lbr0ElementId=BasePlateLE02

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-publish -objectsFromFile -libraryId=***MechanicalPartsLibrary* **-nodeId=***BasePlate* **-inputFile=***BasePlate.txt*

To display a library with a library element:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* **-id=***BasePlate*

Output	Description
`[MechanicalPartsLibrary] MechanicalPartsLibrary`	Library created in *Create a standalone library*.
`|-[FixturesHierarchy] FixturesHierarchy`	Hierarchy created in *Create a hierarchy*

Output	Description
`|-[Assemblies] Assemblies`	Node created in *Create a library node*
`|-[BasePlate] BasePlate`	Child node created in *Create a child library node*
`->[BasePlateLE01] BasePlateItem1`	Element created in *Publish an item to a library*
`->[BasePlateLE02] BasePlateItem2`	Element created in *Publish an item to a library*

Retract an item from a library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-retract -byElementkey -libraryId=***MechanicalPartsLibrary* **-nodeId=***BasePlate* **-elementIds=***BasePlateLE01*

Setup membership rules

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -memberRule -name=***TestRule* **-nodeId=***BasePlate* **-libraryId=***MechanicalPartsLibrary* **-propName=***sml01001* **-sml01001="***steel***"**

Evaluate membership rules to populate a library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-evaluate -memberRules -libraryId=***MechanicalPartsLibrary*

Instantiate a library element into a collaborative design

Prerequisite data:

- A collaborative design with ID **DesignID01**

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-instantiate -test -designId=***DesignID01***-libraryId=***MechanicalPartsLibrary* **-elementId=***BasePlateLE02*

Creating a library from the Fixtures branch of the Manufacturing Resource Library—Example

```
# Create library: "Resources Library"
lbrmanager -u= -p= -g= -create -library -id=MRM_LIB -name="Resources"
-descr="Resources Library" -administrators=Tool_Admin

# Create hierarchy: "Fixtures"
lbrmanager -u= -p= -g= -create -hierarchy -libraryId=MRM_LIB
-id=MRM_FIXT_ROOT
-name=Fixtures -descr="Fixtures Hierarchy"

# Create library nodes and members
```

```
lbrmanager -u= -p= -g= -create -nodesFromClassification
-libraryId=MRM_LIB
-hierarchyId=MRM_FIXT_ROOT -includeInstances -clsNodeId=FIXT

# Create hierarchy: "Machines"
lbrmanager -u= -p= -g= -create -hierarchy -libraryId=MRM_LIB
-id=MRM_MACH_ROOT
-name="Machines" -descr="Machines Hierarchy"

# Create library nodes and members
lbrmanager -u= -p= -g= -create -nodesFromClassification
-libraryId=MRM_LIB
-hierarchyId=MRM_MACH_ROOT -includeInstances -clsNodeId=MDL

# Create hierarchy: "Factory Resources"
lbrmanager -u= -p= -g= -create -hierarchy -libraryId=MRM_LIB
-id=MRM_FRL_RES_ROOT
-name="Factory Resources" -descr="Factory Resources Hierarchy"

# Create library nodes and members
lbrmanager -u= -p= -g= -create -nodesFromClassification
-libraryId=MRM_LIB
-hierarchyId=MRM_FRL_RES_ROOT -includeInstances -clsNodeId=FRL

# Create library: "Tools Library"
lbrmanager -u= -p= -g= -create -library -id=TOOLS_LIB -name="Tools
Library"
-descr="Tools Library" -administrators=Tool_Admin

# Create hierarchy: "Tools"
lbrmanager -u= -p= -g= -create -hierarchy -libraryId=TOOLS_LIB
-id=MRM_TOOLS_ROOT
-name=Tools -descr="Tools Hierarchy"

# Create library nodes and members
lbrmanager -u= -p= -g= -create -nodesFromClassification
-libraryId=TOOLS_LIB
-hierarchyId=MRM_TOOLS_ROOT -includeInstances -clsNodeId=TOOL_MRL
```

Sharing classification data

Sharing classification data

Sharing classification data

Very often, company data must be managed across many sites. Users, however, require access to consistent, up-to-date data at each site. This is achieved by sharing data across the multiple sites. Active Workspace uses the TC XML based Multi-Site, Briefcase Browser, or PLM XML mechanisms to share classification data.

It is recommended that you read the general documentation about sharing data before you begin sharing your classification data.

There are two types of classification data that you can share across sites:

- Classification hierarchy

  Sharing this data is normally performed by a classification administrator. The node hierarchy must be shared before classification objects can be shared.

- Classified objects

  This data can be shared by classification users. When classified objects are shared, the hierarchy data is not shared with them.

The sharing mechanisms require that you share a workspace object. Traditional basic classification classes are not workspace objects. You can share the node that references the traditional classification classes. You find the workspace object to share (hierarchy node or classified object) by using the Advanced search.

Note:

- You cannot transfer the ownership of advanced classification standard taxonomy (CST) standard master nodes or classified objects. You can, however, replicate released data to another site.
- Classification does not support Multi-Site publication.
- You cannot use Multi-Site with traditional basic classification data.

Share the classification hierarchy

Generally, only classification administrators or power users transfer hierarchy data. Use one of the following methods to share a classification hierarchy:

- **Multi-Site**

  Use Multi-Site to replicate classification hierarchy nodes to another site. Additionally, you can transfer the ownership of the nodes.

  You cannot transfer the ownership of classification standard taxonomy objects.

- **Briefcase file**

  Replicate classification hierarchy nodes at another site.

  You cannot transfer the ownership of hierarchy objects in a briefcase file.

- **PLM XML**

  Replicate classification hierarchy nodes to another site. Unlike the other two sharing methods, sharing using PLM XML creates a copy of classification hierarchy nodes at the target site (creates a different UID of the object at the target site). This can prevent you from sharing using the other two methods.

  This method is not supported for classification standard taxonomy objects. If you export a CST object with PLM XML, the resulting file is blank.

Share hierarchy data using Multi-Site

1. Search for the node that you want to share using a **General** query of the Advanced search.

   Select **Master Node**, **Group Node**, or **Standard Master Node** in the **Type** filter when searching.

2. Select the desired node in the search results and choose **Share**>**Share with Sites** in the primary toolbar.

3. Select the target site.

4. In addition to the regular transfer options, select the following classification-specific options:

Option	Description
Option Set	Select **MultiSiteExpOptSet** to enable other classification sharing options.
Transfer Site Ownership	Transfers the class ownership to another site. This should only be selected after all dictionary attributes referenced by the class are shared. If you do not select this option, the node is replicated to the target site but cannot be changed at that site. It can be used for classifying objects at the target site.
Include node sub-hierarchy	Shares all child nodes of the selected node. Parent nodes of the selected node are always shared.
Include node members	Shares all the objects classified in the nodes being shared. Note: This option is not supported for classification standard taxonomy (CST) classes.
Include dictionary properties	Includes dictionary properties when sharing master nodes. Dictionary attributes must be replicated at the second site before the ownership of the master node can be shared. There are two scenarios to consider when you want to share the ownership of a master node:

Option	Description
	• You want to transfer ownership of a master node and the ownership of its dictionary properties. This is a one-step process that replicates the dictionary properties at the same time the ownership is transferred: • Select **Transfer Site Ownership** and **Include dictionary properties**. • You want to transfer the ownership of the master node but you do *not* want to transfer the ownership of the dictionary properties. This is a two-step process ensuring that the dictionary properties are replicated before ownership of the node is transferred: a. Share the master node without selecting **Transfer Site Ownership**. b. Share the master node with the selection of the **Transfer Site Ownership** option but without the selection of the **Include dictionary properties** option. Dictionary properties are always replicated when sharing a master node without ownership transfer of the node even if you do not select the **Include dictionary properties** option. If you select **Transfer site ownership** but have not already replicated the dictionary attributes, an error is displayed.

5. Click **Share**.

The nodes and, if selected, their classification objects, are shared to the target site. This is an asynchronous process. You can monitor the status in the **Alerts** area.

After replication, the node is displayed in search results with the symbol.

You can now classify objects in both the original and the replica class but you can only change a class at the site that has ownership.

Share hierarchy data using briefcase file

1. Search for the node that you want to share using a **General** query of the Advanced search.

   Select **Master Node** in the **Type** filter when searching.

2. Select the desired node in the search results and choose **Share**>**Export to Briefcase** in the primary toolbar.

3. Select the target site.

4. In **Transfer Option Set**, select **TIEUnconfiguredLLBCZExportDefault**.

5. Click **Export**.

6. Download and give the exported BCZ file to the target site.

7. Import the file with the **Import from Briefcase** command using the **TIELLBCZImportOptionSetDefault** transfer option set.

The nodes and, if selected, their classified objects, are shared to the target site. This is an asynchronous process. You can monitor the status in the **Alerts** area.

After replication, the node is displayed in search results with the symbol.

You can now classify objects in both the original and the replica class but you can only change a class at the site that has ownership.

Share hierarchy data using PLM XML

Sharing the hierarchy with PLM XML is a two step process: first you export the node, then you can share the resulting zip file with the second site that must import it using PLM XML.

1. Search for the hierarchy node that you want to export using the **General** query of the Advanced search.

   You can only share **Master Node** type nodes. CST nodes (**Standard Master Node** are not supported).

2. Select the desired node in the search results and choose **Share**>**PLMXMLExport** in the primary toolbar.

3. Select the appropriate transfer mode.

   For classification objects, you must select the transfer modes that begin with **CLS**.

   Do not use the **ICS** transfer modes as these are not applicable to classification in Active Workspace.

4. Click **Export**.

5. Download and give the exported ZIP file to the target site.

6. Import the file with the **Import PLMXML** command.

Sharing classified objects

Use one of the following methods to share a classified object:

- **Multi-Site**

  Use Multi-Site to replicate classification object to another site. Additionally, you can transfer the ownership of the nodes.

  You cannot transfer the ownership of classification standard taxonomy objects.

- **Briefcase file**

  Replicate classified object at another site and, optionally, transfer ownership.

  All object types for which you want to transfer ownership using a briefcase file must be added to the **Briefcase_ownership_transfer_supported_types** preference.

  You cannot transfer the ownership of classification standard taxonomy objects.

- **PLM XML**

  Replicate a classification object to another site. Unlike the other two sharing methods, sharing using PLM XML creates a copy of a classification object at the target site (creates a different UID of the object at the target site). This prevents you from sharing using the other two methods.

  This method is not supported for classification standard taxonomy objects. If you export an object classified to a CST class with PLM XML, the resulting file is blank.

Share classified object using Multi-Site

1. Search for the classified object that you want to share and choose **More Commands** > **Share**>**Share with Sites** in the primary toolbar.

2. Select the target site.

3. In addition to the regular transfer options, select the following classification-specific options:

Option	Description
Option Set	Select **MultiSiteExpOptSet**.
Transfer Site Ownership	Transfers the ownership of the classified object to another site.

4. Click **More Commands** > **Share**.

The classified objects are shared to the target site. This is an asynchronous process. You can monitor the status in the **Alerts** area.

Replicating the classification object using a briefcase file

1. Search for the classified object that you want to share.
2. (Optional) Mark the object for ownership transferal by selecting **More Commands**••• > **Manage**>**Transfer Ownership** in the primary toolbar.
3. Choose **Share**>**Export to Briefcase**.
4. Select the target site.
5. In **Transfer Option Set**, select **TIEUnconfiguredLLBCZExportDefault**.
6. Click **Export**.
7. Download and give the exported BCZ file to the target site.
8. At the target site, import the file with the **Import from Briefcase** command.

The nodes and, if selected, their classification objects, are shared to the target site. This is an asynchronous process. You can monitor the status in the **Alerts** area.

Replicating classified objects using PLM XML

Sharing a classified object using PLM XML is a two-step process: first you export the object, and then you can share the resulting zip file with the second site that imports it using PLM XML.

1. Search for the classified object that you want to export and choose **More Commands**••• > **Share**>**PLM XML Export** in the primary toolbar.
2. Select the appropriate transfer mode.

   The transfer mode you select depends on your data. For classification objects, you must select transfer modes that begin with **CLS**, for example, **CLSExportInstance**.

   Do not use the **ICS** transfer modes as these are not applicable to classification in Teamcenter.
3. Click **Export**.
4. Download and give the exported ZIP file to the target site.
5. Import the file with the **Import PLM XML** command.

   If you import a standalone classification object, you must select the **ConfiguredDataImportDefault** transfer mode.

About Multi-Site and the presentation layer

There are three scenarios when working with Multi-Site and the presentation layer:

- You have installed classification and the presentation layer and are just beginning with Multi-Site.

  1. **Share the node hierarchy**.

  2. **Share the classified objects.**

- You have a functioning Multi-Site in a traditional basic classification environment in the rich client. Your class hierarchy and data is already shared. You now want to move to Active Workspace and have installed the presentation layer.

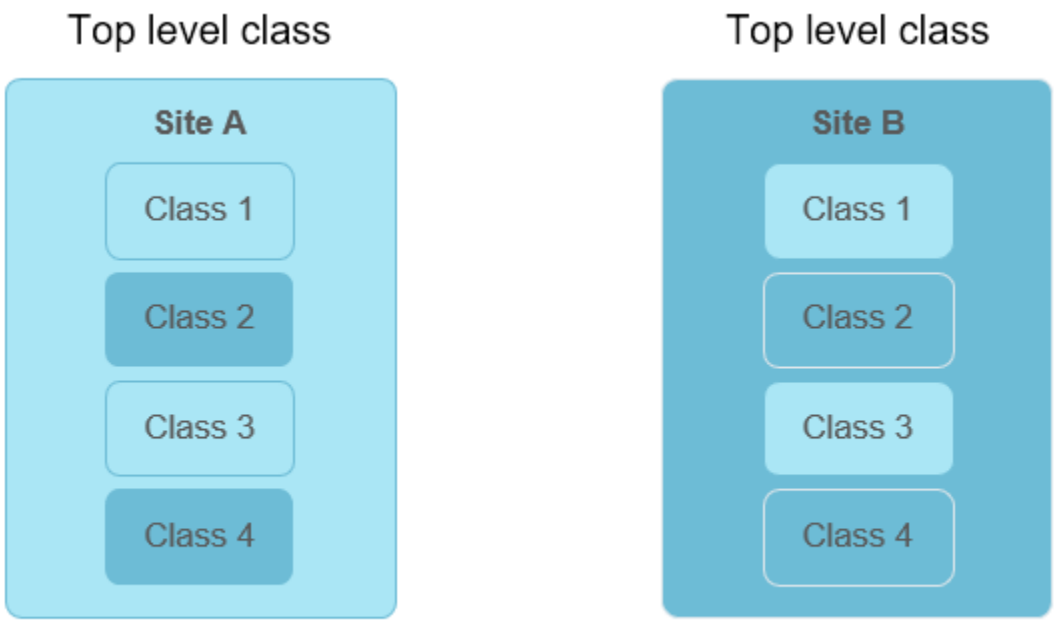
  1. Extend the hierarchy at the first site using **clsutility**.

  2. Share the extended node hierarchy to the second site.

  3. Share the extended node hierarchy to the second site.

- You have a functioning Multi-Site in a traditional classification environment in the rich client. Your class hierarchy and data is already shared and you have distributed ownership over your sites. You now want to move to Active Workspace and have installed the presentation layer.

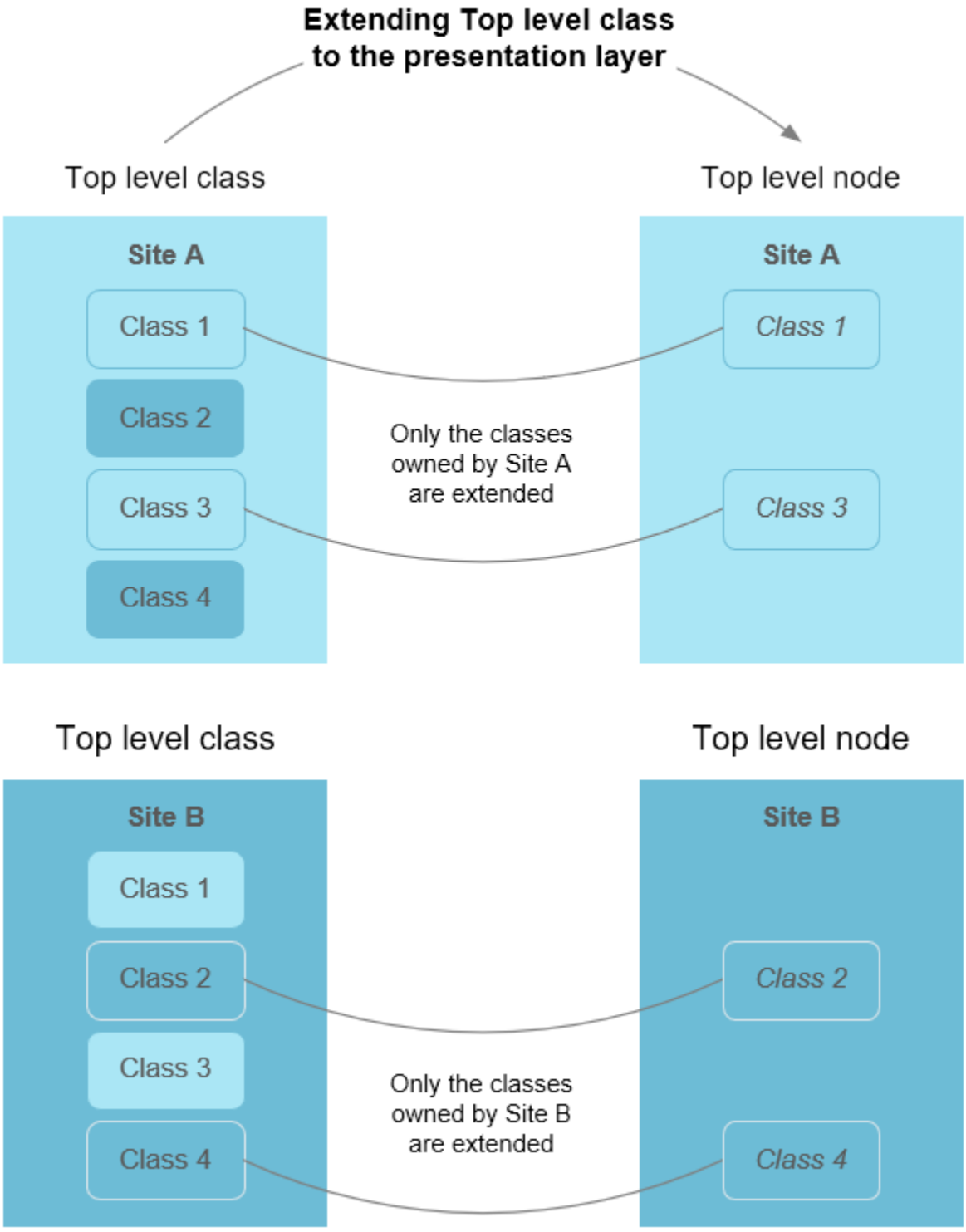
  In this scenario, there are two sites, **Site A** and **Site B**. **Site A** owns **Class 1** and **Class 3**. **Site B** owns **Class 2** and **Class 4**.



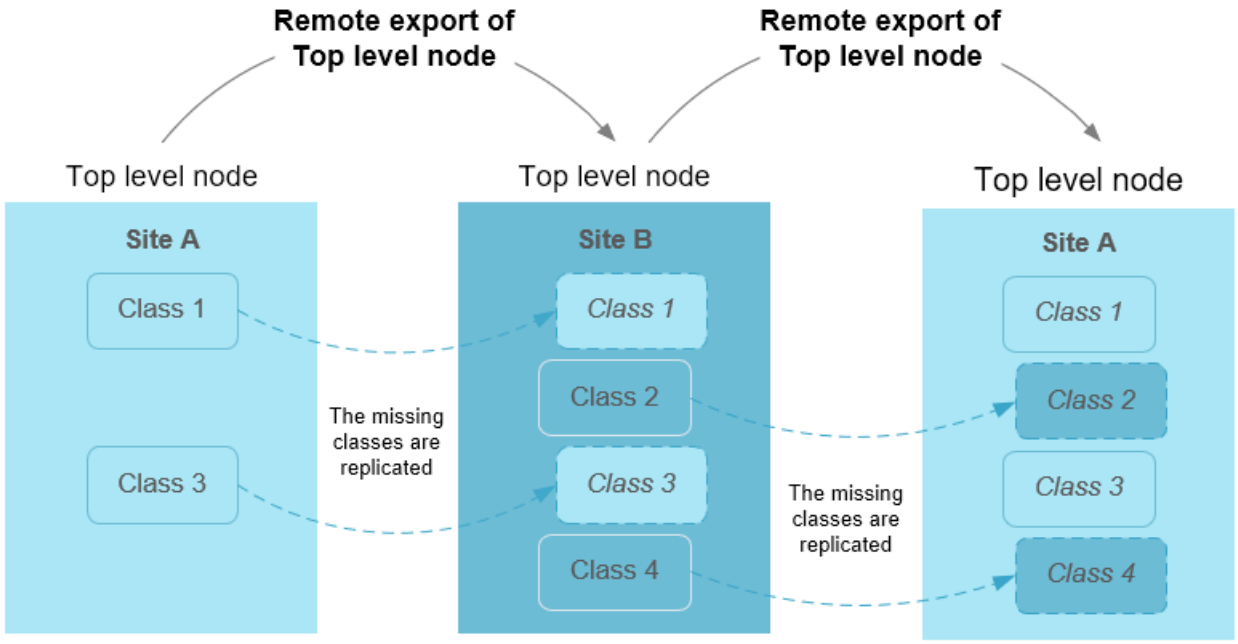
Class 1 & 3 - owned by Site A

Class 2 & 4 - owned by Site B

After extending **Top level class** to the presentation layer at each site using **clsutility**, note that only the classes owned by each site are extended.



To remedy this, you must perform a remote export of **Top level node** at each site separately. Doing this causes the all the replicated classes to be extended to the presentation layer at each site. In this scenario, after performing a remote export on **Top level node** from **Site A** to **Site B**, the presentation layer at **Site B** is filled with the missing classes. Similarly, after performing a remote export on **Top level node** from **Site B** to **Site A**, the hierarchy at that site is also fully extended:



You must perform the remote export on the top level node of any shared structures. Only the children of the node on which you perform the remote export are extended.

To ensure that all your data is replicated correctly, contact your Siemens Digital Industries Software representative to decide on the best strategy for extending your hierarchy in shared environments.

Classification preferences and utilities

lbrmanager

Performs various library management functions. Using a multitude of arguments, you can create, update, and delete data, show data, publish and retract it, as well as search for various library elements.

To use the **lbrmanager** utility, you must install the **Teamcenter Classification Collector** feature in your Teamcenter configuration.

SYNTAX

lbrmanager -u=*user-name* {**-p=***password* | **-pf=***password-file*} **-g=***group-name*

[**-create** | **-delete** | **-show** | **-find** | **-update** | **-revise** | **-copy** | **-share** | **-unshare** | **-evaluate** |

-publish | **-retract** | **-instantiate** | **-search** | **-clone** | **-import** | **-h**]

ARGUMENTS

The **lbrmanager** utility contains many arguments and subarguments whose descriptions and syntax you can find in the help contained within the utility.

- List the utility help in the customary fashion:

  ```
  lbrmanager -h
  ```

- List the utility help for subarguments as follows:

  `lbrmanager -`*subargument* `-h`

  For example:

  ```
  lbrmanager -show -h
  ```

  Running this command lists the help for all the subarguments of the **-show** command, such as the following:

```
Usage:  lbrmanager -show -libraries
[-id=<ID> | <MFK of the form: attr1=value, attr2=value...,
     object_type=boName>]
[-norecurse : Do not show the Library structure.]
[ -h | -help : This option will print usage details for this operation. ]
```

ENVIRONMENT

This utility must be run in the Teamcenter shell environment.

FILES

As specified in Log files produced by Teamcenter and the **filetypeDefaults.txt** configuration file.

The entries in the **filetypeDefaults.txt** configuration file map the extensions of all imported files to specific Teamcenter data structures: GRM relationship type, dataset type, named reference, optional default tool, and optional subdirectory name. The mapping of file extensions to specific Teamcenter data structures can be configured per import directory/subdirectory combination.

eclass2json.pl

Converts the various ECLASS formats to JSON or, alternatively, converts JSON files to BMEcat format. Specifically:

- Converts ontoML XML ECLASS hierarchy data to JSON format
- Converts BMEcat XML property record data to JSON format
- Converts JSON property records to BMEcat XML format

The **eclass2json** utility is a platform-independent Perl script that can be found in the following directory:

\\TR\bin\eclass2json

Syntax

eclass2json -inputFile: -objectType:
-h

Arguments

-inputFile:

Contains the source file for the conversion. This can be an ontoML XML file, a BMEcat XML file or a JSON file, depending on the conversion taking place.

-objectType:

Must be one of the following:

unitsmap

Creates an internal unit conversion file necessary for converting from ontoML files to JSON. This conversion is required once per ECLASS release.

definitions

Used to convert the following administrative objects in the order given:

1. Key-LOVs
2. Properties
3. Aspects
4. Blocks
5. Application classes
6. Classification classes

data

Used to convert BMEcat XML property records to JSON format.

json2BMEcat

Used to convert JSON format property records to BMEcat XML format.

-h

Displays assistance in using the utility.

Environment

This utility must be run in the Teamcenter shell environment. It is configured in the **eclass2json.properties** file contained in the same directory as the utility. The properties file contains directions on how to set the properties. In this property file you configure such things as:

- The status of the object being converted, for example, **Develop** or **Released**.
- Whether existing objects are skipped for conversion.
- Whether the converted object is to be added as a new release.
- The name of the top level node.
- The ECLASS level.
- The number of objects output in each file.

runClsAITraining

Runs the engine that trains classification data to create a model that is used for automatic class suggestions when classifying classes. Using artificial intelligence, classes are suggested based on the probability of them being an appropriate match.

Syntax

runClsAITraining -u=*user-name* {**-p=***password* | **-pf=***password-file*} **-g=***group-name*

-h

Arguments

-u

Specifies the user ID.

This is generally a user with administration privileges.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-h

Displays help for this utility.

The **-filename** argument that was available in earlier releases is no longer supported.

Environment

This utility must be run in the Teamcenter shell environment. It is found in the following directory:

*Teamcenter-root (TR)***\classification\ai**

cls_ai_auto_classify

Classifies many objects at the same time using AI. Objects classified in this way are sent to a workflow where administrators can check the validity of the AI predictions and make changes to the classification or reject the suggested classification.

Syntax

cls_ai_auto_classify -u=*user-name* {**-p=***password* | **-pf=***password-file*} **-g=***group-name*

-startDate -endDate -objectType -classifyObjects -outputPath -item-h]

Arguments

-u

Specifies the user ID.

This is generally a user with administration privileges.

-p

Specifies the password.

This argument is mutually exclusive with the **-pf** argument.

-pf

Specifies the password file.

This argument is mutually exclusive with the **-p** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-startDate

Restricts classification AI training data to objects created after the date specified.

-endDate

Restricts classification AI training data to objects created before the date specified.

-objectType

Restricts classification AI training data to objects with a specified object type.

-classifyObjects

Classifies objects that have a maximum probability above the percentage set in the **CLS_AI_Engine_Probability_Cutoff** preference.

Running the utility without this argument runs it in a dry run mode. A report containing the object, its classification, the probability, and an indication of whether the utility is capable of classifying the object, is stored in the specified location.

-outputPath

Specifies the file path where the auto-classify report is stored.

-item

Specifies a suffix for the training model output.

-h

Displays help for this utility.

Environment

This utility must be run in the Teamcenter shell environment.

smlutility

Use this command to create or delete search indexing views for classification. Classes that do not have a search index view or properties that are not included in the search index view cannot be searched for in the global search. Additionally, only properties that are included in the search index view are displayed as filters when searching.

For **smlutility** details not included below, refer to the *Teamcenter Utilities* in the Teamcenter collection.

SYNTAX

smlutility -create_indexing_views [**-u=***user-id*] [**-p=***password*] [**-g=***group*]
[**-reportfile=***file-name*] [**-listIds=***class-id*] [**-delimiter=***user-specified-delimiter*] [**-recursive**]

smlutility -delete_indexing_views [**-u=***user-id*] [**-p=***password*] [**-g=***group*]
[**-reportfile=***file-name*] [**-listIds=***class-id*] [**-delimiter=***user-specified-delimiter*] [**-recursive**]

ARGUMENTS

-create_indexing_views or -delete_indexing_views

Creates or deletes search indexing views for classification.

-u

Specifies the user ID.

This is a user with Teamcenter administration privileges. If this argument is used without a value, the operating system user name is used.

> Note:
>
> If Security Services single sign-on (SSO) is enabled for your server, the **-u** and **-p** arguments are authenticated externally through SSO rather than being authenticated against the Teamcenter database. If you do not supply these arguments, the utility attempts to join an existing SSO session. If no session is found, you are prompted to enter a user ID and password.

-p

Specifies the user's password.

If used without a value, the system assumes a null value. If this argument is not used, the system assumes the *user-ID* value to be the password.

This argument is mutually exclusive with the **-pf** argument.

-g

Specifies the group associated with the user.

If used without a value, the user's default group is assumed.

-reportfile

Specifies the path to the output file. If you do not specify this argument (or if the file cannot be opened), the output is written to **stdout**.

-listIds

Specifies a list of classes for which search indexing views should be created or deleted.

When creating, if no class ID is specified, indexing views for all classes that do not have indexing views are created. If a class contains a default view, the attributes of the default view are copied to the Search Index view. Otherwise, all the attributes of the class are set as attributes of the search indexing view.

When deleting, if no class ID is specified, indexing views for all classes are deleted.

To create Search Index views for a class and all its child class (that is, for a single branch of a hierarchy), include the topic level class ID in the **listIds** argument along with the **recursive** argument.

-delimiter

Specifies a user-defined delimiter.

-recursive

Specifies that the index view be created (or deleted) for all subclasses. This argument is used together with the **listIds** argument.

clsutility

Performs administrative tasks when the presentation layer is installed. Using a **multitude of arguments**, you can create, update, delete, and migrate data. You can also use this utility to **display the node hierarchy** in various levels of detail.

Syntax

clsutility -u=*user-name* {**-p=***password* | **-pf=***password-file*} **-g=***group-name*

[**-create** | **-delete** | **-save** | **-find** | **-ask** | **-add** | **-import** | **-search** | **-set** | **-update** | **-get** |

-remove | **-describe** | **-migrate** | **-list** | **-h**]

Arguments

The **clsutility** utility contains many arguments and subarguments whose descriptions and syntax you can find in the help contained within the utility.

- List the utility help in the customary fashion:

  ```
  clsutility -h
  ```

- List the utility help for subarguments as follows:

  `clsutility -`*subargument* `-h`

  For example:

  ```
  clsutility -create -h
  ```

  Running this command lists the help for all the subarguments of the **-show** command, such as the following:

```
USAGE

clsutility
  -u=<username>
  {-p=<password> | -pf=<password_file> }
  -g=<group>
  -output=<output file name>
  <command> <sub-command> <command options...>
  ------------------------------------------------------------
  For the command "-create" the following sub-commands are available:
    -default_objects            [Creates default Classification Context, Hierarchy and
Scheme objects.]
    -node                       [Creates a Classification Hierarchy node object.]
    -class_definitions          [Creates a Classification standard taxonomy Class.]
    -property_definitions       [Creates a Classification standard taxonomy Property.]
    -keylov_definitions         [Creates a Classification standard taxonomy Keylov.]
    -classification_object      [Creates a new instance of Classification object.]
    -classification_objects     [Creates a set of Classification objects.]
    -node_hierarchy             [Creates a Classification node hierarchy from
Classification hierarchy.]
    -node_definitions           [Creates a Classification standard taxonomy Master
node.]
    -view_definitions           [Creates Classification Standard Taxonomy View
Definitions.]
```

Environment

This utility must be run in the Teamcenter shell environment.

clsutility reference tables

All **clsutility** commands begin with the following validation:

clsutility -u=*user-name* {**-p=***password* | **-pf=***password-file*} **-g=***group-name*

Add the following to the command to perform the required actions. Optional arguments are displayed in brackets ([ ]).

All hierarchy definitions

Use this command	To
-save -classification_definitions -request=*JSON-file-name*	Create or update key-LOV, property, class, or node definitions described in the given JSON file. This argument is available beginning with schema 1.5.0. and can be used to import all previous hierarchy definitions.

Key-LOV definitions

Use this command	To
-create -keylov_definitions -request=*JSON-file-name*	Create key-LOVs described in the given JSON file.
-delete -keylov_definitions -request=*JSON-file-name*	Delete key-LOVs described in the given JSON file.
-update -keylov_definitions -request=*JSON-file-name*	Update key-LOV definition. If it does not exist, the utility creates a new key-LOV based on the data in the JSON file.
-update -status -request=*JSON-file-name*	Update the key-LOV definition status.

Property definitions

Use this command	To
-create -property_definitions -request=*JSON-file-name*	Create properties described in the given JSON file.
-delete -property_definitions -request=*JSON-file-name*	Delete properties described in the given JSON file.
-find -property_definitions -request=*JSON-file-name*	Find properties described in the given JSON file.
-update -property_definitions -request=*JSON-file-name*	Update properties described in the given JSON file.
-update -status -request=*JSON-file-name*	Update the property definition status.

Class definitions

Use this command	To
-create -class_definitions -request=*JSON-file-name*	Create classes described in the given JSON file.
-delete -class_definitions -request=*JSON-file-name*	Delete classes described in the given JSON file.
-find -class_definitions -request=*JSON-file-name*	Find classes described in the given JSON file.

Use this command	To
-update -class_definitions -request=*JSON-file-name*	Update classes described in the given JSON file.
-update -status -request=*JSON-file-name*	Update the class definition status.

Nodes

Use this command	To
-create -node_definitions -request=*JSON-file-name*	Create the hierarchy nodes described in the given JSON file. If a hierarchy node already exists, the utility updates the node with the values provided.
-delete -node_definitions -request=*JSON-file-name*	Delete the hierarchy nodes described in the given JSON file.
-update -node -id=*hierarchy-node-ID* **-name=***new-name-of-hierarchy-node* [**-descr=***new-description-for-given-node*]	Update a hierarchy node with the values provided. Update a hierarchy node with a new description.
-find -all_nodes	Find and display all nodes in the system.
-find -node -id=*hierarchy-node-ID-including-namespace*	Find and display a single node based on its ID.
-find -top_level_nodes	Display all top-level nodes in the system.
-get -classification_objects -id=*hierarchy-node-ID*	List the classification objects belonging to the given hierarchy node.
-ask -node_class_props -id=*hierarchy-node-ID*	List all properties belonging to a hierarchy node object.
-ask -node_class_props -id=*hierarchy-node-ID* **-name=***name-of-property-including-prefix*	List all properties belonging to a hierarchy node object with a given name.
-ask -node_parent -id=*hierarchy-node-ID* [**-full_hierarchy**]: lists all ancestor nodes of a hierarchy node	List the parent of a given hierarchy node object.
-ask -node_ancestors -id=*hierarchy-node-ID*	List all parents of a given hierarchy node object.
-ask -node_children -id=*hierarchy-node-ID* [**-type=***{***0***: group |* **1***: master |* **3***: all}*]	List all child nodes of a given type for the specified hierarchy node object and optionally, for the given type.
-ask -is_top_level_node -id=*hierarchy-node-ID*	Check whether a hierarchy node is a top-level node.
-ask -is_descendent_node -id=*hierarchy-node-ID* **-ancestor_id=***ancestor-hierarchy-node-ID*	Check whether a hierarchy node is a descendent of another node.
-ask -node_type -id=*hierarchy-node-ID*	Display the type of the given hierarchy node.

Use this command	To
-ask -node_instances -id=*hierarchy-node-ID*	List the classification objects belonging to the given hierarchy node, including all the objects belonging to child nodes.
-ask -node_class_props -id=*hierarchy-node-ID*	List properties belonging to a hierarchy node.
-ask -node_class_prop -id=*hierarchy-node-ID* **-name=***name-of-property-being-sought-including-prefix*	List all properties belonging to a hierarchy node.
-ask -node_attachments -id=*hierarchy-node-ID*	List all the attachments belonging to the given hierarchy node.
-ask -node_characteristic -id=*hierarchy-node-ID* **-characteristic=***{***0***: is-an-assembly* | **1***: has-multiple-instances* | **3***: is-a-leaf-node}*	List whether a given node is an assembly, has multiple instances, or is a leaf node.
-import -image -id=*hierarchy-node-ID* **-os_path=***path-to-image-file* [**-new_file_name=***new-file-name-used-by-Active-Workspace*] [**-do_update**]: updates existing primary image [**-is_primary_image**]: sets the image to primary image	Import an image for the given hierarchy node. Images are displayed in the right pane of the user interface. Teamcenter supports the following file types for class images: GIF, JPG, JPEG, PNG, BMP, SVG, and PDF.
-import -icon -id=*hierarchy-node-ID* **-os_path=***path-to-image-file* [**-new_file_name=***new-file-name-used-by-Active-Workspace*] [**-do_update**]: updates existing icon	Import an icon for the given hierarchy node. Icons are displayed in the classification hierarchy and help to visualize the class names. Teamcenter supports the following file types for node icons: GIF, JPG, JPEG, PNG, BMP, SVG, and PDF.
-remove -image -id=*hierarchy-node-ID* [**-delete_dataset**]: removes image dataset in addition to image	Remove an image attached to a node.
-remove -icon -id=*hierarchy-node-ID* [**-delete_dataset**]: removes icon dataset in addition to icon	Remove an icon attached to a node.
-get -image -id=*hierarchy-node-ID*	Identify the primary image used by the given hierarchy node.
-get -images -id=*hierarchy-node-ID*	List all the images used by the given hierarchy node.
-get -icon -id=*hierarchy-node-ID*	Identify the icon used by the given hierarchy node.

Classification objects (ICOs)

Use this command	To
-create -classification_objects -node_id=*hierarchy-node-ID* [**-obj_count=***number-of-classification-objects*]	Create multiple classification instances.

Use this command	To
[**-prefixid=***prefix-assigned-to-new-objects*] | [**-request=***JSON-file-name*]	
-find -classification_objects -request=*JSON-file-name*	Export the classification properties of the classification objects listed in the JSON request file. In the request file, the ICO object ID must be listed with its revision, for example, 123456/A.

Classification views

Use this command	To
-create -base_view_definitions	Create base views for every released application class. If a base already exists for a class, the class is skipped.
-create -base_view_definitions -class_definition *IRDI-of-class*	Create a base view for the given application class.
-create -base_view_definitions -force	Create or recreates a base view for every released application class.
	Create a user, group, or role view for a class, or modifies the order of or suppresses properties in a base view for a class.
	List the class descriptor for specified classes.

Examine the hierarchy using the clsutility -list -hierarchy utility

For most general use cases, classification admin objects are listed in the **Classification Manager**. Additionally, you can use the **clsutility**, to list classification standard taxonomy hierarchy objects to help you better understand the classification data in the database. This method of viewing the hierarchy is based on the *class descriptor*. Whereas a class definition describes only the class, or a key-LOV definition only one key-LOV, the class descriptor compiles all the definitions required to describe a classification object (ICO). Each node of the hierarchy shares a class descriptor. The class descriptor provides a view on the data from the classification consumer perspective.

> Note:
>
> When you list all the nodes in the database, the utility lists hierarchies that are not migrated to CST, but it cannot display ICOs belonging to these hierarchies.

You can direct the output to a text file that you can open in a text editor by using the **-output=***output-file-name.txt*.

The following arguments are available when using **clsutility -list -hierarchy**:

Argument	Description
-nodeId=	Specifies the unique ID of the node for which you want to display the hierarchy. The utility displays all the parents of the given node, as well as the node itself. If a node ID is not provided, all top level nodes are displayed. To view the children of the given node, additionally use the **-recursive** argument. If you use the **-nodeId** argument, you cannot use the **-classId** and **-classNamespace** arguments.
-classId=	Specifies the class ID of the class definition referenced from the given node. If you specify a class ID here, you must also specify a namespace in the **-classNamespace** argument. If you use this argument, you cannot use the **-nodeId** argument.
-classNamespace=	Specifies the namespace of the class definition referenced from the node. If you specify a namespace here, you must also specify a class ID in the **-classId** argument. If you use this argument, you cannot use the **-nodeId** argument.
-recursive	Displays the entire hierarchy below the specified node. You cannot use this argument with the **-showClass** argument.
-showClass	Shows detailed information about the referenced classes of the given node. You cannot use this argument with the **-recursive** argument.
-showJSON	Displays the JSON response from **CST_class_describe**, which is used to build the report of a class.
-showType	Displays the class definition of the dynamic runtime type. This argument is used for Siemens Digital Industries Software internal troubleshooting purposes.
-showICOs	Displays the ICOs of the nodes.
-showICOProperties	Displays all ICO properties if you have also specified **-showICOs**.
-icoLimit=	Limits the number of ICOs of a single node that are to be displayed.
-icoId=	Shows only the ICO with this ID.

- To view all the top nodes in the hierarchy, type:

  clsutility -list -hierarchy

- To view all the nodes in the hierarchy, type:

  clsutility -list -hierarchy -recursive

- To view all the nodes underneath a specified node, type:

  clsutility -list -hierarchy -nodeId=*node ID* **-recursive**

- To view the details of a specific class, type:

  clsutility -list -hierarchy -nodeId=*node ID* **-showClass**

  For example, for an ECLASS class called **AC-DC supply**, the **clsutility -list -hierarchy -recursive** command can be used to find the node ID (in this example, **0173-1#AFX043**):

```
  "27-04-06-90" [0173-1#BAB258 ==> 0173-1#01-ADP432#006] "[27-04-06-90] No-break power supply (complete, unspecified)"
}
"27-04-07-00" [0173-1#AFR649] "[27-04-07] Power supply device"{
  "27-04-07-01" [0173-1#AFX040 ==> 0173-1#01-AFR739#002] "[27-04-07-01] Continuous current supply"
  "27-04-07-02" [0173-1#AFX041 ==> 0173-1#01-AFR741#002] "[27-04-07-02] Voltage stabilizer"
  "27-04-07-03" [0173-1#AFX042 ==> 0173-1#01-AFR743#002] "[27-04-07-03] Alternating current supply"
  "27-04-07-04" [0173-1#AFX043 ==> 0173-1#01-AFR745#002] "[27-04-07-04] AC-DC supply"
  "27-04-07-05" [0173-1#AFX039 ==> 0173-1#01-AFR737#002] "[27-04-07-05] Stand-by unit"
  "27-04-07-90" [0173-1#AFX044 ==> 0173-1#01-AFR747#002] "[27-04-07-90] Power supply device (unspecified)"
}
"27-04-91-00" [0173-1#AAB638] "[27-04-91] Power supply (parts)"{
  "27-04-91-01" [0173-1#AFQ933 ==> 0173-1#01-AFR005#003] "[27-04-91-01] No-break power supply (complete, parts)"
  "27-04-91-90" [0173-1#AFZ646 ==> 0173-1#01-ADP554#006] "[27-04-91-90] Power supply (parts, unspecified)"
}
"27-04-92-00" [0173-1#AKP085] "[27-04-92] Power supply (accessories)"{
  "27-04-92-01" [0173-1#AKN587 ==> 0173-1#01-ADO360#006] "[27-04-92-01] UPS system (accessories)"
  "27-04-92-90" [0173-1#AEI674 ==> 0173-1#01-AEX753#004] "[27-04-92-90] Power supply (accessories, unspecified)"
}
```

  You can then use the node ID as input for the **clsutility -list -hierarchy -nodeId=0173-1#AFX043 -showClass** command. The output resembles the following:

```
"00-00-00-00" [0173-1#4711A1] "eCl@ass ADVANCED 4711-A1"
"27-00-00-00" [0173-1#AAB572] "[27] Electric engineering, automation, process control engineering"
"27-04-00-00" [0173-1#AAB631] "[27-04] Power supply devices"
"27-04-07-00" [0173-1#AFR649] "[27-04-07] Power supply device"
"27-04-07-04" [0173-1#AFX043 ==> 0173-1#01-AFR745#002] "[27-04-07-04] AC-DC supply"
              { Class[0173-1#01-AFR745#002] "AC-DC supply"
                Status: "Released"
                01:   [0173-1#01-ADN464#006] Aspect  "Add on Documentation"
                      { Class[0173-1#01-ADN464#006] "Add on Documentation"
                        Status: "Released"
                        01:01 [0173-1#02-AAN469#002] Integer  "number of documentation"
                              IsCardinalityController
                        02:   [0173-1#02-AAQ680#006] Reference  "Additional Information"
                              CardinalityController: 0173-1#02-AAN469#002
                              { Class[0173-1#01-ADN356#006] "Additional Information"
                                Status: "Released"
                                01:01 [0173-1#02-AAC312#001] Date  "calendrical validity from"
                                02:   [0173-1#02-AAN466#001] L10NString  "Description"
                                03:01 [0173-1#02-AAN467#003] String  "Country/region classification the document"
                                      { KeyLOV[0173-1#09-AAD528#002] String (entries:246)
                                        001: "AF"  "Afghanistan                    "  "0173-1#07-AAS015#001" ""
                                        002: "AX"  "Åland                         "  "0173-1#07-AAS156#001" ""
                                        003: "AL"  "Albania                        "  "0173-1#07-AAS157#001" ""
                                        004: "DZ"  "Algeria                        "  "0173-1#07-AAS158#001" ""
                                        005: "AS"  "American Samoa                 "  "0173-1#07-AAS159#001" ""
                                        006: "AD"  "Andorra                        "  "0173-1#07-AAS161#001" ""
                                        007: "AO"  "Angola                         "  "0173-1#07-AAS162#001" ""
                                        008: "AI"  "Anguilla                       "  "0173-1#07-AAS163#001" ""
```

  This output displays all the parent nodes of **0173-1#AFX043**, as well as all the classes (application, block, aspects) referenced by it, the properties, and the key-LOVs included in the class. The properties are listed as follows:

1. Index:internal code
2. IRDI
3. Data type
4. Unit
5. Property name

If key-LOVs are used repeatedly in a class, their details are listed the first time they are used.

```
04:02 [0173-1#02-AAN350#003] String  "Part relation to the main device"
    { KeyLOV[0173-1#09-AAD520#003] String (entries:7)
      1: "accessory                "  "accessory                "  "0173-1#07-AAR847#001" ""
      2: "counterpart              "  "counterpart              "  "0173-1#07-AAR849#001" ""
      3: "Label parts (inscription)"  "Label parts (inscription)"  "0173-1#07-AAR848#002" ""
      4: "Wartungsteil             "  "maintenance part         "  "0173-1#07-AAR846#001" ""
      5: "resolve part             "  "resolve part             "  "0173-1#07-AAR844#001" ""
      6: "spare                    "  "spare                    "  "0173-1#07-AAR845#002" ""
      7: "system component         "  "system component         "  "0173-1#07-AAR843#002" ""
    }
```

All subsequent uses of the key-LOV are abbreviated with a reference to the first usage:

```
04:02 [0173-1#02-AAN350#003] String  "Part relation to the main device"
      KeyLOV[0173-1#09-AAD520#003] { ADDITIONAL USE: For details see output above }
```

- To find the node from which you want to begin displaying the hierarchy, assuming you know the class ID and the class namespace, type:

  clsutility -list -hierarchy -classId= *class ID* **-classNamespace=***class namespace*

  If the class ID of the **AC-DC supply** class is **AFR745** and the namespace is **0173-1**, type:

  clsutility -list -hierarchy -classId=AFR745 -classNamespace=0173-1

  The output is as follows:

```
"00-00-00-00" [0173-1#4711A1] "eCl@ss ADVANCED 4711-A1"
"27-00-00-00" [0173-1#AAB572] "[27] Electric engineering, automation, process control engineering"
"27-04-00-00" [0173-1#AAB631] "[27-04] Power supply devices"
"27-04-07-00" [0173-1#AFR649] "[27-04-07] Power supply device"
"27-04-07-04" [0173-1#AFX043 ==> 0173-1#01-AFR745#002] "[27-04-07-04] AC-DC supply"
```

- To display the JSON file from which the output information is drawn, type:

  clsutility -list -hierarchy -showClass -showJSON

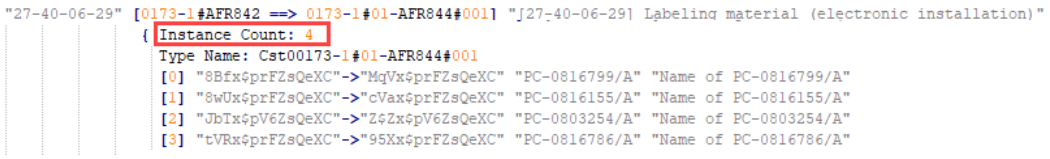
This argument functions only with the **-showClass** argument.

The output includes the description of the class as well as the JSON that the user interface uses to display the class.

- To display all the ICOs of a specific node, type:

  clsutility -list -hierarchy -nodeId= 0173-1#AFR842 -showICOs

  All the ICOs are displayed in addition to the complete hierarchy, with the number of instances displayed for each class:



  To display a specific ICO from the preceding list, type

  clsutility -list -hierarchy -showICOs -icoId=*ICO ID*

  This is particularly useful when combined with **-showICOProperties**.

- To display no more than *x* number of ICOs within a class, type

  clsutility -list -hierarchy -showICOs -recursive -icoLimit=*x*

View output in a text editor

The text files created by this utility can be extremely large. To assist you in manipulating these files, you can try enabling the collapsing of code in a text editor. This can be done, for example, by viewing the text file as C++ code, or a similar viewing feature depending on the text editor. This adds the ability to collapse at every curly bracket in the output of the hierarchy. Collapsing unwanted portions of the output makes it easier to find what you are looking for.

Setting classification preferences

The following preferences are available to configure your classification environment.

ICS_classifiable_types

Stores the business object types that can be classified. If your company works with custom business objects, you must add the item type to this preference to be able to classify it. If you want to classify, for example, items only, you must remove the **Item Revision** entry from the list of classifiable types.

AWC_classification_facets_threshold

Sets the number of search facets displayed using the preference. The default value is 200. The facets are sorted with the most common facets at the top. Exercise caution when setting the number of returned facets as a very large number can cause a decrease in performance.

CLS_is_presentation_hierarchy_active

Specifies whether traditional classes or the presentation hierarchy (nodes) are visible in the classification hierarchy. If this preferences is set to **true**, the presentation node hierarchy is visible. If it is set to **false**, traditional classes are visible. The default value of the preference is **true**.

This user preference is not delivered by default and must be added manually after installing the presentation layer (**Next Generation Classification Server** installation option).

CLS_auto_sync_node_hierarchy

Allows the system to mirror operations performed on the classification storage class hierarchy onto the node hierarchy in the presentation layer.

The default value of the preference is **true**.

CLS_search_similar_wso_props_enabled

Specifies the list of workspace object properties that are displayed in the **Filters** pane when using the **Search Similar** feature.

AWC_cls_object_filter_sorting

Defines how the classification hierarchy is sorted in the **Filter** panel. Valid values are:

- **Ascending**

  Sorts the hierarchy in alphabetical order, from A to Z.

- **Descending**

  Sorts the hierarchy in reverse alphabetical order, from Z to A.

- **Count** (default)

  Sorts the hierarchy based on the number of instances for the given filter value.

CST_supported_eclass_releases

Specifies which releases are shown in the **Release** filter when working with multiple revisions of a hierarchy.

The following preferences are required when migrating from traditional basic classes to advanced (classification standard taxonomy, CST) classes using the **clsutility -migrate -classification2cst** utility:

CST_default_namespace

Specifies the namespace of all CST objects created by the migration utility.

CST_default_migration_status

Specifies the status of the CST objects created by the migration utility. Valid values are **Develop**, **Released**, or **Retired**. Objects set to **Released** can no longer be modified.

The following preferences are used to configure classification artificial intelligence (AI):

CLS_AI_Enable_AI_Engine

Set this preference to **true** to enable class suggestions.

TC_Microservice_Base_URL

Specifies the URL of the microservice that supports classification AI. This is the same path that you specify during installation of classification AI.

CLS_AI_Object_Properties

Specifies the properties that are extracted from the Teamcenter database and used to train the AI engine. These properties form the basis for the class suggestions offered when classifying objects.

By default this property is set to:

object_name

object_desc

object_type

owning_user

owning_group

CLS_AI_Enable_Geolus

Set this preference to **true** to enable the additional comparison of shape parameters using the Geolus search engine with classification AI.

CLS_AI_Geolus_Max_Num_Results

Limits the number of results returned by the Geolus search engine. By default, this preference is set to **1000**.

CLS_AI_Engine_Probability_Cutoff

Specifies the percentage probability above which classes are considered appropriate suggestions for classification AI. If too many class suggestions are displayed or you would like to increase the accuracy of the classes displayed, you can increase the value of this preference. By default, this preference is set to **10**.

CLS_AI_Engine_Suggestions_Cutoff

Specifies the maximum number of suggestions displayed to the user. The default value is **10**. This preference is not delivered with the software. You must add it manually.

CLS_AI_Query_Start_Date

Specifies the earliest modification date of the data to include in training. Only objects created or modified after this date are included in the training. If this date is not set, there is no restriction on objects used in training.

CLS_AI_Query_End_Date

Specifies the latest modification date of the data to include in training. Only objects created or modified before this date are included in the training. If this date is not set, there is no restriction on objects used in training.

CLS_AI_Query_Object_Type

Specifies the type of data to include in training. Only objects of the given type (and any child objects of this type) are included for training. If left blank, the value of this preference is **WorkspaceObject** by default.

8. Configuring and administering basic classification on Rich Client

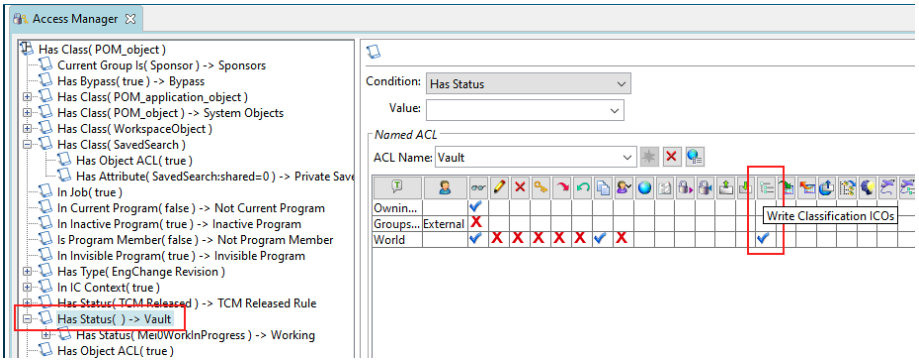
Getting started with Classification Admin

Getting started with Classification Admin

The Classification application utilizes a classification hierarchy to categorize your company's product data. As the system administrator, you use Classification Admin to define the groups, classes, and views that form the classification hierarchy. You also define and format the attributes that, when associated with a class, determine the type of information that is stored.

The Classification Admin application is best suited for creating a new hierarchy structure or for maintaining an existing hierarchy. If you are creating a large classification structure and have a form of bulk data, such as an existing classification structure that can be imported into the database, it is more expeditious to import the data.

Accessing Classification Admin	There are three ways to grant a user access to Classification Admin. • Make the user a member of the **dba** group. • Set the **ICS_admin_requires_dba** preference to **false**. Caution: This grants all users access to the application. • Grant the user access through the Authorization application.
Configure Classification Admin	Ensure that the relevant workspace objects can be classified: • All workspace object types to be classified must be listed in the **ICS_classifiable_types** preference. If your business use case requires classifying items instead of item revisions, you must remove the **ItemRevision** entry from this preference. • You must have write access to the workspace object. If this object is already released, you must have write access to the classification object (**write_ico** access). The workspace object access is set in the **Access Manager** application in rich client. The classification entries in the **Has Status() → Vault** rule must be set to write access.

	
Start Classification Admin	Click **Classification Admin**  in the navigation pane.

Classification Admin interface

Classification Admin buttons

Button	Function	Description
	Close navigation pane	Opens or closes the navigation pane that is normally displayed at the left side of the Teamcenter window. Closing this pane gives you more space to work.
	Edit	Changes the mode of your Classification Admin session, allowing you to edit the definitions of objects in the hierarchy tree.
	Create	Changes the mode of your Classification Admin session, allowing you to create objects in the hierarchy tree.
	Cancel	Clears the **Definition** pane and terminates the Classification Admin editing session.
	Save	Saves the information entered for a selected Classification instance.
	Delete	Deletes the object from the hierarchy tree.
	Import	Imports Classification object definitions from an XML file.
	Export	Exports definitions of the selected subset of Classification objects to an XML file.
	Refresh privileges	Refreshes access rules for the hierarchy tree.
	Import vendor hierarchy	Imports a tool vendor catalog hierarchy into the classification hierarchy.
	Import vendor product data	Imports tool vendor components into the classification hierarchy.
	Import translated object	Imports translated string values when working in a localized environment.

Button	Function	Description
	Export translated object	Exports string values for translation when working in a localized environment.
	Class information	Provides information about the project associated with a class, the name of the parent class, and the library in which the class is found. This information is useful when working with a data dictionary.
	Provide translations	Displays the **Language Translations** dialog box that lists existing translation values. This button appears only if special localization configurations are made.

Classification Admin tabs

Tab	Description
Hierarchy	Displays the **Classification Hierarchy** pane for accessing the group, class, and view definition panes.
Dictionary	Displays the **Dictionary** pane for creating and maintaining attribute definitions.
Key-LOV	Displays the **Key-LOV** pane for creating and maintaining lists of legal attribute values.
GCS Types	Displays the **GCS Types** pane for creating and maintaining data required by the guided component search.

Classification Admin symbols

> Note:
>
> You can customize group and class symbols. Therefore, the default symbols displayed in the following table may not represent those used at your site.

Symbol	Function	Description
	Root	Anchor of the classification system. There is one root per database.
	Group	Collection of related classes.
	Storage class	Class in which Classification instances (ICOs) are stored. Storage classes can be positioned anywhere in the hierarchy, including the leaf node position.
	Abstract class	Class used to combine common attributes for use in storage classes. Classification instances cannot be stored in abstract classes.
	View	Object that is associated with a storage or abstract class and enables access to the class attributes to be customized for specific users and groups of users.
	SML class	Class used in the legacy SML hierarchy structure to store subclasses and other classes. Instances cannot be stored in SML classes.

Symbol	Function	Description
	SML subclass	Storage container possessing a subset of the attributes assigned to an SML class. Note: Subclasses are intended for use only with existing SML hierarchy data and should not be created for new additions to the Classification hierarchy. Use abstract and storage classes instead.
	Deprecated key-LOV value	The key-LOV value is deprecated and should not be used to set an attribute value when classifying an object. However, you can use deprecated key-LOV values when searching to find objects that were previously classified with this attribute value.

Basic concepts for using Classification Admin

The following key terms and concepts apply to the creation and management of the classification hierarchy using Classification Admin:

- **Hierarchy tree**

  The hierarchy tree displays the classification structure, groups, classes, and views in a tree-like structure. The hierarchy tree helps you maintain context when performing various functions using Classification Admin.

- **Data sharing**

  Classification hierarchy data—class, view, attribute, and key-LOV definitions—can be shared with one or more remote sites, using basic *Multi-Site Collaboration* concepts.

- **Classes**

  Classes define the attributes that are stored for classified objects and determine which attributes are inherited by other classes. They are the primary building blocks of the Classification system. You can create a class within a class to build a hierarchy structure.

  Classification defines two types of classes:

  - Abstract

    Abstract classes are used primarily to gather common attributes that are inherited by child (descendant) classes.

  - Storage

Storage classes are used to stores ICOs, that can be related to workspace objects.

Classes can be flagged as assembly classes to enable the attribute values of the classified item to be propagated from individual components within the resource assembly structure in the Resource Manager application.

Classes can be nested into other classes. The only limitation on the level to which classes can be nested are those imposed by attribute inheritance.

- **Views**

  Views enable you to define how a class is displayed in the Classification application by setting attribute protections and hiding class attributes on a user, group, role, or project basis. View objects are created and managed in the Classification Admin application. The correct view is automatically set for a user in the Classification application (and cannot be dynamically changed). You can create, remove, and modify views without any impact to the Classification data.

- **SML subclasses**

  An SML subclass comprises a subset of class attributes and is used to narrow the display of SML class attributes for specific users or categories of items.

  SML subclasses inherit the attributes of their parent classes. However, unlike classes, which inherit every attribute of their parent classes and cannot be edited, you can define which inherited attributes are assigned to a subclass.

  Warning:

  Subclasses are intended for use only with existing SML hierarchy data and should not be created for new additions to the classification hierarchy. Abstract and storage classes should be used instead.

Basic tasks using Classification Admin

Use Classification Admin to do the following basic tasks:

- Create a classification hierarchy containing groups, classes, and views.
- Create and manage the attribute dictionary to store attributes for reuse.
- Create and manage the key-LOVs you need to select attribute values.
- Control access to groups, classes and ICOs through access management.
- Manipulate the appearance of the classification hierarchy.

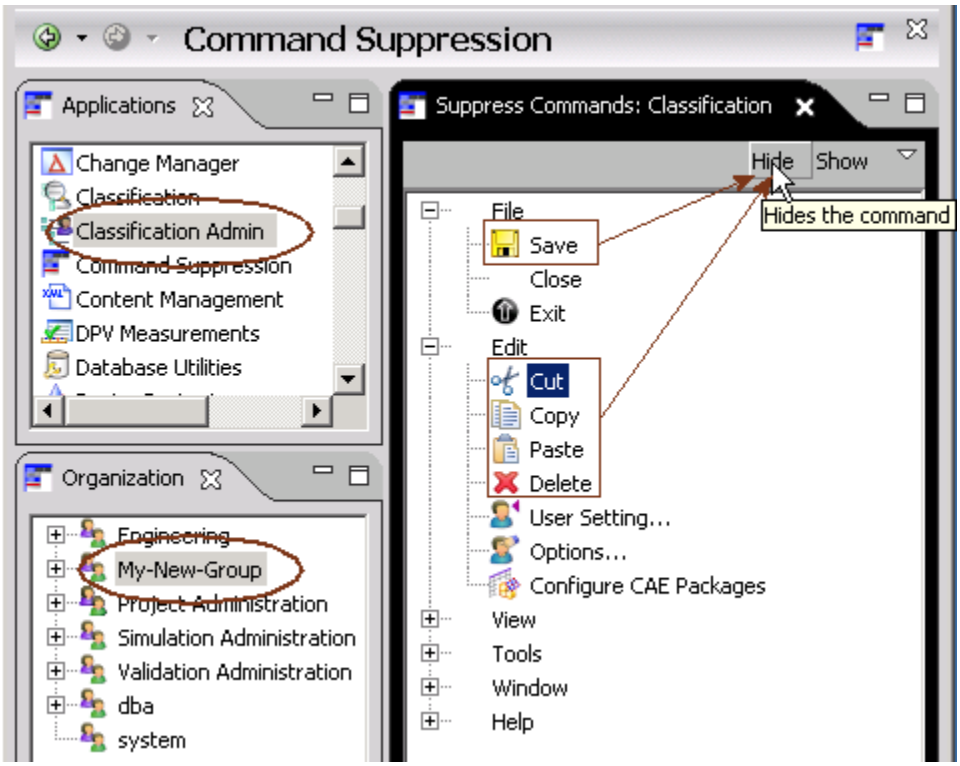
- Share hierarchy data, class, view, attribute, and key-LOV definitions with one or more remote sites.
- Import and export classification data.
- Configure the guided component search used in Resource Manager or Structure Manager.

Customizing Classification menu commands

You can hide Classification menu commands using the Command Suppression application.

If you create new groups within Teamcenter, you can hide the following Teamcenter menu commands (and buttons) that are displayed by default but not used by the Classification applications. These menu commands and buttons are hidden by default for the groups that are delivered with the software.

- **File→Save**
- **Edit→Cut**
- **Edit→Copy**
- **Edit→Paste**
- **Edit→Delete**



Specifying classifiable item types

Teamcenter stores the business object types that can be classified in the **ICS_classifiable_types** preference. If you want to classify additional item types or a new custom item type, you must add the business object type to the preference.

If you right-click an object in other applications that support Classification, you can send an item to Classification to classify it. If the item is not of a type listed in the **ICS_classifiable_types** preference, the **Send To** command is not available in the context menu.

Localizing Classification

When localizing Classification, there are several points to note:

- You must install the Classification template in TEM before you start.
- In Teamcenter, you cannot provide a translation if the master value is not set. But in Classification, you can provide a translation for a display name immediately, before saving. In view mode, you can see the translations but you cannot edit them.
- In the properties of the dictionary, you cannot edit the translations until you add a master value in the dialog box (but you do not have to save it, you simply have to type something).
- You can inherit default class values and overwrite the translations at the class level.
- You cannot enter more characters for a translation than are allowed by the string length in the dictionary.
- With inherited values, you cannot edit or view translations. You must do that at the parent object level.
- All key-LOV, all entries must have approved localized values when you log into the localized rich client before any entry's localized value is displayed. If one of the entries for the localized values is not approved, all key-LOV values display the master value, although some translations may already have an approved status.
- If localization is enabled for Classification, all Classification users can see the **Localization** button. However, Classification users must either have **dba** privileges or be granted specific translation access privileges to enter translations.
- In Classification Admin, you can export translations without translation privileges, but you must have translation privileges to import translations. Similarly, you must have translation privileges to import translations using the **I10n_import_export** utility.
- When exporting translations, the Classification user should always use the transfer modes that begin with **ICSL10N**.

- If you want to work with unit definitions in another locale, you must import the unit definition file corresponding to that locale using the **l10n_import_export** utility. The localized unit definition files are located in the following directory:

  *Teamcenter_root_directory***l10n_cots**

  For example, if you want to see German unit definitions, import using the following command:

  ```
  l10n_import_export -u=user-name -p=password -g=group-name -mode=import
  -file=L10N_classification_unit_definitions.xml
  ```

  Alternatively, you can import the unit definition file using the **Import translated objects** button in the Classification Admin application.

- When localization is enabled, attempting to import a PLM XML file containing key-LOVs that have blank values causes errors in the import and the key-LOVs are not imported. The classification attribute shows question mark symbols in the Classification application in place of the attributes. To correct this, you must edit the PLM XML file manually, adding the missing values, and run the **plmxml_import** utility again to import the updated file. This replaces the question mark symbols in the user interface with valid values.

Administering Classification

Teamcenter provides you with the following utilities to administer Classification from a Teamcenter command window:

- **icsutility**

  Imports classification data, including class definitions, attributes, and key-LOVs, as well as Resource Manager resource assemblies.

- **ics_connect**

  Associates classification objects (ICOs) with workspace objects, based on item ID.

- **smlutility**

  Updates shared classification hierarchy definitions to all sites with which they are shared.

Displaying the classification hierarchy

Displaying the classification hierarchy

The classification hierarchy shows a tree structure of nested classes. It provides an overview of all classification classes contained in the database. These classes contain a compilation of attributes related to a group of objects.

When you first open , the hierarchy on the left is still closed. You can only see the root node.

1. Click **Search by Classification Dialog or ID** to open a resource that already exists in the database.

   -or-

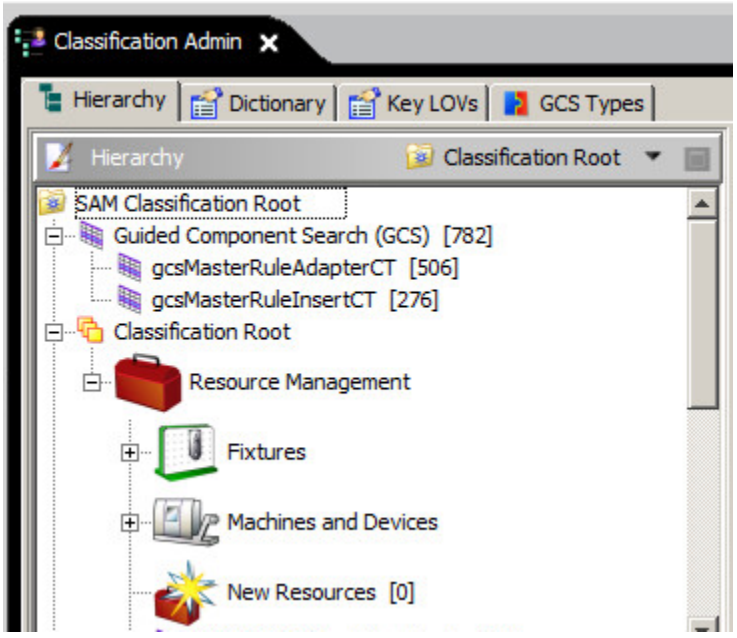
   Click **Add component to the resource** to add a component to an existing resource assembly.

2. Double-click the root node of the hierarchy.

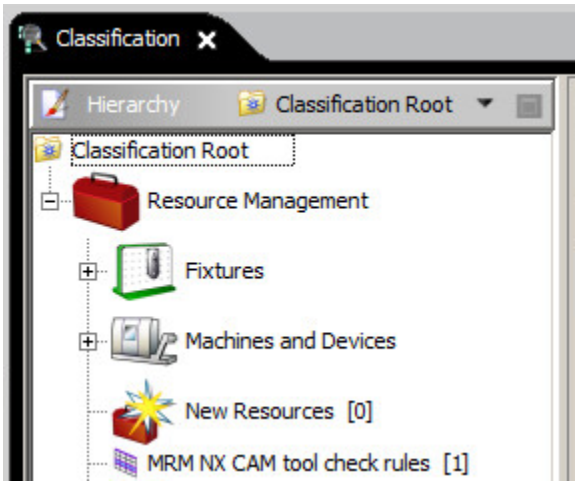
3. Move down the hierarchy by clicking the plus sign (+) in front of the name of the class you want to open. You can right-click any class and choose **Expand All** to open all child classes.

Displaying classes in the hierarchy

Classification Admin displays all classes that exist in the classification hierarchy.



Only the classes under the **Classification Root** node, however, are displayed in the Classification application, in the Classification Search Dialog, or in Resource Browser. The classes above this node, under the **SAM Classification Root** node, are administrative classes.



When running a search in Resource Browser, you may see members of administrative classes displayed in the search results.

To add, remove, or modify classification objects in administrative classes, you must move them temporarily below the **Classification Root** node, modify them, and then move them back under the **SAM Classification Root** node to protect them from modification.

View the tree graphically

You can use a graphical browser to navigate through the classification hierarchy. This browser shows a hierarchy of the tree structure symbols and allows you to enlarge them easily with a slider. You can switch back and forth between the tree structure and the graphical browser. The class that is currently highlighted in the graphical browser is then highlighted in the tree and vice versa. If you select a class in the hierarchy tree, it is also selected in the graphical browser when you open it.

1. Turn on the graphical browser using the **ICS_enable_graphical_browser** preference.

2. In the hierarchy pane, click the **Graphical Browser** tab.

3. Navigate through the classes.

   The graphical browser shows two sets of graphics. The top set, a horizontal list of graphics, is the hierarchy list. This represents the path you have taken down the classification tree from parent to child node. The second set of graphics represents all the members of the current class, that is, the last class shown in the hierarchy list. Leaf classes are displayed with a blue border around the graphic.

   - To move one level down in the tree, click the graphic.

   - To move one level up the tree, click the second-to-the-last graphic in the list of graphics at the top.

- To select a class, click the link below the graphic of the class you want to open. If the class you are selecting is a leaf node, clicking the link also selects it.
- To move to any class in the hierarchy list, click the class.
- To select a leaf class (a graphic with a blue border), click the class or click the link below the graphic.

4. Modify the graphic size by moving the slider at the right of the browser pane to the right to enlarge the graphics and moving it to the left to make them smaller.

Displaying a subset of the hierarchy

Set a node as root

In a complex classification hierarchy, you can select any point in the hierarchy and set that node as the root node, blending out all the node's ancestor classes and groups. You can store any number of classes in the options list to change the hierarchy root node at your convenience.

1. In the title bar of the hierarchy pane, click ▼ to display the root node selection. By default, this list contains only **Classification Root**. You can configure the entries in this list in the **ICS_default_root_selector_entries** preference.

2. Select the node that you designate as the new root node in your classification tree.

   Teamcenter displays the name and symbol of the new root class in the title bar.

3. (Optional) Display the full tree again by selecting **Classification Root** from the options list.

Store nodes for subsequent selection as root

1. Select the class in the hierarchy that you want as the new hierarchy root node.

2. In the title bar of the hierarchy pane, click ▼ to display the root node selection list.

3. Choose **Add Current Class**.

- or -

1. Modify the entries in the **ICS_default_root_selector_entries** preference.

These entries are then available the next time you open the root node selection list.

Delete stored nodes from the list

1. In the title bar of the hierarchy pane, click ▼.

   The root node selection list appears.

2. Right-click the class that you want to remove from the list and choose **Remove**.

Display a node as temporary root

1. Right-click the class in the hierarchy that you want as the temporary hierarchy root node and choose **Set Root Node**.

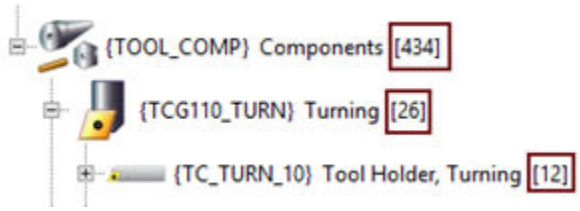
   Teamcenter displays this class as the root node but does not add the class name to the root node selection list.

2. (Optional) Return to viewing the complete tree by selecting **Classification Root** from the root node selection list.

Refresh the hierarchy tree

The classification hierarchy does not always update automatically. In certain situations, such as the following, you must update the tree manually:

- A class is moved to a new group or to see a change to the hierarchy.
- Objects are classified. The ICO count does not update automatically.



To manually refresh the classification hierarchy:

- Right-click the root node or affected branch, and choose **Refresh**.

  Teamcenter refreshes all branches beneath the selection, as well as the selection itself.

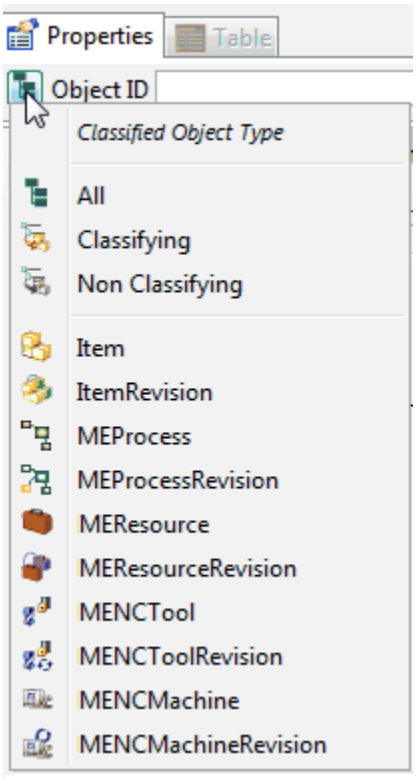
If you add attributes in Classification Admin to a class that you are currently viewing in Classification, you must refresh the class in Classification using the **Refresh** button or by selecting another class and returning to the viewed class to see the newly added attributes.

Note:

The **Refresh** menu command is not available if you are in edit mode.

Configuring the search by type

In the Classification application, you can perform a *search by type*. In this search, Teamcenter displays only matches of a certain business object type.



You can configure the entries in the search by type list in the **g4mDragIcon.searchTypes.Types** property in the **common.properties** file found in *TC_ROOT***portal\plugins\com.teamcenter.rac.tcapps_***version_number**.**jar\com\teamcenter\rac\classification\common**.

Creating and managing groups

Create a group

You use groups to organize related classes within the classification hierarchy. For example, you can create a **fasteners** group to organize all of the classes of fasteners used by your company. If the fasteners are standard parts, you can then create a **standard parts** group and associate the **fasteners** group as a subgroup of **standard parts**. In this example, the **standard parts** group is considered the

parent of the **fasteners** group. You can also modify the placement of a group within the hierarchy by changing the group's parent.

1. Click the **Hierarchy** tab.

   The system displays the **Hierarchy** pane.

2. Display the **Add New Group** dialog box by performing either of the following substeps:

   > Note:
   >
   > You can select only the root node and other group nodes as a parent group. Classes cannot be parents to groups.

   a. Choose the node in the hierarchy tree to serve as the parent of the newly created group. You can use either the **quick search** feature or the **Search Class** dialog box to quickly locate a parent group.

   b. Click the **Add Group** button in the **Group Definition** pane.

   -or-

   a. Right-click the node in the hierarchy tree to serve as the parent of the newly created group.

   b. Choose **Add Group** from the shortcut menu.

   Teamcenter displays the **Add New Group** dialog box.

   > Note:
   >
   > Only groups that you create under the ICM node are visible in the Classification application.

3. Type an ID for the new group and click **OK**.

   The group ID can comprise of up to 31 alphanumeric characters. By default, the ID cannot contain blank spaces or special characters (**|%*:(){}[] **). You can use the dash (**–**) and dot (**.**) characters as separators in the ID.

   If you want to use a special character in the ID, you can modify the **ICS_allowed_chars_for_class_id** preference.

> Note:
>
> After the group ID is assigned, the **Group Definition** pane displays information about the group, including object type, group ID, and the name of the group parent. You can now define a name for the group.

4. Type a name for the group in the **Name** dialog box.

5. (Optional) **Associate a custom symbol** with the class.

> Note:
>
> You can click the **Cancel** button at any time *prior to saving* the new group to clear the form and remove the allocated group ID from the database.

6. Click **Save** .

   Teamcenter displays the **Add New Group** dialog box.

At this point, you can **create a new class** to add to the group by clicking the **Add Class** button.

Modify a group definition

1. Choose the group in the hierarchy tree. You can use either the **quick search** feature or the **Search Class** dialog box to quickly locate the group that you want to modify. The **Group Definition** pane displays information about the group, including object type, group ID, and the name of the groups parent.

2. Click the **Edit** button . You must be in edit mode to modify a group definition.

> Note:
>
> You can click the **Cancel** button at any time *prior to saving* the modifications to clear the form and revert to the previous definition.

3. Modify the group name or parent.

> Note:
>
> Changing the parent of a group affects the group's placement within the hierarchy.

4. Click **Save** .

Delete a group

1. Choose the group or subgroup in the hierarchy that you want to delete. You can use either the quick search feature or the Search Class dialog box to quickly locate the group that you want to delete. The **Group Definition** pane displays information about the group, including object type, group ID, and the name of the group's parent.

2. Click the **Edit** button .

3. Click the **Delete** button .

   If the group is not referenced, Teamcenter deletes it immediately. If it is referenced (contains child groups or classes), Teamcenter displays the **Remove** dialog box.

4. (Optional) Click **Show Details**.

   Teamcenter displays the groups and classes in the hierarchy to be deleted.

5. Select the **Remove hierarchy including data** option.

6. Click the **Yes** button to confirm.

   Teamcenter deletes the group and all its child groups and classes (including ICOs) from the hierarchy structure.

   > Note:
   >
   > If you are using Multi-Site Collaboration, you must first remove the remote sites from the list of shared sites for the group and its classes and delete the group at all sites.

Add an image to a group

Images can be associated with the definitions of hierarchy objects to help identify the contents of a group or class. Images should be generic to adequately represent the variations in the contents of the group or associated classes.

1. Right-click the node in the hierarchy tree and choose **Add Image**.

   Teamcenter displays the **Open** dialog box.

2. Locate and choose an appropriate image file to be associated with the group or class and click **Open**.

   Teamcenter displays the image in the image viewer.

Note:

If the image does not immediately load in the image viewer, reselect the node in the hierarchy. This action starts the image loading process.

Display information in the Properties pane

You only see information in the **Properties** pane once you select a class in the hierarchy.

- Double-click a storage class in the hierarchy.
- Right-click a class from the hierarchy and choose **Select**.

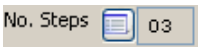
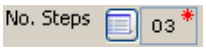
Viewing attribute values

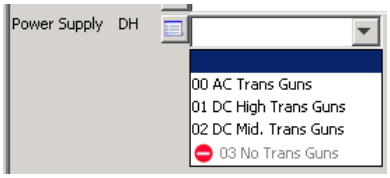
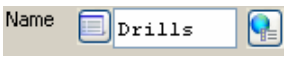
Attributes are placeholders for values that distinguish one instance of a class from another. For example, within the **Sheet Metal Screws** class, the **length**, **diameter**, and **thread** attributes are used to distinguish one sheet metal screw from another. Attributes and their values are displayed in the **Properties** pane.

In addition to the list of attributes that is displayed when you select a class, your administrator may have added custom attributes that are listed in a tool tip in the **Properties** pane.

There may be restrictions set by your administrator on what attribute values you can modify or what values you can enter. For example, an attribute value may not be modifiable because your administrator has set a default value, or you can only enter attributes between a certain range. In addition, the display of the values can be changed. For example, by modifying the **ICS_display_unformatted_numbers** preference, you can add or remove the leading zeros.

Teamcenter provides you visual aid to see the restrictions set on an attribute. If you are in edit mode, you may see the following indicators.

Symbol	Restriction
	Attribute property is set to **Protected** or a default value is set to **Fixed**.
	Attribute property is set to **Mandatory**. Whether you can save an ICO without setting mandatory attribute values is controlled by the **ICS_force_mandatory_attribute_check** preference.
	Attribute property is set to **Mandatory** and **Protected**.

Symbol	Restriction
	Attribute value is set to **Auto Computed**. Note the gray frame. This value is calculated automatically based on external logic.
	Attribute value is displayed in centimeters. To display the value in a different unit, click the unit and select the desired display unit. This does not change the storage unit in the database. If the unit is not hyperlinked, it means that Teamcenter cannot find the given unit in the **Unit Definition** class. Notice that the **cm** entry is in bold type. This indicates that the storage unit for this attribute is centimeters. Tip: Type the unit in the attribute value box along with the value. Teamcenter changes the unit for you automatically.
	Attribute value is restricted to a specific range.
	Attribute value falls outside of allowable range.
	Key-LOV value **03 No Trans Guns** is deprecated and should no longer be used.
	Attribute value is localized. Click to enter or view values in other supported languages.

Creating and managing classes

Creating and managing classes overview

Workspace objects are classified by choosing a class from the hierarchy and providing values for the attributes of the class. Teamcenter Classification defines two types of classes: *storage classes* and *abstract classes*. Abstract classes collect attributes that are common to a number of different classes that are, in turn, defined as children of the abstract class. Both types of classes are used to create the hierarchy and both types of classes can be created without any associated attributes. Attributes are inherited from any and all parent classes and cannot be removed from the class definition. However, you can add attributes to any class *that has not been designated as the parent of another class*.

Classification instances cannot be stored in abstract classes. Instances are stored in storage classes that can hold both parent and child positions in the hierarchy.

As an example of how abstract and storage classes are used, consider the screws used to manufacture a product. All screws possess common attributes such as **thread diameter** and **length**. They may also possess unique attributes, such as **head type**.

An administrator creates the **screws** abstract class containing the common attributes, thus ensuring that each child class automatically inherits the **thread diameter** and **length** attributes. To classify the screws according to their unique head types, you create separate storage classes and each storage class includes an attribute representing the unique head type.

Note:

A class can contain a maximum of 200 attributes, both inherited and local.

Classifying a single workspace object multiple times

In the Classification application, a single workspace object is classified multiple times in different classes by default and the attribute values are synchronized between the instances. For example, if **part 1** is classified in **class 1**, **class 2**, and **class 3**, and all of these classes contain the attribute **vendor**, if the value for **vendor** is changed in the instance stored in **class 1**, the value is also updated in the instances stored in **class 2** and **class 3**.

The **Allow Multiple Instances** option in the **Class Definition** pane modifies the default behavior to allow a single workspace object to be classified multiple times within the same class. Given that attribute values are synchronized between multiple instances of a single object, when this option is active, users can potentially create instances that are exact duplicates. Therefore, Siemens Digital Industries Software

recommends that you disable synchronization of at least one attribute by applying the **Local Value** property. This property is applied on an attribute-by-attribute basis at the class level. In addition to preventing synchronization between multiple instances of a single object stored in the same class, the **Local Value** property also prevents synchronization of values between multiple instances of a single object stored in different classes.

Create a class

The process of creating a class consists of identifying the class in the database, assigning attributes to the class, assigning properties to the attributes, and saving the class definition.

1. Click the **Hierarchy** tab.

   Teamcenter displays the **Hierarchy** pane.

2. Display the **Add New Class** dialog box using one of the following methods:

   a. Choose the group, abstract class, or storage class in the hierarchy tree that will be the parent of the new class.

   b. Click the **Add Class** button located at the bottom of the class definition pane.

   -or-

   a. Right-click the node in the hierarchy tree that will serve as the parent of the newly created class.

   b. Choose **Add Class** from the shortcut menu.

   Teamcenter displays the **Add New Class** dialog box. You use this dialog box to create and reserve an ID for the new class in the database.

3. Enter a class ID in the **Add New Class** dialog box and click **OK**.

   The class ID can be comprised of up to 31 alphanumeric characters. By default, the ID cannot contain blank spaces or special characters (**|%*:(){}[] **). You can use the dash (**–**) and dot (**.**) characters as separators in the ID.

   If you want to use a special character in the ID, you can modify the **ICS_allowed_chars_for_class_id** preference.

Note:

- Once a class ID is assigned, it cannot be modified. After the class ID is assigned, the pane displays information about the class, including object type, class ID, and the name of the parent group or class.
- If you build queries using property finder formatter objects, you cannot include the dot character in class names.

4. Type a name for the class in the **Name** box.

5. Click **metric**, **nonmetric**, or **both** to specify the system of measure to be applied to the attributes of the class.

   You can set which measurement system Teamcenter displays as the default measurement system in the **ICS_unit_default** preference.

6. Specify whether the class is abstract or storage. You can use abstract classes to combine common attributes for use in storage classes, but you cannot store classification instances (ICOs) in them. You store instances in storage classes. You can create these anywhere in the hierarchy, including the leaf node position. By default, new classes are abstract. To create a storage class, clear the **Abstract** option.

7. (Optional) Select the **Allows multiple Instances** check box to enable users to create multiple instances of the same workspace object with in a single class.

8. (Optional) Choose the **Assembly** option if any of the attribute values in this class propagates from individual components within a resource assembly structure. When defined as an assembly class, the Resource Manager application allows the attribute values of the classified item to be propagated from individual components within the resource assembly structure.

9. (Optional) Select **Prevent remote ICO creation** to allow creation of ICOs at the owning site only.

10. (Optional) Create an alias name for the class. Classification takes all alias names as well as class names into consideration when performing a search. You can specify any number of alias names and these appear in the tool tip of the class node in the classification tree.

11. (Optional) Specify the language of the alias by selecting the language from the list below the **Alias Names** box. The languages that appear in this list are those specified in the **ICS_available_languages** preference.

12. (Optional) Enter text in the **User Data 1** and **User Data 2** boxes. Any values entered in the **User Data 1** or **User Data 2** boxes are shown in the attribute's tool tip in the Classification or Resource Manager applications. Alternatively, you can specify the location of the Java code required to support a **user-defined button**.

13. (Optional) In the **Multi-Site Collaboration** section, choose the sites that you want to share this class definition.

14. (Optional) **Associate a custom symbol** with the class.

> Note:
>
> You can click the **Cancel** button at any time *prior to saving* the new class to clear the form and remove the allocated class ID from the database.

Assign attributes to a class

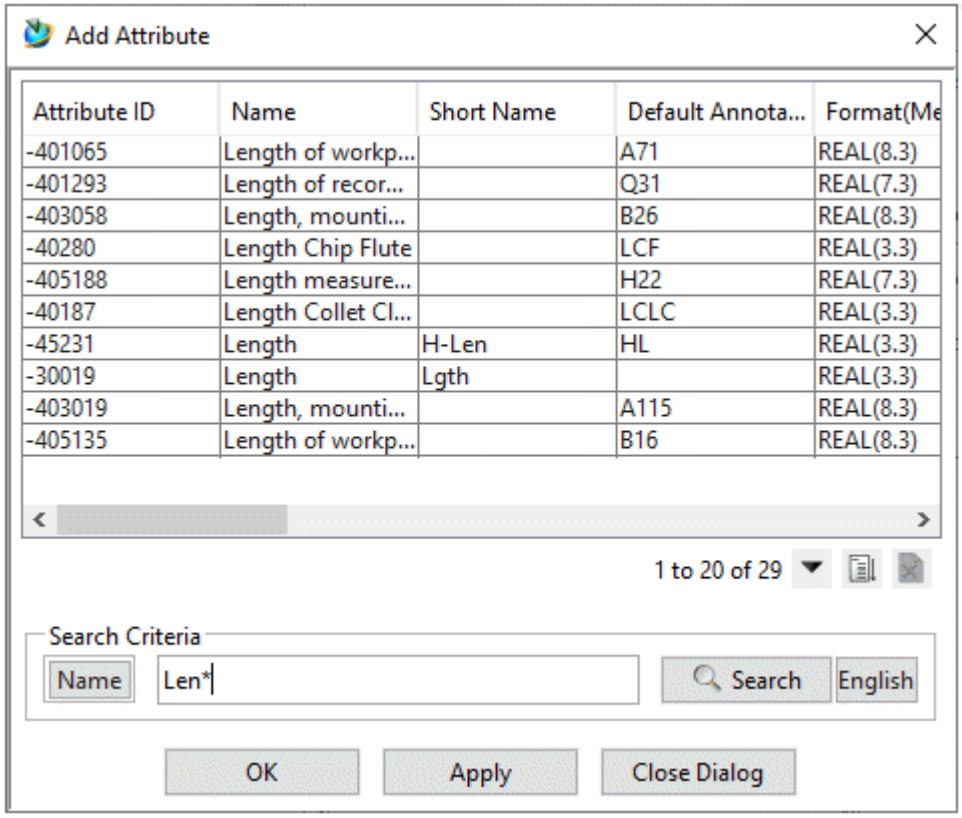
Attributes are either inherited or directly associated with the class. The **Inherited Attributes** list displays all of the attributes that are inherited from any and all parent classes. Inherited attributes cannot be deleted or modified. You can, however, add attributes to the definition of the class. Attributes added to the class become inherited attributes of any child classes. A class can contain a maximum of 200 attributes, both inherited and local.

You can **arrange the layout of class attributes** when you create a view for the class.

> Tip:
>
> If you simultaneously view a class in Classification and add attributes in Classification Admin, you must refresh the class in Classification before you can see the new attributes.

1. Click the **Add Attribute** button.

   Teamcenter displays the **Add Attribute** dialog box.



2. Locate the attributes that you want to add to the class by **performing a search of the attribute dictionary**.

   Teamcenter displays the search results in the attribute table.

3. Choose the attributes in the table that you want to add to the class. You can select multiple attributes in contiguous or noncontiguous rows by using the shift and control keys.

4. Add the selected attributes to the **Class Attributes** list by performing one of the following steps:

   - Click **OK** to accept the attribute selections and dismiss the dialog box.

   -or-

   - Click **Apply** to accept the attribute selections and retain the dialog box for further use. At this point, you have identified the new class and assigned class attributes. You can now assign an annotation and/or properties to individual attributes. Properties that are assigned to attributes at the class level do not change the definition of the attribute itself.

   > Note:
   >
   > The **Attribute ID**, **Name**, and **Format** boxes are display-only and cannot be modified.

The new attributes are displayed automatically in default views. If you have created custom views, you must explicitly add the new attribute to these views in one of two ways:

- Manually by **editing the view**.
- Automatically by using the **smlutility** utility.

Apply properties to attributes

1. On the **Class Attributes** tab, choose the attribute from the **Class Attributes** list to which you want to add an annotation or property.

   Teamcenter displays the properties of the attribute in the **Attribute Details** pane.

2. Enter an annotation value in the **Annotation** box.

   If the attribute has an annotation value assigned to it in the attribute dictionary, this value is displayed in italics in the **Class Attributes** pane. You can overwrite it. If you type a value that is the same as the dictionary value, Teamcenter recognizes this and displays the value in italics again.

   The annotation value must be unique within the class.

   > Note:
   >
   > If localization is enabled in your installation, Teamcenter displays the localization button  beside properties for which you can specify multiple languages.

3. (Optional) Type information into the **User Data 1** and **User Data 2** dialog boxes.

   If the attribute has **User Data 1** and **User Data 2** values assigned to it in the attribute dictionary, these values are displayed in italics in the **Class Attributes** pane. You can overwrite them. If you type a value that is the same as the dictionary value, Teamcenter recognizes this and displays the value in italics again.

   User Data 1 and **User Data 2** values are shown in the attribute's tool tip in Classification or Resource Manager. Alternatively, you can use these boxes to store the location of the Java code used to **create a user-defined button**.

4. (Optional) Set a default value by selecting the **Default Value** check box. You can:

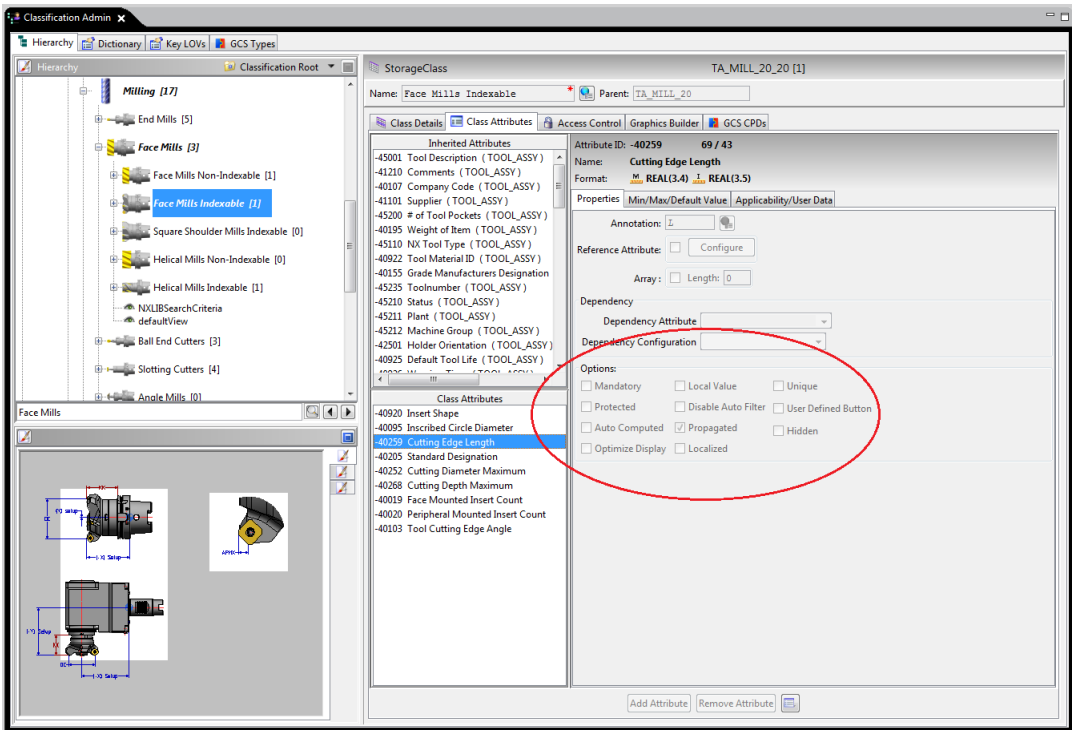
   - Enter a default value into the **Default Value** box.
   - Select **Fixed** and type a value to make this attribute value unchangeable in the class.

- Clear **From Dictionary** and type a new value to overwrite a default value that is inherited from the attribute dictionary.
- Select **From Dictionary** to resume inheriting the default value that is set in the attribute dictionary.
- Enter default values for both metric and nonmetric systems of measurement.

5. Assign minimum and maximum allowable values for the attribute by selecting the **Minimum Value** and **Maximum Value** check boxes and typing the values. If you want to use minimum and maximum values set in the attribute dictionary, select the **From dictionary** check box. Entering a value at the class level overrides any values set at the dictionary level. Use the following preferences to determine whether a Classification user must adhere to these ranges.

Set this preference to true	To
ICS_enforce_min_max_ constraints_on_update	Prevent Classification users from saving or updating classification instances or classified objects whose attribute values are beyond the specified range.
ICS_enforce_min_max_ constraints_on_copy	Prevent Classification users from creating a copy of an existing classification instance or classified object containing attribute values beyond the specified range. This includes revising or using the **Save As** command on an item or item revision outside the Classification application.
ICS_allow_min_max_ range_override	Allows the Classification administrator to specify maximum or minimum values in the class or view that are outside the range set in the attribute dictionary.

6. Select one or more properties to apply to the attribute. You can associate any of the following properties with a class attribute:



- **Reference**

  Values for **reference attributes** are stored in the workspace object and not in the ICO. They are retrieved while reading an ICO.

- **Array**

  Attributes designated as array can be used to store multiple values in the database. The number of values displayed is defined by the class.

  > Note:
  >
  > The total amount of characters allowed in the array is 256. Due to internal restrictions, each text field requires an additional two characters. Therefore, the total characters of an attribute value is calculated by:
  >
  > Format length of the attribute x VLA (variable length array) length + VLA length x 2 (the format length of a key-LOV is always 31).

  > Example:
  >
  > 31 * 7 + 2 * 7 = 231 (valid)
  >
  > 31 * 8 + 2 * 8 = 264 (exceeds limit of 256)

- **Mandatory**

Specifies that a value must be entered for the attribute in the Classification form. Mandatory fields are designated by a red triangle in the upper-right corner of the field.

Classification always checks that mandatory fields are filled in when performing any action in the user interface. For all other actions, whether the check takes place is controlled by the **ICS_force_mandatory_attribute_check** preference.

- **Unique**

  Specifies that the value entered for the attribute must be unique. When displayed in the Classification form, unique fields are designated by a blue triangle in the lower-right corner of the field.

- **Protected**

  Attributes designated as protected cannot be edited in the Classification form. You can, however, use a protected attribute to perform searches.

- **Hidden**

  Attributes can only be defined as hidden if they are used in association with custom logic.

- **Auto Computed**

  Specifies that values for this attribute are calculated based on other attribute values using external custom logic.

- **Local Value**

  Designates that values for this attribute will not be synchronized when a single workspace object is classified multiple times. This is particularly important when the **Allows multiple Instances** option is selected, because synchronizing the attribute values of multiple instances of an object that are stored in the same class can result in duplicate instances.

  You can also apply the local value property to prevent synchronization when classifying a single object in multiple classes. For example, if instances of **Part 1** are stored in **Class 1**, **Class 2**, and **Class 3**, and all of these classes contain the attribute vendor, the **Local Value** property can be applied to the attribute in **Class 1** to prevent the value from being synchronized if it is changed in the instances stored in **Class 2** or **Class 3**.

- **Disable Autofilter**

  Turns off the autofilter on a per class basis. To turn the filter on and off globally, use the **ICS_show_autofilter** preference.

- **User-Defined Button**

Creates a user-defined button for the class that you can use to programmatically access information stored outside the currently selected class.

- **Propagated**

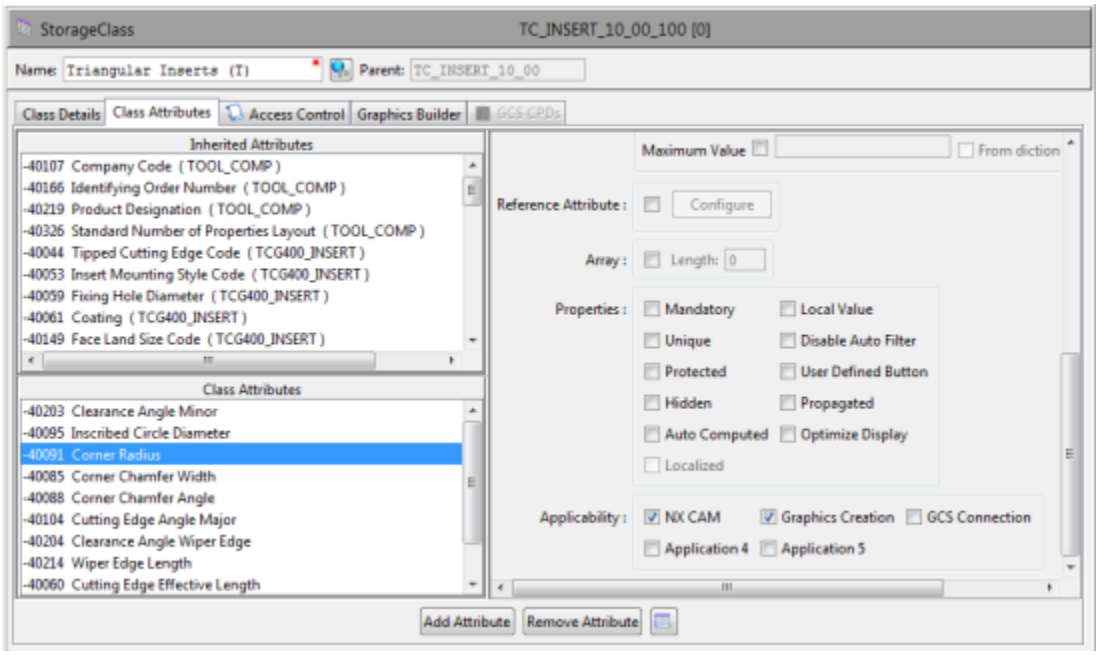
  Available only when the **Assembly** option is selected. Specifies that the value for this attribute is propagated from a component within the resource assembly structure. The component from which this value is propagated depends on the location of the propagation start point in the resource structure. If this option is not selected, the attribute's value is a base value for the assembly class.

- **Optimized Display**

  Provides the most readable unit with the least number of leading or trailing zeros.

Specify the applicability of attributes

For some applications, you may want to highlight a specific set of attributes that are relevant to that application. If a class contains a long list of attributes, it is often difficult to spot these attributes in the Classification application or Classification Search Dialog. You can specify for which applications or uses an attribute is applicable in the Classification Admin application.



Once you specify the applicability in the Classification Admin, you can then choose to highlight these attributes in applications that display them, such as Classification or the Classification Search Dialog.

1. (Optional) Modify the default names of the applications displayed in the **Applicability** section of the **Class Attributes** pane in Classification by specifying them in the **common_locale.properties**

file located in the **classification** directory in the **com.teamcenter.rac.tcapps plugin** or in the **ICS_application_names** preference.

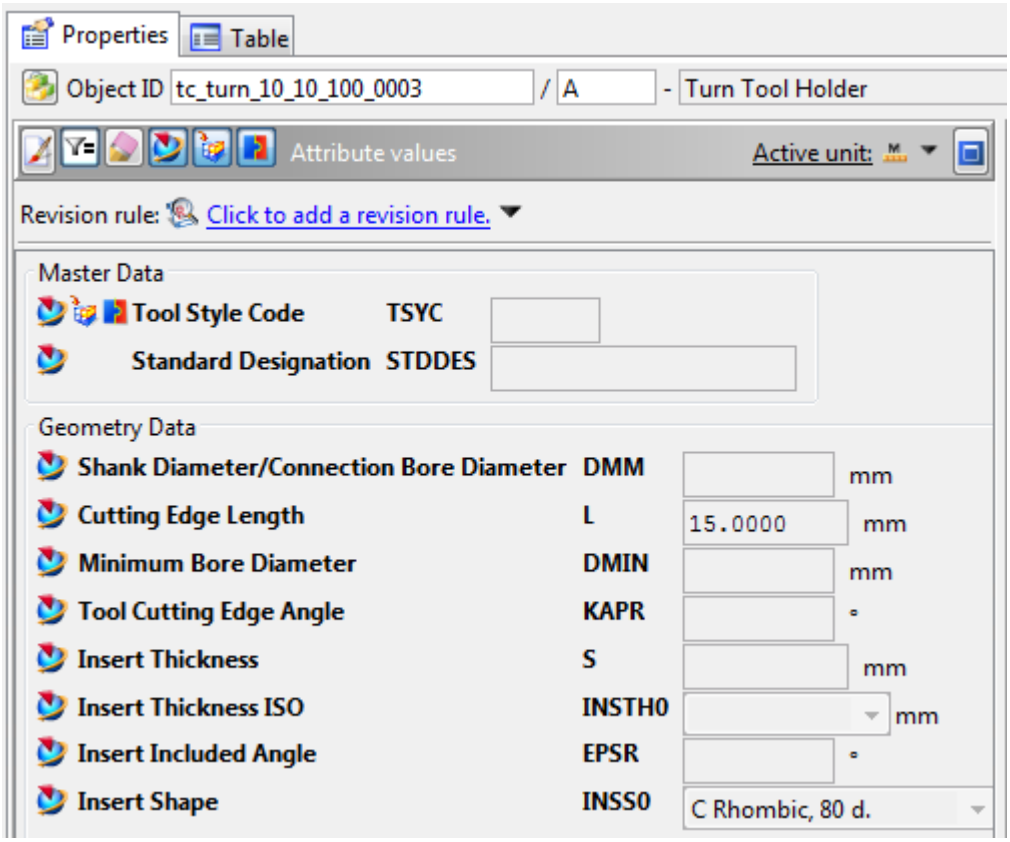
> Note:
>
> If you modify the application names in the preference, you must specify all five of them. If you specify fewer than five, Teamcenter does not recognize the preference.

If you modify the properties file, Teamcenter takes these names into consideration when localizing the environment. If you modify the preference, the application names are visible to all users.

2. (Optional) Change the symbol shown in the Classification application in the **common.properties** file located in the **classification** directory in the **com.teamcenter.rac.tcapps plugin**.

3. When creating an attribute, select the application for which this attribute is relevant in the **Applicability** section of the **Class Attributes** pane.

4. In Classification or the Classification Search Dialog, in the **Properties** pane, click the application button.



Teamcenter displays the application symbol beside all those attributes that are relevant to that application.

Remove an attribute from a class

The process of removing attributes from a class is identical to modifying class properties. If you delete an attribute that is already referenced in child classes or views, Teamcenter deletes the attribute from all views of all child classes and deletes the value of the attribute from any existing ICOs containing this attribute.

1. Select the class in the hierarchy tree.
2. Click the **Edit** button on the toolbar.
3. Select the attribute that you want to delete from the **Class Attributes** list.

   > Note:
   >
   > You cannot remove inherited attributes from a class.

4. Click the **Remove Attribute** button.
5. After you complete removing attributes from the class, click **Save**.

   Teamcenter saves the modified class definition to the database.

   > Caution:
   >
   > If you attempt to remove attributes that are already referenced, Teamcenter displays a warning message that it will delete the existing attribute from all the views and clear the attribute value from all ICOs. At the moment you accept this warning by clicking **Yes**, Teamcenter performs those actions, and you cannot subsequently revert them by undoing the **Edit** mode. The actions are irreversible.

Setting a default value for a class attribute

You can set a default value for a class attribute. Teamcenter then pre-populates the attribute value's box with this default value when you create an ICO in that class. If you copy or reclassify ICOs, empty attribute values are populated with the default value. If you specify that the default value is **Fixed**, you cannot change the value during any ICO operation (such as create, copy or reclassify).

If a default value is already defined for this attribute in the dictionary, then the value is inherited to the class, and the **From Dictionary** option is selected. You can overwrite the dictionary definition with a new value or remove the default value for this particular class.

If you set a default value, this value is applied when using ITK functions or during import.

Be careful when setting default values for interdependent key-LOVs. Consider the following interdependent key-LOV for country and city:

- **USA**
  - **New York**
  - **San Francisco**
- **Germany**
  - **Munich**
  - **Cologne**

If you have a class containing two attributes, **Country** and **City**, although it is technically possible to set default values for each of these resulting in conflicting entries (i.e., **Country** = **USA** and **City** = **Munich**), it does not make sense to set those values.

List attribute values

1. Select a class.
2. Click the **Edit** button .
3. Select an attribute.
4. In the **Class Attributes** pane, click **List attribute values of the current class** .

   Teamcenter displays the **List of Values** dialog box that shows all the values assigned to the selected attribute within the selected class. In addition, it displays the number of times each value is assigned.

Save the class definition

Save the class definition by clicking **Save** .

At this point, you can add a new nested class to the class by clicking the **Add Class** button and then creating a class, assigning attributes to the class, and applying properties to the attributes.

You can add a subordinate view with all or a subset of the class attributes to the class by clicking the **Add View** button.

Modify a class

> Caution:
>
> While it is possible to modify the properties of a class, you should approach such changes with caution. Changing class properties can compromise the integrity of your Classification data.

1. From the hierarchy tree, choose the class that you want to modify. The **Class Definition** pane, located to the right of the hierarchy tree, displays the details and attributes of the class.

2. Click the **Edit** button on the toolbar to activate the pane.

3. (Optional) Modify the attributes, attribute properties, Multi-Site Collaboration sharing, or parent class.

4. Click **Save** on the toolbar. Teamcenter saves the modified class definition to the database.

> Note:
>
> You can click **Cancel** at any time *prior to saving* the modifications to clear the form and revert to the previous definition.

Copy a class

You can copy a class with or without its child classes within the hierarchy. An almost identical copy of the original class (hierarchy) is created under the selected parent. The differences between the original and the copied class are:

- The copy has a different class ID than the original class. This applies to all children if a hierarchy is copied.
- The copy has no shared sites.
- The copy contains no instances, part family templates, or connection point definitions.

1. Select the class that you want to copy in the hierarchy.

2. Either drag this class to the new position or cut it and paste it into the new position in the hierarchy using the **Copy** and **Paste** commands in the shortcut menu.

   The **Copy Class** dialog box appears.

3. Enter a new ID into the **New Class ID** box. If you leave this box empty, the system automatically generates a new ID based on the value you enter in the **ICS_copy_classid_pattern** preference.

4. (Optional) Assign the attributes that are inherited in the original class to the new class by selecting **Insert inherited attributes**. Click **Show** to display these attributes. The inherited attributes are added as new attributes to the copied class at the beginning of the attribute list (**Case 1**). If you choose not to assign inherited attributes to the class, the copied class contains only the attributes of the original class (**Case 2**). Attributes that are inherited from the new parent are deleted from the copied class.

```
Group 1
    Abstract Class 1 (Atts. -1000, -1001)
      Original Storage Class 1 (Atts. -1002, -1003, -1004)

Group 2
    Abstract Class 2
      Copy of Storage Class 2 (Atts. -1000, -1001, -1002, -1003,
-1004)
```

Case 1 – Copying a class with inherited attributes

```
Group 1
    Abstract Class 1 (Atts. -1000, -1001)
      Original Storage Class 1 (Atts. -1002, -1003, -1004)

Group 2
    Abstract Class 2
      Copy of Storage Class 2 (Atts. -1002, -1003, -1004)
```

Case 2 – Copying a class without inherited attributes

```
Group 1
    Abstract Class 1
      Original Storage Class 1 (Atts. -1002, -1003, -1004)

Group 2
    Abstract Class 2 (Atts. -1002, -1003)
      Copy of Storage Class 2 (Atts. -1004)
```

Case 3 – Copying a class where attributes are inherited from new parent

5. (Optional) Copy the class hierarchy. When you do this, Classification Admin makes copies of all the child classes underneath the selected class. To assist you in copying over the hierarchy, the **Copy Class** dialog box expands with additional options. The **Target Class Hierarchy** dialog box displays how the copied class hierarchy appears. You can modify the new hierarchy in this dialog box by:

   - Inserting a new item ID and pressing the Enter key.

The new ID appears in the **Target Class Hierarchy** box. If you leave this box empty, the system automatically generates a new ID based on the value you enter in the **ICS_copy_classid_pattern** preference.

- Removing classes that should not be copied by selecting the class and clicking ✕.

6. Click **OK**.

The selected class is copied to a new position in the class hierarchy.

Moving a class

You can use drag-and-drop to move a class to a new location in the hierarchy. This action moves the class and all of its children and instances. During moving:

- Attributes that were originally inherited are added as new attributes to the moved class.
- Attributes inherited from the new parent class are deleted from the moved class.

The following restrictions apply to moving a class:

- The class cannot be shared.
- All children of the class must be readable.
- The new parent class must be locally owned.

> Caution:
>
> While it is possible to move a class, you should approach such a change with caution. Moving a class can compromise the integrity of your Classification data.

Delete a class

> Caution:
>
> While it is possible to delete a class, you should approach such a change with caution. Deleting a class can compromise the integrity of your Classification data.

1. From the hierarchy tree, select the class that you want to delete.
2. Click the **Edit** button on the toolbar.
3. Click the **Delete** button ✕ on the toolbar.

If the class is not referenced, Teamcenter deletes it immediately. If it is referenced (contains child classes), Teamcenter displays the **Remove** dialog box.

4. (Optional) Click **Show Details**.

   Teamcenter displays the classes in the hierarchy to be deleted.

5. Select the **Remove hierarchy including data** option.

6. Click the **Yes** button to confirm.

   Teamcenter deletes the class and all its children (including ICOs) from the hierarchy structure.

> Note:
>
> If you are using Multi-Site Collaboration, you must first remove the remote sites from the list of shared sites for the class and delete the class at all sites.

About deep copy rules

Deep copy rules define whether objects belonging to a business object instance can be copied when a user performs a save as or revise operation on that instance. Deep copy rules can be applied to any business object type and are inherited by children business object types.

When you set deep copy rules for **Revise** and **Save As** operations on classification objects (**IMAN_classification**), observe the following:

- Only the **CopyAsObject** or **NoCopy** actions are permissible. A **CopyAsReference** action would corrupt your data.

- If the presentation hierarchy is installed and you do not want classification objects included in the copy action, then you must disable both the **IMAN_classification** and **Cls0ClassifiedBy** relationships.

For more information about deep copy rules, see *BMIDE for Data Model Design*.

Managing units of measurement

When creating classification classes, you can define whether a class contains only metric ICOs, only nonmetric ICOs, or both. If the classification administrator specifies that a class can contain both, you can search for an object using either of the unit systems you define, and the search mechanism finds a match, regardless of the unit in which the object is stored. For example, if you search for a bolt with a width of 5/8th inches, the classification search mechanism finds a bolt that is stored with a width of 1.6 centimeters.

The unit management system in Classification is defined in a classification class called the **Unit Definition** class containing an ICO for each measurement unit. The unit management system manages the:

- Type of measure (for example, length, area, volume)
- Unit name
- Unit display name
- System of measurement
- Default (base) unit
- Conversion factors

The following table describes how the measurement system stores values for length, in both metric and nonmetric units. Each measurement category has a base unit that is used for the conversion both within factors of that unit (for example, centimeters to millimeters or inches to feet) as well as for converting from metric to nonmetric (for example, inches to millimeters).

Object ID	Unit name	Unit display name	System of measurement	Conversion equation (multiplication factor)
Length_mm	**Millimeter**	**mm**	Metric	1
Length_in	**Inch**	**in**	Nonmetric	25.4
Length_cm	**Centimeter**	**cm**	Metric	10
Length_mil	**MilliInch**	**mil**	Nonmetric	0.0254
Length_ft	**Feet**	**ft**	Nonmetric	304.8

In the example, the base unit is millimeter. Any conversion that takes place is always in relation to the metric base unit. For example, to convert from feet to inches, Teamcenter first converts from feet to millimeters, and then from millimeters to inches.

Use the **Conversion multiplication factor E notation** attribute to specify very large or very small values in scientific notation. If a value is entered in this attribute, the conversion factor is ignored.

You can set the default active system of measurement with the **ICS_unit_active** preference. Additionally, you can set the default system of measurement when creating classes using the **ICS_unit_default** preference.

Teamcenter displays the unit of an attribute value as a hyperlink beside the value in the Classification application. You can change this unit for viewing or searching purposes. If a unit is not hyperlinked, Teamcenter cannot find the unit in the **Unit Definition** class. If the attribute is not yet used in any classes, the classification administrator can change the unit in the attribute dictionary or add the new unit definition to the **Unit Definition** class.

When working with legacy data, Classification searches for the object ID of the unit or, if it does not find that, the display name. If it finds a match, it displays the hyperlinked unit label and the new functionality is available to you. If Teamcenter does not find a match, it displays the legacy unit label.

> Tip:
>
> Teamcenter displays the storage unit in bold type in the list of available units.

If you work with NX, you can map Classification units to NX units using the **populate_nx_unit_definitions** argument of the **smlutility**. Once this is complete, the NX counterpart unit is displayed in the NX unit ID box. The respective units are then displayed in both NX and Teamcenter.

Create or modify a unit definition

By default, the **Unit Definition** class contains a wide selection of the most common metric and nonmetric units of measurement.

1. In the Classification Admin application, move the **Unit Definition** class from the **SAM Classification Root** node to the **Classification Root** node using drag-and-drop.

2. In the Classification application, refresh the classification hierarchy.

3. Select the **Unit Definition** class.

4. Add a new ICO for each measurement unit that you want to define or edit an existing measurement ICO.

5. In the Classification Admin application, select and refresh the **Unit Definition** class.

   The new or modified unit appears in the unit list when you create a new attribute in the **Dictionary** pane.

6. (Optional) Move the **Unit Definition** class back to the **SAM Classification Root** node to protect it from inadvertent modification.

Warning:

Do not change a unit definition (for example, the conversion factor or base unit) after you use it in an attribute. Changing the unit definition corrupts your data.

Protecting unit definitions from modification

The class hierarchy contains unit definitions. These definitions are stored as regular classification objects in a dedicated classification class. To maintain the unit definitions in the classification application, you must move the **Unit Definition** class from the **SAM Classification Root** node to the **Classification Root** (ICM) node. However, even if the administrator does not move the class, it is possible that Classification users can see the unit definition classification objects if they perform an object ID search (for example, using ***). To prevent any unintentional modifications to the unit objects, it is advisable to set up write protection for regular Classification users using access control on the **Unit Definition** class. Only write protection should be restricted. All users must have read access to the unit definitions.

Set a measurement system for a class

1. Create a class.

2. Add attributes. For each numerical attribute that you add, you must have defined both a metric and nonmetric format in the **Dictionary** pane.

3. In the **Class Details** pane on the **Hierarchy** tab, click:

   - **metric** to specify that the class can contain only metric values for ICOs.
   - **nonmetric** to specify that the class can contain only nonmetric values for ICOs.
   - **both** to specify that the class can contain both metric and nonmetric attribute values for ICOs.

   By selecting both types of units, you allow the search mechanism to find matches that have been stored in either of the systems of measure.

   Note:

   To change an existing class (containing data) from a metric or nonmetric class to one containing both systems of measure, you must first run the **smlutility** utility using the **-migrate** option.

If a class is set to **both** (metric and nonmetric), and if any of the attributes do not have a unit definition for both metric and nonmetric, Teamcenter uses the available unit for both unit systems. This behavior supports use cases where certain attributes may not have different metric and nonmetric units, such as time.

Mapping classification attributes to NX part attributes

You can map NX part file attributes to classification attributes. Classification attribute synchronization supports all types of classification attributes, including integer, real, double, string, date, variable length array (VLA), and key LOV.

Add images to classes

Images can be associated with the definitions of hierarchy objects to help identify the contents of a class. Images should be generic to adequately represent the variations in the contents of the group or class.

1. In Classification Admin, select the class and click **Edit current instance**.

2. Right-click the node in the hierarchy tree and choose **Add Image**.

   Teamcenter displays the **Open** dialog box.

3. Locate and choose an image file to be associated with the group or class and click **Open**.

   Teamcenter displays the image in the image viewer.

Mapping a class to another class

You can map ICOs from one classification class to another based on a definition specified in a mapping view. You can map one source class to one or more target classes.

Display the autofilter

You can activate a filter in the Classification application that allows the user to see a preview of those attribute values available in the currently selected class. This is especially useful when you are searching as you can select an existing value and search for it.

1. Set **ICS_show_autofilter** to **true**.

2. (Optional) Turn the autofilter off for a particular attribute:

   a. Select the class containing the attribute.

   b. Click the **Edit current instance** button.

   c. Open the **Class Attributes** pane.

   d. In the **Class Attributes** list, select the attribute for which you want to disable the autofilter.

   e. Click **Disable Auto Filter** in the **Properties** section.

f. Click **Save** .

Creating and managing views

Working with Classification views

Views enable you to define how a class is displayed in the Classification application by setting attribute protections and hiding class attributes on a user, group, role, or project basis. View objects are created and managed in the Classification Admin application. The correct view is automatically set for a user in the Classification application (and cannot be dynamically changed). You can create, remove, and modify views without any impact to the Classification data.

Note:

- Attribute properties that are assigned to attributes at the class level are inherited by the view. While some inherited properties can be overwritten, they cannot be removed. Additional properties can be added to the attribute at the view level.

  Attributes that are mandatory in a class are also mandatory in all views of the class.

- If a view is not explicitly defined for a class, Teamcenter automatically generates a default view, **default_view**.

  Unlike explicitly created default views, auto-generated default views are not visible in the Classification Admin hierarchy tree.

- When importing ICOs, Teamcenter considers the views belonging to the user who is importing. It looks at the views that take precedence for that user. Teamcenter then imports only attribute values for attributes existing in those views. If you define a custom view or a default view that does not contain all the attributes in the parent class, some ICO values may not be imported.

- Access control to specific groups, classes, and ICOs is performed at the group or class level. You cannot do this at the view level.

  If you want to use the graphics builder to generate graphics for ICOs, do this at the class level.

You can set view precedence for displaying views in Classification using the **ICS_view_selection_order** preference.

By default, the system uses the following view precedence when displaying classes in Classification:

- **User views**

  Define the way a class is displayed to a specific user. By default, user views take highest precedence over role, project, group, and default views when the class is displayed.

- **Role views**

Define the way a class is displayed to users with a specific role. By default, role views take precedence over project, group, and default views when the class is displayed.

- **Project views**

  Define the way a class is displayed to users within a specific project. By default, project views take precedence over group and default views when the class is displayed.

- **Group views**

  Define the way a class is displayed to a specific group of users. By default, group views take precedence over default views but are subordinate to user, role and project views.

- **Default views**

  Define the way a class is displayed to users for whom no user or group view is defined. Default views take lowest precedence when a class is displayed.

Create a view

1. From the hierarchy tree, select the storage or abstract class to be used as the basis of the new view. To determine whether a class is abstract or storage, right-click the symbol. The label on the shortcut menu indicates whether the class is a storage class. If not denoted as such, the class is considered abstract. The **Class Definition** pane at the right of the hierarchy tree displays the details and attributes of the selected class.

2. Click the **Add View** button, located at the bottom of the definition pane.

   The **Add View** dialog box displays the ID of the selected class in the banner. The view type and ID are set to **Default View**.

   > Note:
   >
   > You can also select the class, right-click, and choose **Add View**.

3. To create a type of view other than **Default View**, choose **User View**, **Role View**, **Project View**, or **Group View** from the **Type** list.

   > Note:
   >
   > The **Subclass** option in the **Type** list is intended only for use with legacy SML classes.

4. Enter an ID for the view, as follows:

   - **User View**

The ID must exactly match the user's Teamcenter user ID.

- **Role View**

  The ID must exactly match the Teamcenter role ID.

- **Project View**

  The ID must exactly match the Teamcenter project ID.

- **Group View**

  The ID must exactly match the Teamcenter group ID.

If necessary, you can locate user, role, project, and group IDs in the Teamcenter Organization application.

5. Click **OK**.

   The **View Definition** pane displays the view ID and the parent class of the view and its associated attributes.

6. Type a name for the view in the **Name** box.

   > Warning:
   >
   > The name box cannot contain blank spaces. Replace any spaces in the name with an underscore character.

7. (Optional) Type text in the **User 1** and **User 2** boxes. Any values entered in the **User 1** or **User 2** boxes are shown in the attribute's tool tip in the Classification application.

8. (Optional) In the **Multi-Site Collaboration** pane, choose the sites that you want to share the view definition.

9. Click the **View Attributes** tab and add attributes to the view.

10. Click **Save** on the toolbar.

Assign attributes to the view

1. Choose the attributes to be displayed in the view.

   a. Select the attribute in the **Class Attributes** list.

b. Click the right-arrow button to move the attribute to the **View Attributes** list. Repeat these steps to select additional attributes.

   Remove attributes from the **View Attributes** list by choosing the attribute or attributes and clicking the left-arrow button.

Teamcenter displays the attributes in the **View Attributes** list and displays a check mark next to the attribute in the **Class Attribute** list indicating that it is selected for this view.

2. (Optional) Use the up-arrow and down-arrow buttons to change the order in which the attributes are displayed on the Classification form.

3. (Optional) Click the **Preview** button to preview the Classification form.

> Tip:
>
> If you want to assign an attribute to multiple views, you can use the **smlutility** utility.

Apply properties to attributes

1. Select an attribute in the **View Attributes** list.

   Teamcenter displays the details of the selected attribute in the bottom section of the **View Definition** pane. The properties that were assigned to the attribute at the class level are displayed but cannot be modified. However, you can apply additional properties to the attribute at the view level.

2. Choose one of the following properties to apply to the selected attribute:

   - **Mandatory**

     Specifies that a value must be entered for the attribute in the Classification form. Mandatory fields are designated by a red triangle in the upper-right corner of the field. If an attribute is specified as mandatory in the parent class, you cannot overwrite this in the view.

   - **Protected**

     Specifies that a value is protected on the Classification form. Attributes designated as protected cannot be edited. You can, however, perform searches using protected attributes.

   - **Keep Width**

     Specifies that the width of the attribute's text box in the search pane should be maintained and not doubled, as is the default. This is useful when you are creating a more complex attribute layout.

- **Unique**

  Specifies that the value entered for the attribute must be unique. When displayed in the Classification form, unique fields are designated by a blue triangle in the lower-right corner of the field. If an attribute is specified as unique in the parent class, you cannot overwrite this in the view.

- **User Defined Button**

  Creates a user-defined button for the class that you can use to programmatically access information stored outside the currently selected class.

- **Visible**

  Attributes can only be defined as invisible if they are used in association with custom logic.

- **Auto Computed**

  Specifies that values for this attribute are calculated based on other attribute values using external custom logic.

- **Array**

  Specifies that an attribute is an array. Array attributes can be used to store multiple values in the database (called *multivalue fields*). Whether an attribute is an array is determined by the class attribute definition (not the view definition). However, specific array properties, such as length, horizontal, and vertical are defined within the class.

- **Length**

  Defines how many values are displayed for an attribute that is designated as an array.

  This field is enabled only when an attribute is designated as an array in the class definition.

- **Field Layout**

  Specifies the vertical layout for an attribute's name, annotation, dialog box, and unit.

- **Default Value**

  Defines a default value from the view. This value can be inherited from the class and can also be designated as fixed, so that it cannot be changed during ICO creation, modification, or copy.

- **Minimum Value and Maximum Value**

Defines minimum and maximum allowable values for an attribute value. This value can be inherited from the class or you can specify one specifically for the view. If you want to use a minimum and maximum values set in the class, select the **From Class** check box. Entering a value at the view level overrides any values set at the class level. If you check **From Class**, but no range is set at the class level, Teamcenter checks for a range set at the dictionary level. Use the following preferences to determine whether a Classification user must adhere to these ranges.

Set this preference to true	To
ICS_enforce_min_max_ constraints_on_update	Prevent Classification users from saving or updating classification instances or classified objects whose attribute values are beyond the specified range.
ICS_enforce_min_max_ constraints_on_copy	Prevent Classification users from creating a copy of an existing classification instance or classified object containing attribute values beyond the specified range. This includes revising or using the **Save As** command on an item or item revision outside the Classification application.
ICS_allow_min_max_ range_override	Allows the Classification administrator to specify maximum or minimum values in the class or view that are outside the range set in the attribute dictionary.

- **User Data**

  Specifies the location of the Java code required to support a user-defined button.

  Any information you add here is shown in the attribute's tool tip in Classification or Resource Manager.

Setting a default value for a view attribute

You can set a default value for a view attribute. Teamcenter displays this value in the attribute value's box when you create an ICO in that class. If you copy or reclassify ICOs, Teamcenter populates empty attribute values with the default value. If you specify that the default value is **Fixed**, you cannot change the value during any ICO operation (such as create, copy or reclassify).

If a default value is already defined for this attribute in the class, then the value is inherited to the view, and the **From Class** option is selected. If a class attribute is designated as **Fixed**, you cannot overwrite the class attribute definition with a new value or remove the default value for this particular view.

If you set a default value, this value is applied when using ITK functions or during import.

Customizing the layout of attributes and attribute fields

Customize the layout of attributes

You can specify the layout of the attributes in Classification or Resource Manager. You can arrange them vertically or horizontally, add separators or frames. You can also specify the layout of the individual attribute's name, annotation, dialog box, and unit. You can arrange these vertically, horizontally, or with the name above the dialog box. You can also dictate whether the width of the dialog box is maintained in search mode.

> Note:
>
> When modifying the layout of attributes in a view, you can associate the attribute layout to specific attributes. If you make extensive use of autocomputation and turn off visibility on attributes, this can create unwanted results, especially when you use frames and separators in the view layout. You must be careful when creating layout elements using attributes that can be hidden.

You can customize the layout of the attributes in Classification or Resource Manager in a view.

1. If none exists, create a view, or click to modify an existing view.
2. Add the necessary attributes to the view.
3. Select the layout tag that you require from the **Layout Tags** list.
4. Select the attribute below which you want to add the layout modification.
5. Click to move the layout tag to the **View Attributes** list.
6. If you have selected a layout tag that requires additional information (such as name of frame or number of columns), type the information in the box.
7. (Optional) Select **Preview** to see the new layout.
8. Click .

   Classification Admin checks that there is an equal number of start and end tags before saving.

Group attributes with a labeled frame

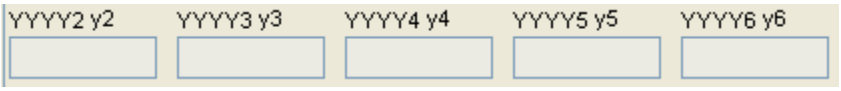
To achieve a layout that looks like this, perform the following steps:

1. Select the attribute from the **View Attributes** list below which the frame should begin.
2. Select **Start Frame** from the **Layout Tags** list.
3. Click ▸ to move the layout tag to the **View Attributes** list.

   The **Layout Tag Parameter** dialog box appears.
4. Type a name for the frame.
5. Select the attribute from the **View Attributes** list below which the frame should end.
6. Select **End Frame** from the **Layout Tags** list.
7. Click ▸ to move the layout tag to the **View Attributes** list.
8. Click **Preview** to see how the layout will look in Classification.

Arrange attributes horizontally

To achieve a layout that looks like this, perform the following steps:



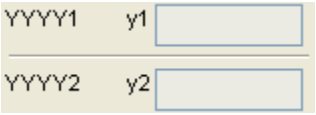
1. Select the attribute from the **View Attributes** list below which the horizontal layout should begin.
2. Select **Start Horizontal Layout** from the **Layout Tags** list.
3. Click ▸ to move the layout tag to the **View Attributes** list.
4. Select the attribute from the **View Attributes** list below which the horizontal layout should end.
5. Select **End Horizontal Layout** from the **Layout Tags** list.

6. Click ▸ to move the layout tag to the **View Attributes** list.

7. Click **Preview** to see how the layout will look in Classification.

Add separators

To achieve a layout that looks like this, perform the following steps:



1. Select the attribute from the **View Attributes** list below which you want the separator to appear.

2. Select **Separator** from the **Layout Tags** list.

3. Click ▸ to move the layout tag to the **View Attributes** list.

4. Click **Preview** to see how the layout will look in Classification.

Arrange attribute fields vertically

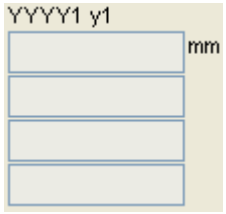
If you want to change the layout of individual attribute fields, use the **Field Layout** options. These are particularly useful when using multivalue fields (arrays):

1. Select the attribute for which you want to change the layout.

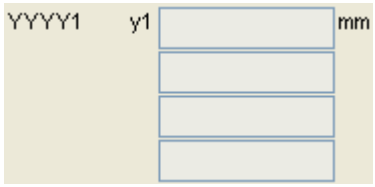
2. In the **Field Layout** list, select from one of the following options:

Horizontal — This arranges the attribute information vertically across the form. If the attribute uses multivalue fields, this appears as follows:

Vertical (Narrow) — This places the name above the boxes and the boxes are lined up beneath each other:

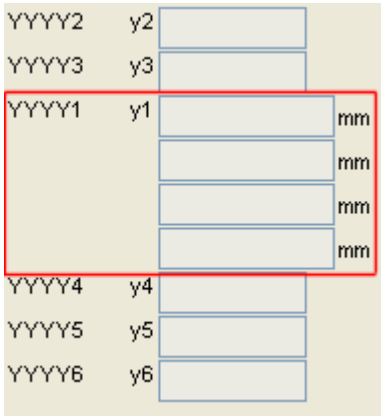


Vertical (Wide) This places the name beside the boxes and the boxes are lined up beneath each other:

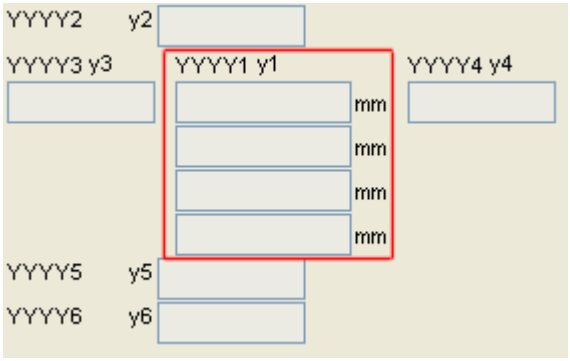


Default The default appearance is dependent on whether the attribute is placed within horizontal layout tags.

This attribute is placed within the normal vertical default layout.



This attribute is surrounded by horizontal layout tags.



3. (Optional) Select **Preview** to see the new layout.

4. Click .

Maintain the width of the text boxes in the search pane

Normally, the width of a text entry in the properties form is doubled in search mode. This provides more space to define the query, such as when you use ranges. If you prefer that the text entry box remains the same in search and show mode:

1. Select the attribute from the **View Attributes** list.

2. Click **Keep Width**.

Creating a user-defined button

The value of a user-defined button

You can create a user-defined button that you can use to programmatically access information stored outside the currently selected class and even outside Teamcenter. You can associate this button with your own Java class that is called when the button is clicked.

You can create a user-defined button in the class or in the view. If you define it in the class, it is automatically defined in the view. You can also create a user-defined button in the view alone.

1. Specify that an attribute has a user-defined button.

2. Define the Java class that you want to hook into the user-defined button in the properties file.

3. Implement the Java class for the user-defined button.

Add a user-defined button to an attribute

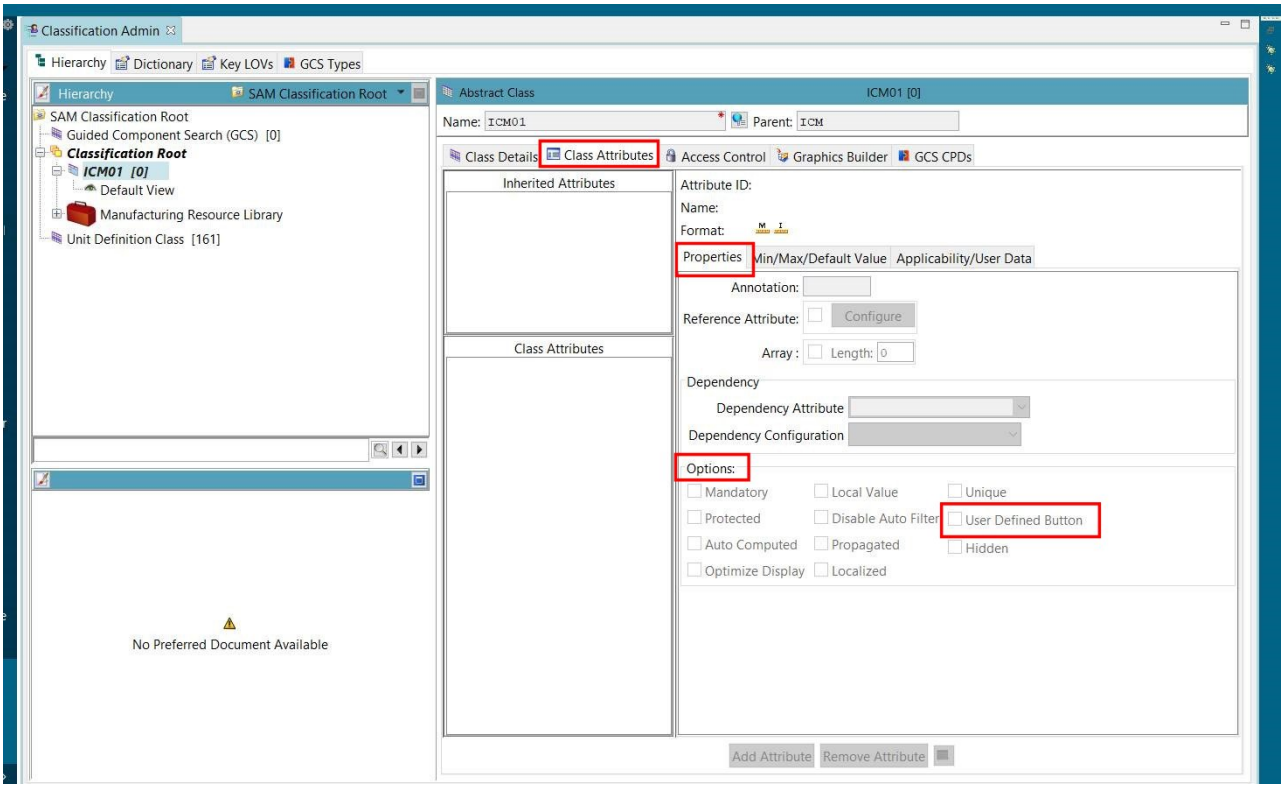
You can create a user-defined button for an attribute in a class or in its view. If you specify that an attribute has a user-defined button in a class, that attribute is automatically assigned a user-defined button in the view.

1. Select the class or the view containing the attribute to which you want to assign the user-defined button. If you specify that a class should have a user-defined button, you must specify that the view has one as well.

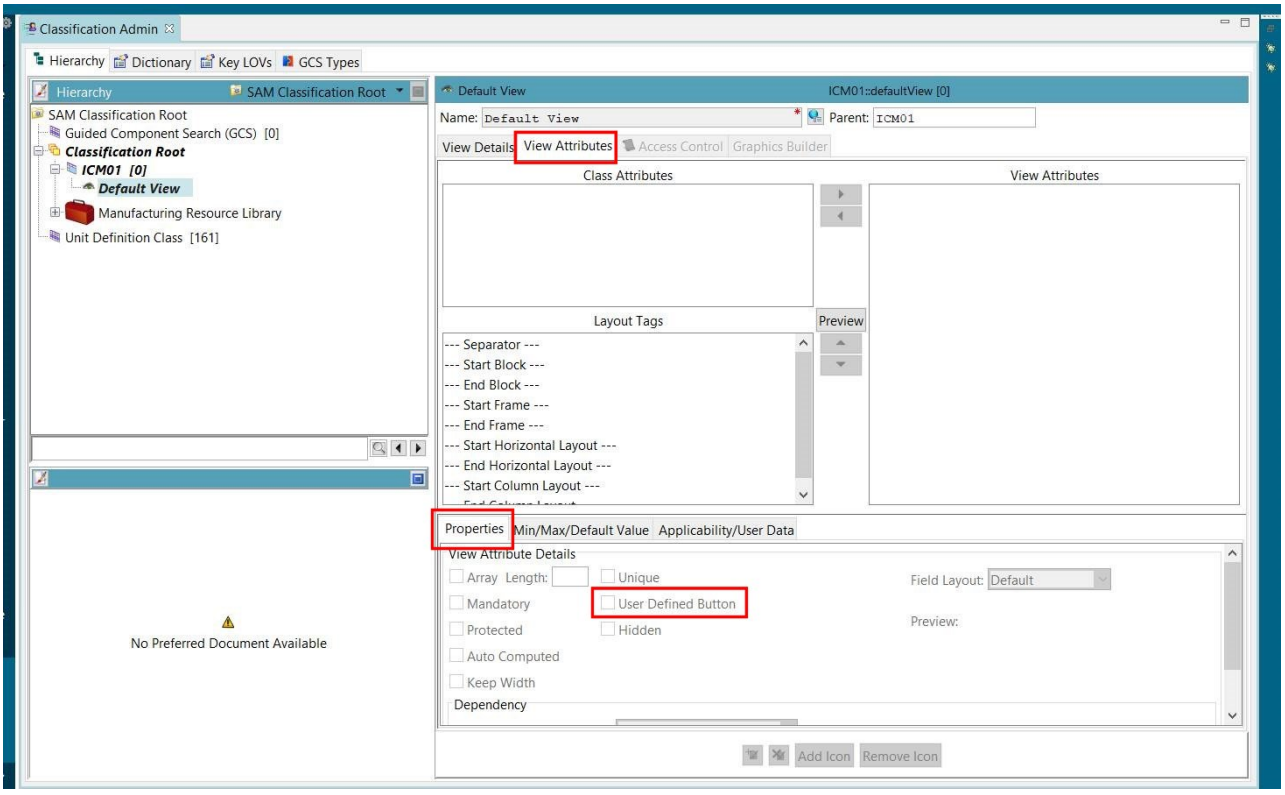
2. Click the **Edit** button .

3. In the **Class Attributes** or **View Attributes**pane, select the attribute.

4. For class attributes, check the **User Defined Button** option in the **Class Attributes** tab, in the **Properties** tab.



5. For view attributes, check the **User Defined Button** option in the **View Attributes** tab, in the **Properties** tab.



6. Save the class or view.

Defining the settings for a user-defined button in the properties file

You can add the user-defined button settings to the **common_user.properties** properties file (in the **com.teamcenter.rac.classification.common** package). This file is located in the **portal\plugins\com.teamcenter.rac.tcapps_***version_number***.jar** in your Teamcenter installation directory.

There are four different methods of defining the Java class to be used for an attribute:

- One Java class for all attributes in all classes:

  If you want all attributes to always have the same user-defined button Java code called **MyGenericHook** in all classes, you define:

  ```
  g4mform.userbutton=com.teamcenter.rac.classification.common.form.MyGenericHook
  ```

  This allows you to write one generic hook (or dispatcher) for all attributes and classes. Within your user code, you can evaluate from which attribute the code was triggered and perform the corresponding actions. If you return **null** in the method **getJComponent()**, no user-defined button is displayed for this attribute.

- One Java class for all attributes in one specific class:

  If, for example, you want all attributes to have the same user-defined button Java code called **MyShankEndCutterHook** in the **Shank End Cutter** class (Class ID **ugc0103**), you define:

  ```
  g4mform.userbutton.ugc0103=com.teamcenter.rac.classification.common.form.
       MyShankEndCutterHook
  ```

- One Java class for one specific attribute in all classes:

  If, for example, you want the **Tool Material ID** attribute (ID **–2503**) to always have the same user-defined button Java code called **MyToolMaterialHook** in all classes, you define:

  ```
  g4mform.userbutton.-2503=com.teamcenter.rac.classification.common.form.
       MyToolMaterialHook
  ```

- One Java class for one specific attribute in one specific class:

  If, for example, you want the **Shank Diameter** attribute (ID **–4110**) to have one user-defined button Java code called **MyToolMaterialHookMilling** for the **Shank End Cutter** class and a different hook called **MyToolMaterialHookDrilling** for the **Twist Drill** class (ID **ugc0301**), you define:

  ```
  g4mform.userbutton.-4110.ugc0103=com.teamcenter.rac.classification.common.form.
       MyShankDiameterHookMilling
  ```

```
g4mform.userbutton.-4110.ugc0301=com.teamcenter.rac.classification.common.form.
     MyShankDiameterHookDrilling
```

If an attribute is specified as having a user-defined button, but the system cannot find a corresponding class name on the four options described previously, the system uses the **G4MFormUserDefaultButton** class (in the **com.teamcenter.rac.classification.common.form** package).

The fourth method has the highest precedence, then the third, second, and first.

Siemens Digital Industries Software recommends specifying the icon used for the user-defined button in the properties file, as well.

> Note:
>
> Do not forget to specify package and class name.

Defining the settings for a user-defined button in the attributes user fields

You can define the user-defined button settings in the attribute's user fields. User-defined button definitions in user fields have higher precedence than definitions in the properties file. If you want to use this option, you must set the **ICS_user_defined_button_class** site preference to **User1** or **User2**. If the preference is set to **User1**, the system tries to find the user-defined button class code in the attribute's **User 1** field. If the preference is set to **User2**, the system searches for the code in the **User 2** field. This mechanism has several advantages:

- The information is stored in the database and no property files (residing on the client) have to be changed.
- The effects are visible immediately, without the need to copy files on each client machine (if the user-defined Java classes are already available).
- The user-defined button classes can be inherited from a class to its child classes. (When using the properties file approach, all relevant classes have to be specified in the properties file. No inheritance is possible there.)

There are three different methods of defining the Java class in the user field that should be used for an attribute:

- One Java class for one specific attribute in all classes:

  If, for example, you want the **Tool Material ID** attribute (ID **–2503**) to always have the same user-defined button Java code called **MyToolMaterialHook** in all classes:

  1. Start Classification Admin.
  2. Click the **Dictionary** tab.

3. Search for your desired attribute.

4. Click **Edit**.

5. Define the following in the **User 1** or **User 2** field (based on your **ICS_user_defined_button_class** setting):

   ```
   com.teamcenter.rac.classification.common.form.MyToolMaterialHook
   ```

- One Java class for one specific attribute in one specific class:

  If, for example, you want the attribute **Shank Diameter** (ID **–4110**) to have one user-defined button Java code called **MyToolMaterialHookMilling** for the class **Shank End Cutter**:

  1. Start Classification Admin.

  2. Click the **Hierarchy** tab.

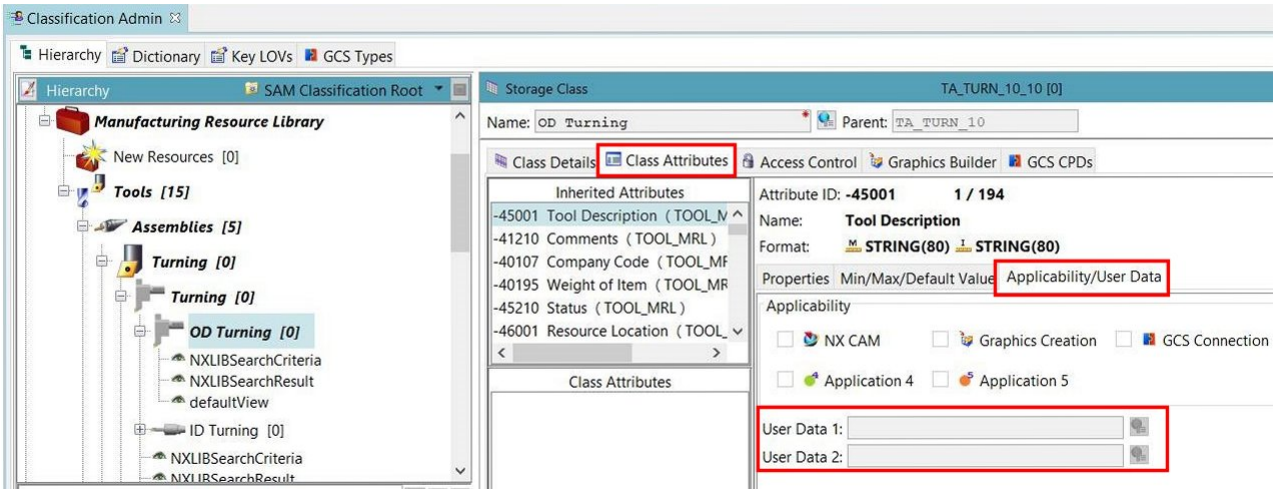
  3. Select the class where the attribute is defined.

  4. Click **Edit**.

  5. Select the desired attribute.

  6. Define the following in the **User 1** or **User 2** field (based on your **ICS_user_defined_button_class** setting):

     ```
     com.teamcenter.rac.classification.common.form.MyShankDiameterHookMi
     lling
     ```

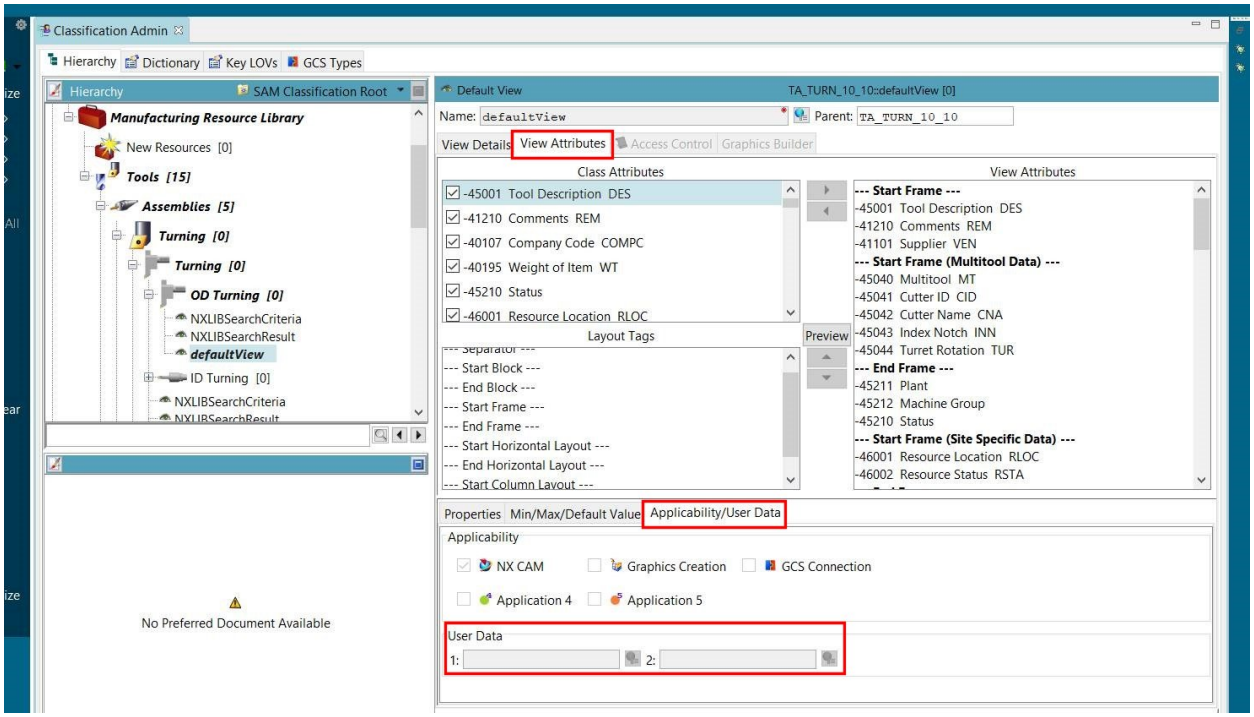


- One Java class for one specific attribute in one specific view:

If, for example, you want the attribute **Shank Diameter** (ID **–4110**) to have one user-defined button Java code called **MyToolMaterialHookMilling** for the **defaultView** view of the **Shank End Cutter** class:

1. Start Classification Admin.
2. Click the **Hierarchy** tab.
3. Select your desired view.
4. Click **Edit**.
5. Select the desired attribute.
6. Define the following in the **User 1** or **User 2** field (based on your **ICS_user_defined_button_class** setting):

```
com.teamcenter.rac.classification.common.form.MyShankDiameterHookMilling
```



If an attribute is defined to have a user-defined button, but the system cannot find a corresponding class name on the three options described previously, it tries to find the class in the properties file.

The third method has the highest precedence, and then the second, then the first.

Note:

Remember to specify package and class name.

Developing the Java code for a user-defined button

All user Java classes for the user-defined button hooks have to implement the **InterfaceG4MFormUserButton** interface (in the **com.teamcenter.rac.classification.common.form** package) and they have to extend the **JComponent** class (or a subclass of **JComponent**).

The following is the code for **InterfaceG4MFormUserButton**:

```
package com.teamcenter.rac.classification.common.form;
import javax.swing.JComponent;
public interface InterfaceG4MFormUserButton
{
    /**
     * @return the Java component that will be added to the Form layout
     */
    public JComponent getJComponent();
    /**
     * @return the AbstractG4MFormElement the user button is assigned to
     */
    public AbstractG4MFormElement getFormElement();
    /**
     * This method allows handling mode changes properly for the User-Defined Button.
     * Every time the mode changes, this method is called.
     * You can change the visibility or sensitivity of your JComponent based on the
mode here.
     * Or you could change the icon or do additional steps, as well.
     *
     * @param theMode the current mode of the G4MForm
     * The possible values for "theMode" are:
     *     G4MConstants.MODE_SEARCH
     *     G4MConstants.MODE_SHOW
     *     G4MConstants.MODE_EDIT
     *     G4MConstants.MODE_NEW
     */
    public void setMode( int theMode );
}
```

The user-defined Java classes you write must have the following constructor:

```
public MyGenericHook( AbstractG4MFormElement theFormElement, int theIndex )
```

theFormElement is the form element for the attribute on which the user-defined button should be displayed. With this object, you can also retrieve all values from the form and you are able to set other values in the form. The mode, class ID, and class name are displayed. In addition, you can access

the **ClassificationService** that allows searching the complete Classification database and manipulating values in the database.

theIndex is the index of the multivalue field. Multivalue fields have one user-defined button for each field. To allow evaluating to which field the user-defined button belongs, this input parameter is passed in. If the attribute is a single value field, **1** is passed in as **theIndex**. The counting of the multivalue fields starts with 1.

For the implementation of simple user-defined buttons, Siemens Digital Industries Software recommends that you subclass the **G4MFormUserDefaultButton** class. Then you can overwrite the **actionPerformed()** method and put your implementation into this method.

Examples of developing Java code for a user-defined button

Example 1: Blue background

The following example shows a user-defined button Java class that displays a JButton with an icon on it that changes the background color of the form to blue when it is clicked:

```
package com.teamcenter.rac.classification.common.form;
import …;
public class MyColorUserButton extends JButton implements InterfaceG4MFormUserButton,
ActionListener
{
    private AbstractG4MFormElement m_formElement;
    private int                    m_multiFieldIndex;
    //============================================================================
    public MyColorUserButton( AbstractG4MFormElement theFormElement, int theIndex )
    {
        m_formElement = theFormElement;
        m_multiFieldIndex = theIndex;
        configure();
        addActionListener(this);
    }
    //============================================================================
    private void configure()
    {
        Icon icon =
m_formElement.getForm().getRegistry().getImageIcon( getClass().getName() +".ICON" );
        if( icon == null)
        {
            icon =
m_formElement.getForm().getRegistry().getImageIcon( "g4mform.userbutton.DEFAULT_ICON" );
        }
        if( icon != null)
        {
            setIcon(icon);
        }
        setMargin( new Insets(0,0,0,0) );
        setFocusPainted( false );
    }
    //============================================================================
    public void actionPerformed(ActionEvent ev)
```

```
    {
        m_formElement.getForm().setBackground(Color.BLUE);
    }
    //==========================================================================
    public AbstractG4MFormElement getFormElement()
    {
        return m_formElement;
    }
    public JComponent getJComponent()
    {
        return this;
    }
    public void setMode(int theMode)
    {
        switch( theMode )
        {
        case G4MConstants.MODE_SEARCH:
            setVisible(true);
            break;
        case G4MConstants.MODE_SHOW:
            setVisible(true);
            break;
        case G4MConstants.MODE_NEW:
            setVisible(true);
            break;
        case G4MConstants.MODE_EDIT:
            setVisible(true);
            break;
        }
    }
} // End of class MyColorUserButton
```

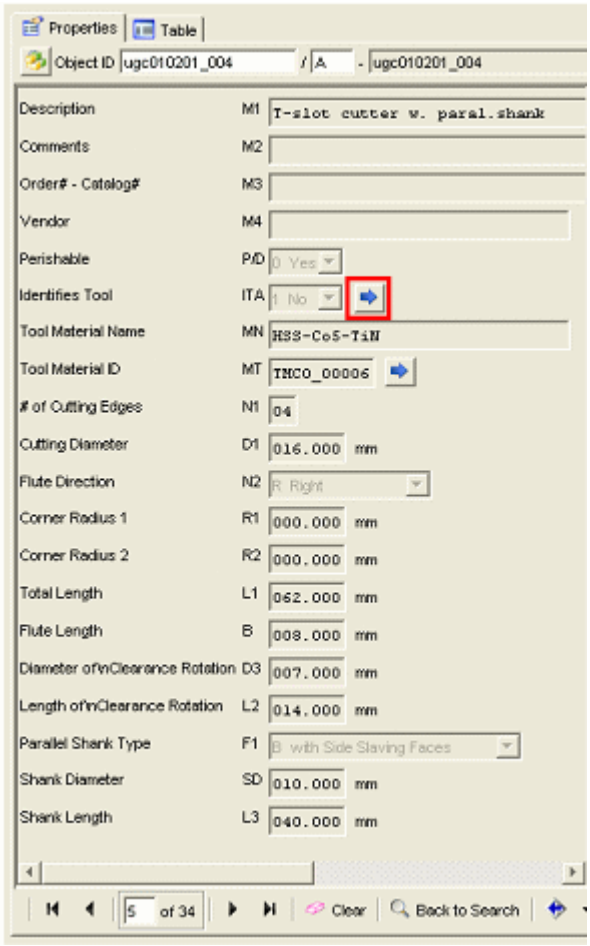
Example 2: Average value

In this example, when you click the displayed button, the average value of all numeric attribute values is calculated and displayed in a message box. This value is written into the first attribute of the form. The following is the code of the **actionPerformed()** method. The remaining code remains the same as in Example 1.

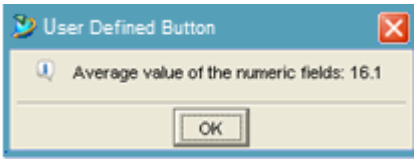
```
package com.teamcenter.rac.classification.common.form;
import …;
public class MyAverageValueUserButton extends G4MFormUserDefaultButton
{
    public MyAverageValueUserButton ( AbstractG4MFormElement theFormElement, int
theIndex )
    {
        super (theFormElement, theIndex);
    }
    //==========================================================================
    public void actionPerformed(ActionEvent ev)
    {
        try
        {
           ICSProperty[]            icsProps     =
m_formElement.getForm().getProperties();
           AbstractG4MFormElement[] formElements =
m_formElement.getForm().getFormElements();
```

```
            // Loop through all properties
            float average      = 0;
            int   numericCount = 0;
            for (int i = 0; i < icsProps.length; i++)
            {
               String value = icsProps[i].getValue();
               if (!value.equals(""))
               {
                   if (formElements[i].getPropertyDescription().getFormat().isInteger())
                   {
                       // If the attribute format is integer -> add the value to average
                       average += Integer.parseInt(value);
                       numericCount++;
                   }
                   else if (formElements[i].getPropertyDescription().getFormat().isReal())
                   {
                       // If the attribute format is float -> add the value to average
                       average += Float.parseFloat(value);
                       numericCount++;
                   }
               }
            }

            // Calculate the average value
            if (numericCount > 0) average /= numericCount;
            // Display the average value in a message box
            MessageBox.post("Average value of the numeric fields: " + average,
                            "User-Defined Button", MessageBox.INFORMATION );

            // Here you could do for example a lookup in an Excel Sheet, as well.
            // Or you could retrieve values from ICOs in other classes and write those
values in the form.
            // Write the calculated value in the first attribute
            ICSProperty prop = new ICSProperty(formElements[0].getProperty().getId(),
Float.toString(average));
            formElements[0].setProperty(prop);
          }
          catch( Exception ex )
          {
          }
      }
}
```

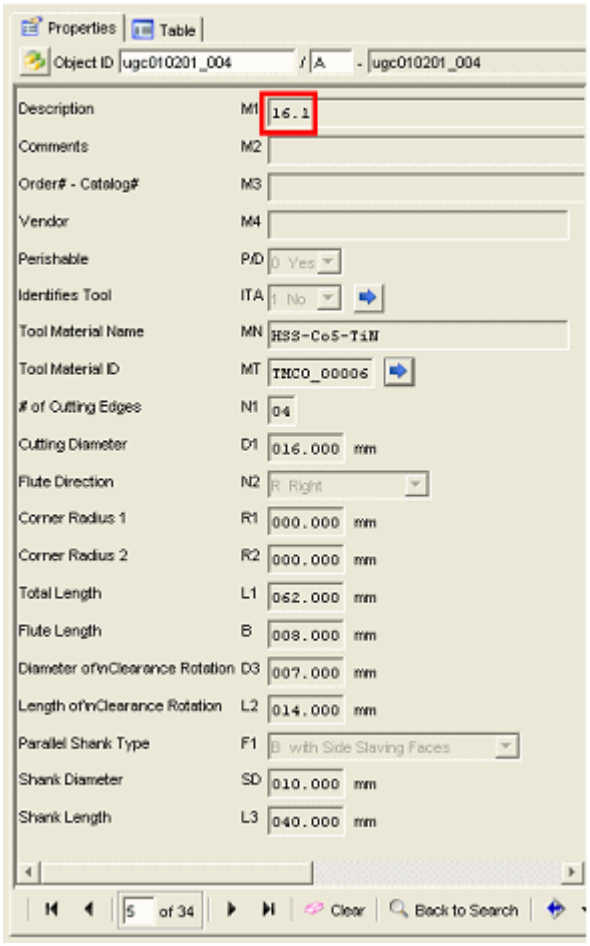
1. Click **User Defined Button**:



2. Teamcenter displays a message box:



3. Teamcenter writes the value into the first attribute:



Example 3: Display attribute and class information

In this example, the IDs, values, formats, and key-LOV definitions of all form attributes are printed to the debug output when you click the user-defined button. A message box appears with the current class ID and the ID name, format, and value of the attribute belonging to the clicked user-defined button are shown. The following is the code of the **actionPerformed()** method. The other framework is still the same.

```
public void actionPerformed(ActionEvent ev)
 {
    try
    {
      ICSProperty[]            icsProps     = m_formElement.getForm().getProperties();
      AbstractG4MFormElement[] formElements = m_formElement.getForm().getFormElements();
      // Loop through all properties
      for (int i = 0; i < icsProps.length; i++)
      {
          // Debug Output: IDs and Values of all form attributes
          Debug.print  ("ID: "+icsProps[i].getId()+"   Value: "+icsProps[i].getValue());
          // Debug Output: Formats of all form attributes
          Debug.println("   Format:
"+formElements[i].getPropertyDescription().getFormat());

          if( formElements[i].getPropertyDescription().getFormat().isList() )
          {
              ICSKeyLov keyLov =
m_formElement.getPropertyDescription().getFormat().getKeyLov();
              String[] keys   = keyLov.getKeys();
              String[] values = keyLov.getValues();
              for (int j = 0; j < keys.length; j++)
              {
                  // Debug Output: Keys and Values for KeyLOVs
                  Debug.println("KeyLOV:   Key: "+keys[j]+"   Value: "+values[j]);
              }
          }
      }

      MessageBox.post("Class ID: "        +
m_formElement.getForm().getClassId()              + "\n" +
                      "Attribute ID: "    +
m_formElement.getPropertyDescription().getId()    + "\n" +
                      "Attribute Name: "  +
m_formElement.getPropertyDescription().getName()  + "\n" +
                      "Attribute Format: "+
m_formElement.getPropertyDescription().getFormat()+ "\n" +
                      "Attribute Value: " + m_formElement.getProperty().getValue(),
                      "User-Defined Button", MessageBox.INFORMATION );
   }
   catch( Exception ex )
   {
   }
}
```

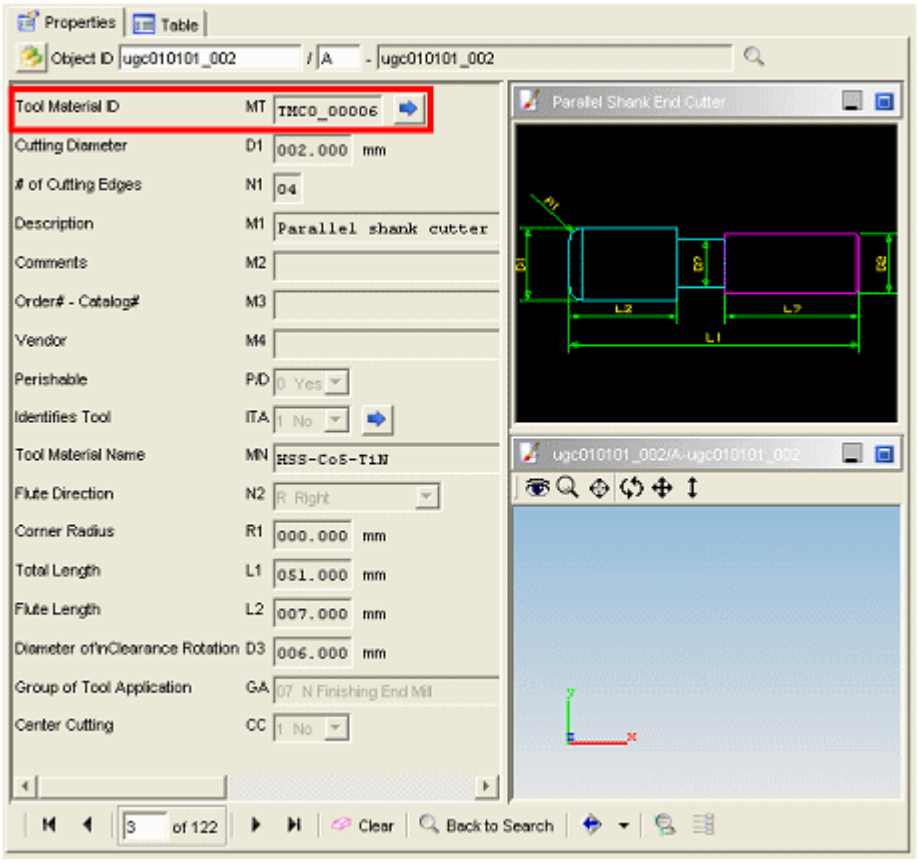
Example 4: G4MFormUserICORefButton

This example displays a dialog box with the form of a defined class. In this dialog box, you can display additional information for this attribute value, or you can search for a specific instance and insert the instance ID in the entry field. The behavior of the dialog is different based on the current mode. This functionality allows you to reference an ICO in another class. This example can be found in the **Sample\IN-CLASS** directory of your installation and can easily be modified to be used for different attributes.

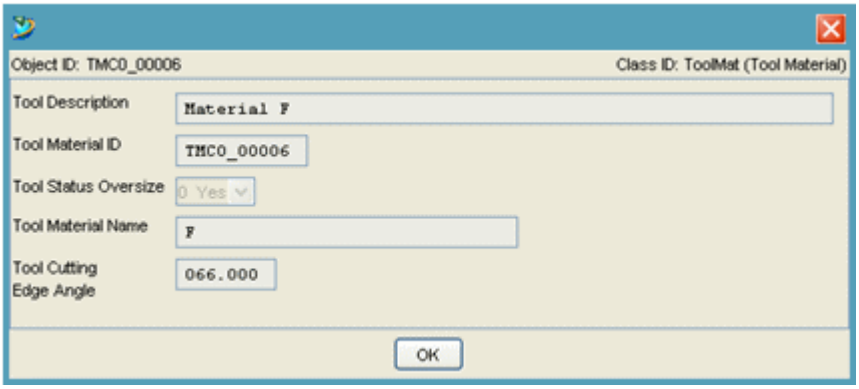
- **Show** mode:

If the form is in **Show** mode and the values of existing ICOs are displayed, you can click the user-defined button (in this example, for the **Tool Material ID** attribute – Step 1). The dialog then displays additional information about the tool material that is specified in the entry field (Step 2). **TMC0_00006** is the ID of an ICO that is stored in the **Tool Material** class. The ICOs in this class store additional information about the tool material.

1. Click **User Defined Button** for **Tool Material ID**.



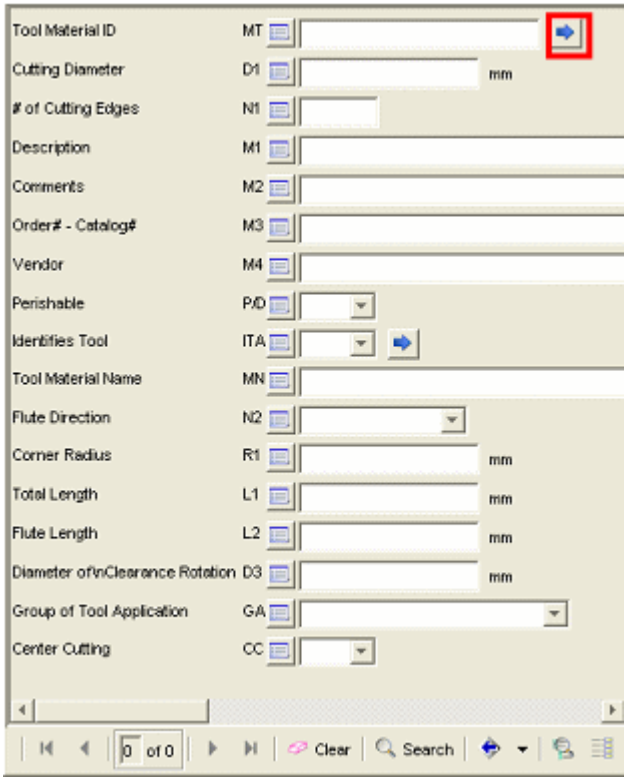
2. Additional information for the current tool material is displayed.



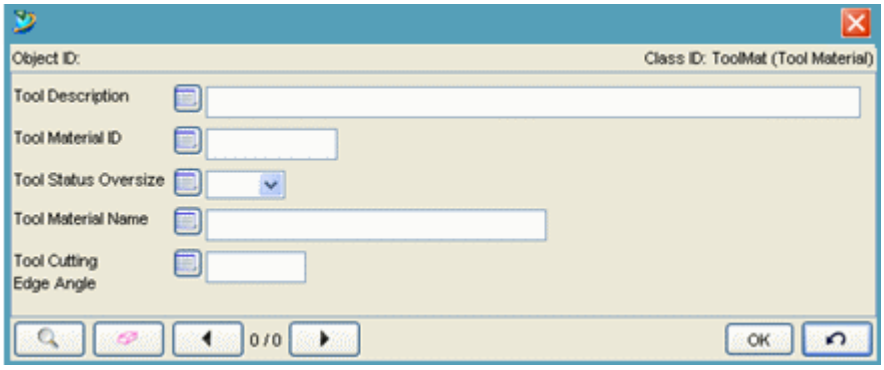
- **New**, **Edit**, or **Search** mode:

  If the form is in **New**, **Edit**, or **Search** mode, the entry field is typically empty. When you click the user-defined button for the attribute **Tool Material ID** (step 1), a dialog box with an empty **Tool Material** class form is displayed (step 2). In this dialog box, you can specify your desired search query and search for ICOs in the additional **Tool Material** class (step 3). After clicking the **Search** button, the corresponding results are displayed (step 4). With the arrow buttons, you can now browse through the matches. If you want, you can click the **Clear** button, define a new search query, and execute a subsequent search. Once you have found the desired instance, click **OK** and the ID of this instance is written into the **Tool Material ID** entry field (step 5).

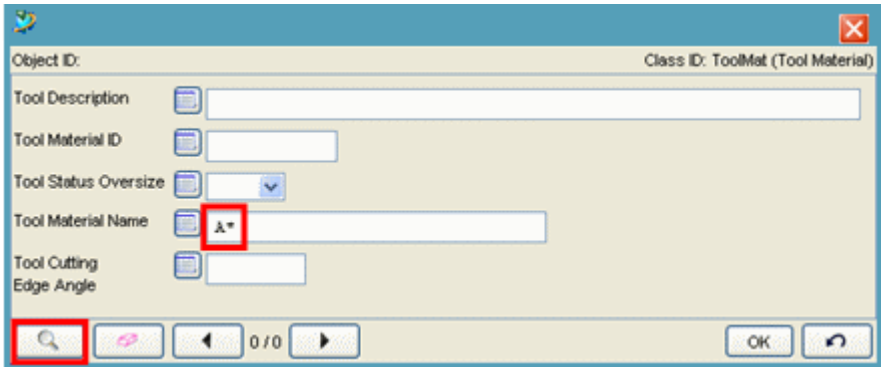
  1. Click **User Defined Button** for **Tool Material ID**.



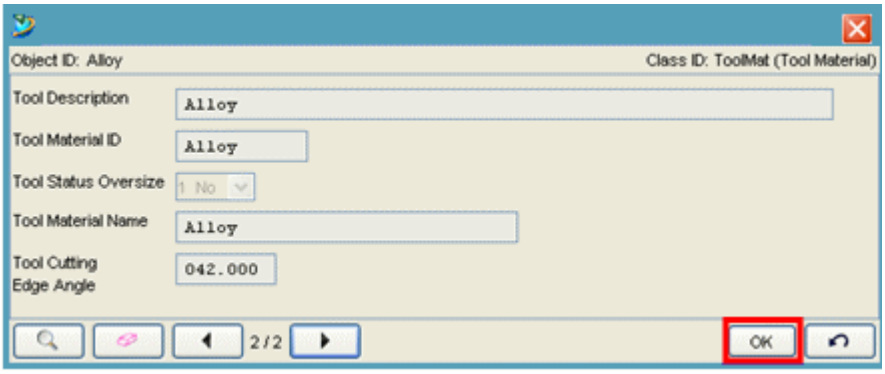
  2. Teamcenter displays a dialog box with an empty **Tool Material** class form.



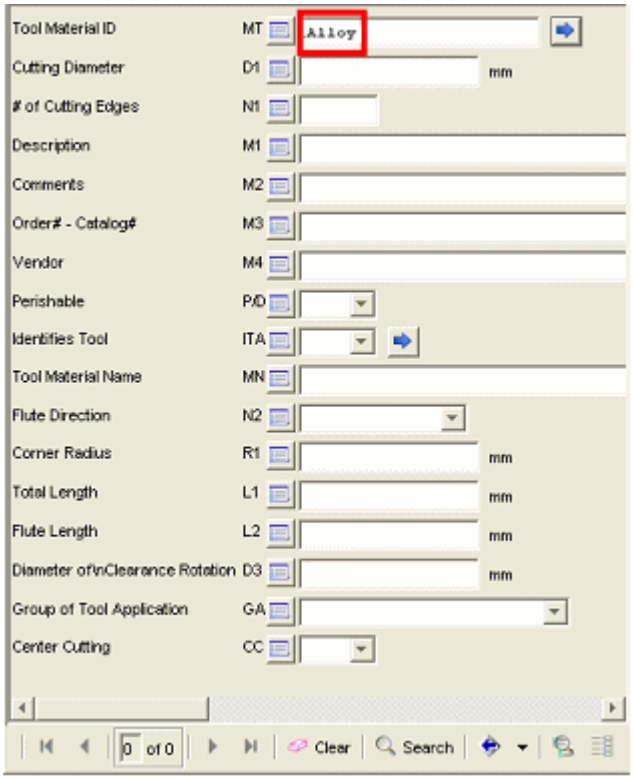
3. Specify a search query and click **Search**.



4. Click **OK** in the dialog box that displays the query matches.



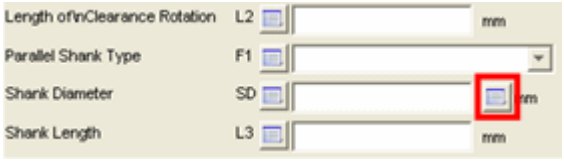
5. Teamcenter writes the object ID of the desired tool material into the entry field of the original class.



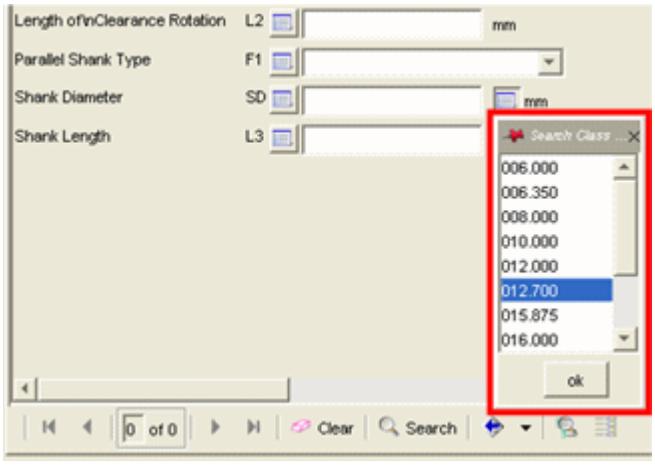
Example 5: G4MFormUserAutoFilterButton

This example is used with the **Shank Diameter** attribute. When you click the user-defined button on the right side of the entry field (step 1), a popup dialog with all existing values for this attribute in the ICOs of this class is displayed (step 2). When you select a value and you click **OK**, this value is written into the attribute entry field and the dialog disappears (step 3). If there are already other attribute values specified in the form, only the existing values that match the other given constraints are displayed. You can find this example in the **Sample\IN-CLASS** directory of your installation and modify it for different attributes.

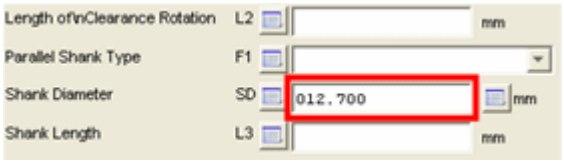
1. Click **User Defined Button**.



2. Select value in the value list dialog and click **OK**.



3. Teamcenter writes the value into the dialog box.



Specify applicable attributes in a view

You can specify attributes in a class to be applicable for a certain application or use.

Additionally, you can specify attributes in a view to be applicable for a certain application. When you do this, you can only add new applications to the selection for each attribute. You cannot remove any applicability set in the class.

1. In the **View Attributes** list in the **View Attributes** pane, select the attribute for which you want to add applicability.

2. In the **Applicability** section, select additional applications for which the attribute is applicable.

Modify a view

Views do not affect the data stored in the classes with which they are associated. Therefore, you can modify the attributes and properties of views without compromising the integrity of your Classification data.

1. From the hierarchy tree, choose the view that you want to modify.

   The **View Definition** pane, located to the right of the hierarchy tree, displays the details and attributes of the view.

2. Click the **Edit** button on the toolbar to activate the pane.

3. (Optional) Modify the attributes, attribute properties, or Multi-Site Collaboration sharing.

4. Click **Save** on the toolbar.

> Note:
>
> You can click **Cancel** at any time *prior to saving* the modifications to clear the form and revert to the previous definition.

Copy a view

1. Right-click the view you want to copy and choose **Copy View** from the shortcut menu.

2. Right-click the class to which you want to copy the view and choose **Paste View** from the shortcut menu.

3. In the **Copy View** dialog box, select the view type from the **Type** list.

4. Type an ID for the view.

   If you copy a default view to another default view, you cannot change the view ID.

5. Clear the **Add attributes from class** option if you do not want to adopt the listed attributes from the target class in the copied view.

   Teamcenter removes any attributes that are contained in the source class but not in the target class from the copied view. These are displayed in the lower list in **Copy View** dialog box.

Delete a view

Views do not affect the data stored in the classes with which they are associated. Therefore, you can delete views without compromising the integrity of your Classification data.

1. From the hierarchy tree, choose the view that you want to delete.

   The **View Definition** pane, located to the right of the hierarchy tree, displays the details and attributes of the view.

2. Click the **Edit** button on the toolbar to activate the pane.

3. Click the **Delete** button on the toolbar.

Teamcenter displays the **Delete Confirmation** dialog box for you to confirm the delete action.

4. Click the **Yes** button to confirm the deletion. Teamcenter deletes the view from the hierarchy structure.

Define a mapping view

Define a mapping view to allow you to map the ICOs from one class to another class. You can map the source class to multiple target classes.

1. Select the desired target class from which the ICOs should be mapped.
2. Click **Add View**.
3. Select **Mapping View** from the list.
4. Type the ID of the source class. This name must be unique within the target class.
5. Click **OK**.

   The **View Attributes** pane displays the attributes found in both target and source classes, as well as additional internal attributes that are commonly needed for mapping.

6. Do one of the following:

   - Map automatically by clicking **Auto**.

     The application automatically maps attributes in the source class to attributes in the target class containing the same ID.

   - Select a source attribute and a target attribute and click **Map**.

     The application maps the source attribute to the target attribute.

   - Map to an absolute value.

     Enter the value in the **Mapping** column, for example, **10**. You must place text within single quotation marks, such as **'Released'**.

   - Map using a mathematical expression.

Enter the expression in the **Mapping** column. You must put a hash sign before the attribute number, for example, **#-2503**. You can use one attribute and a value, such as in the following example:

```
#-4120+0.5
```

The absolute value **0.5** is added to the value of class attribute **-4120** and the result stored in the mapped ICO attribute.

The following four mathematical operators can be used:

+, -, *, /

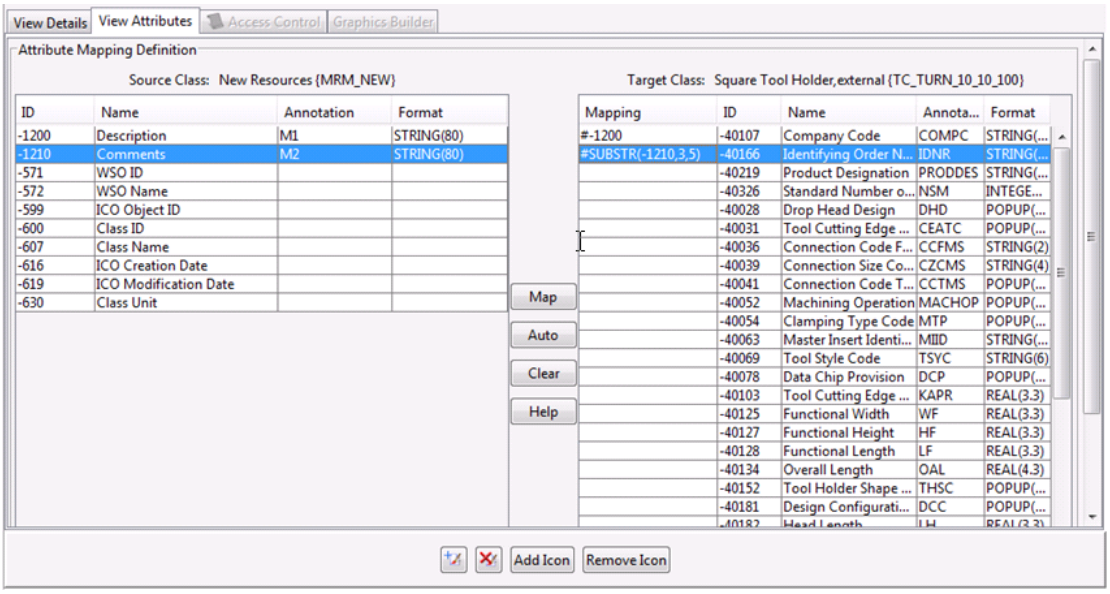
- Map using the **SUBSTR** operator.

> Note:
>
> You must ensure that the format and the number of decimal places of the mapped attribute values match. You cannot, for example, map a string to a float, or a **Real(2.5)** to a **Real(2.3)**.

7. Click **Save** on the toolbar.

Using the SUBSTR operator when mapping attributes

When mapping attributes during classification object (ICO) mapping, you can use a substring operator in the **Mapping** column that specifies that only specific characters in a string are mapped.



The entry must have the following format:

#SUBSTR(*attribute-ID***, *first-character-index*, *last-character-index*)**

Teamcenter looks for the specified attribute, and then copies the characters from *first-character-index* to *last-character-index* to the target attribute.

The table demonstrates which characters are mapped to the target attribute from a source attribute with an ID of **-1102** and a value of **AXR15739**.

This entry	Maps these characters
#SUBSTR(-1102,1,3)	**AXR**
#SUBSTR(-1102,4,*)	**15379** Note: If you use the wildcard symbol as the second index, Teamcenter copies all characters from the first index to the end of the string.
#SUBSTR(-1102,2,4)	**XR1**

Allowable types for mapping classes

You can map the following class attribute types from a source to a target class.

Source class	Target class				
	Integer	String	Date	Real	Key-LOV
Integer	✓	✓	X	✓	X
String	✓	✓	X	✓	✓
Date	X	X	✓	X	X
Real	X	✓	X	✓	X
Key-LOV	✓	✓	X	✓	✓

Note:

You can only use the **SUBSTRING** operator if the source attribute is of the **String** type.

Create mapping views for NX

You can limit and organize the number of attributes you show in the **Search Criteria** and **Search Results** dialog boxes in NX CAM by creating mapping views for classes.

1. Select the class whose attributes you want to limit.
2. In the **Class Details** pane, click **Add View**.
3. In the **Add View for Class** dialog box, select a view type.

View type	Description
NXLIB_SearchCriteria	Specifies which attributes are displayed in the **Search Criteria** dialog box in NX.
NXLIB_SearchResult	Specifies which attributes are displayed in the **Search Result** dialog box in NX.

4. Type a name for the view in the **View ID** box and click **OK**.
5. Click the **View Attributes** tab.

   The **View Attributes** pane displays all the attributes available in the selected class.
6. Select the attributes you want to display in the NX dialog boxes in the **Class Attributes** list and move them to the **View Attributes** list using the left and right arrows.
7. Arrange the attributes in the **View Attributes** list in the desired order by selecting them and clicking the up and down arrows.
8. Save the view.
9. Export the classification hierarchy using the **mrm_export_resources** utility.

Teamcenter lists the attributes that you specify in the **NXLIB_SearchCriteria** view as parameters of the **DIALOG** statement in the DEF file. It writes the attributes that you specify in the **NXLIB_SearchResult** view as parameters of the **RSET** statement. If you do not create an **NXLIB_SearchResult** view, Teamcenter automatically uses the same attributes for the **Search Result** dialog box as it does for the **Search Criteria** dialog box. If you do not specify either of these views, all Teamcenter attributes are displayed in both dialog boxes.

> Tip:
>
> Use the **mrm_export_resources** utility to export hierarchy images for use in the NX dialog boxes.

Creating and managing key-LOVs

Create a key-LOV

You use key-LOVs to define one or more values that can be set for specific classification attributes. Once created, these lists are stored in the data dictionary and can be used and reused as required.

The process of creating a key-LOV consists of creating the key-LOV in the database, adding and inserting entries and submenus, previewing the key-LOV form, and saving the key-LOV. You can also modify key-LOV entries or remove entries from the key-LOV.

The following constraints apply when creating a key-LOV.

Object	Length in characters
Key-LOV key	31
Key-LOV value	63
Key-LOV ID	10

Use caution when using key-LOVs with both systems of measure. When you convert from one system to another, Teamcenter does not convert the values.

1. Click the **Key-LOV** tab.

   Teamcenter displays the **Key-LOV** pane.

2. Click the **Create** button .

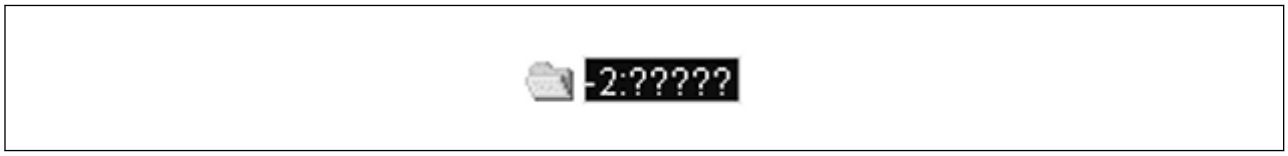
   Teamcenter displays the **New Key-LOV** dialog box. You use this dialog box to create and reserve an ID for the new key-LOV within the database.

   > Caution:
   >
   > The key-LOV ID must be a negative number. It cannot be longer than 31 characters.

3. Type a unique identifier for the key-LOV in the **Key-LOV ID** box and click **OK**.

   > Note:
   >
   > Once a key-LOV ID is assigned, it should not be modified.

   The system displays the new key-LOV as an open folder (root) in the **Detailed Key-LOV Definition** pane.



Open folder

The label of the root folder contains the key-LOV ID that you created along with a series of question marks. The question marks indicate that the name of the key-LOV is not defined.

The key-LOV ID and name are also displayed in the **Key/ID** and **Entry Value** boxes.

4. Type a name for the key-LOV in the **Entry Value** box. This is a mandatory entry.

> Caution:
>
> The **Entry Value** string cannot be longer than 63 characters.

5. Click anywhere in the **Detailed Key-LOV Definition** pane to update the key-LOV tree display.

6. (Optional) In the **Multi-Site Collaboration** pane, choose the sites you want to share this Key-LOV definition.

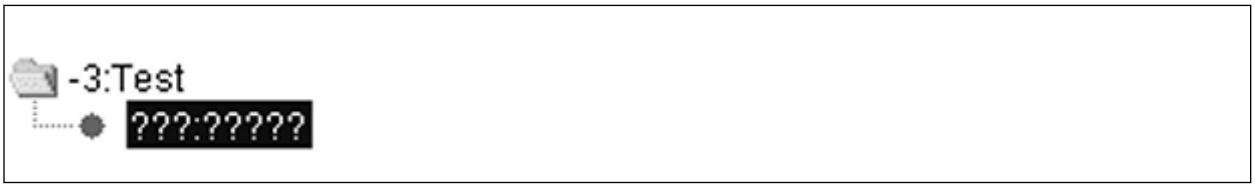
Add or insert key-LOV entries

> Note:
>
> You must be in create or edit mode to add or insert entries to a key-LOV.

1. Choose the key-LOV root or any node and click:

   - **Add Entry**

     Teamcenter displays the entry below the last entry in the selected folder or below the last entry in the folder of the selected entry in the key-LOV tree.



Key-LOV tree

   - **Insert Entry**

     The system displays the new entry in the tree above the selected node.

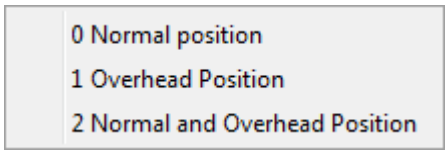
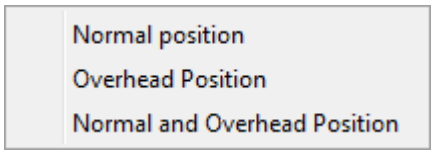
The label of the new entry is comprised entirely of question marks. In the next two steps, you define the key and description for the entry.

2. Click the text in the **Key/ID** box and type a key for the entry. This key cannot exceed 31 characters.

> Note:
>
> Entry keys must be unique within the context of a single key-LOV, however, they need not be unique within the database. For example, multiple key-LOVs can contain entries with a key of **02**, but a single key-LOV cannot have two entries with a key of **02**.

3. (Optional) Type a value for the entry in the **Entry Value** box. This entry cannot exceed 63 characters.

4. (Optional) Hide the key of the key-LOV by selecting **Hide Key**.

Key-LOV with key not hidden	Key-LOV with key hidden
	

5. Click anywhere in the pane to update the key-LOV tree display.

   The entry is now identified and displayed in the key-LOV tree.

Repeat these steps to add or insert additional entries to the key-LOV tree.

Add or insert a key-LOV submenu

You can add submenus to the root node of the key-LOV or to other submenus creating cascading key-LOVs. You cannot add submenus to key-LOV entries.

> Note:
>
> You must be in create or edit mode to add or insert submenus into a key-LOV.

1. Choose a node in the key-LOV tree. The new entry is inserted above the selected node.

2. Click:

   - **Add Submenu**

     Teamcenter adds a new submenu as a child of the current folder.

- **Insert Submenu**

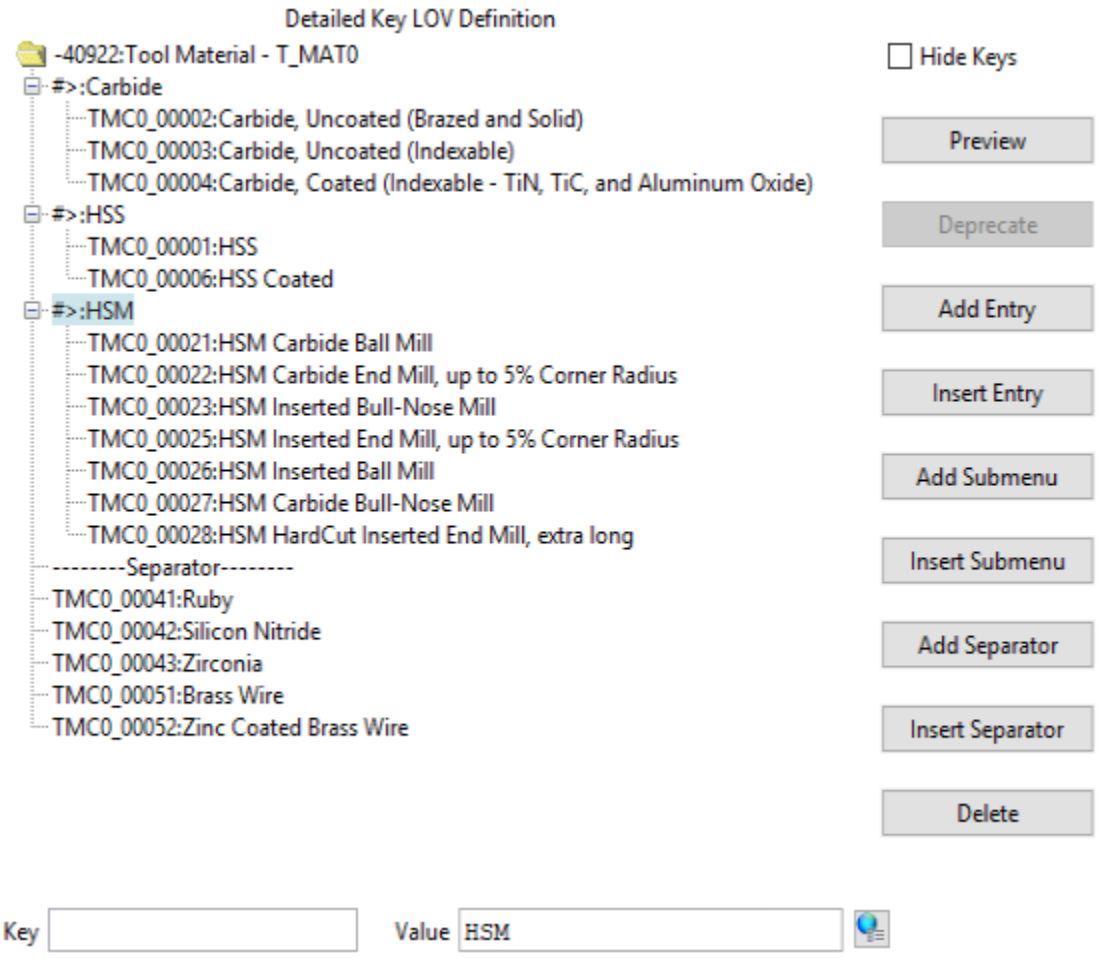
  The system inserts a new submenu in the tree at the same level as the currently selected folder, or at the same level as the folder containing the currently selected entry.

The label of the new entry is comprised entirely of question marks. In the next step, you define the description for the submenu.

3. Type a description for the submenu in the **Entry Text** box. This entry cannot exceed 63 characters.

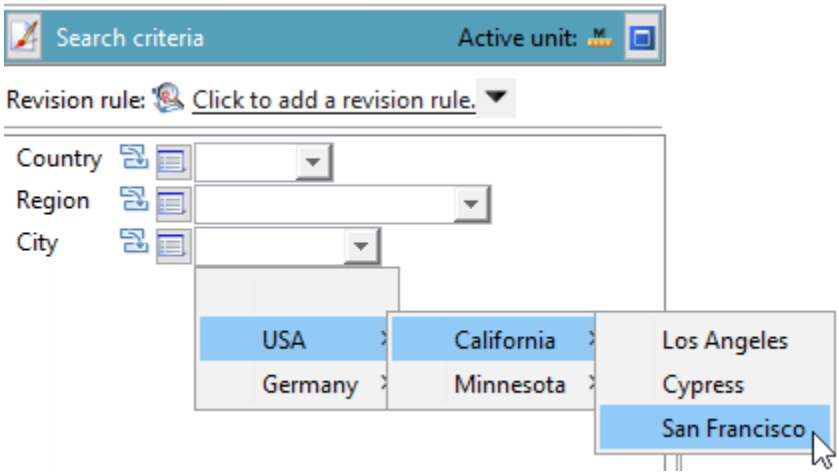
4. Click anywhere in the pane to update the key-LOV tree display. The submenu is identified and displayed in the key-LOV tree.

Repeat these steps to add or insert additional submenus in the key-LOV tree.



Create an interdependent key-LOV

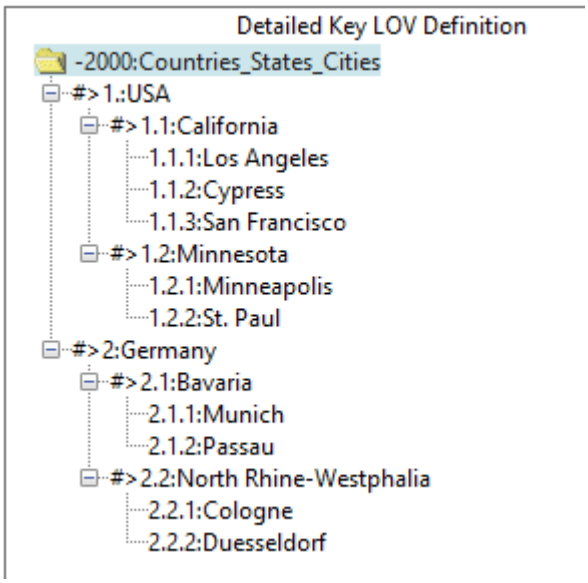
In an interdependent key-LOV, setting the value of one key-LOV attribute automatically sets the values of other key-LOV attributes.



If you select **USA** as the value for **Country**, the values of **Region** are limited to **California** and **Minnesota**. Alternatively, if you select **San Francisco** in this example, the values for **Region** and **Country** are automatically set.

You set up interdependency between attributes using one key-LOV that contains all the keys and values required. Then, you create attributes referencing each level of the key-LOV. In this example, there are three attributes called **Country**, **Region**, and **City**. The value of **Region** depends on what is selected in **Country**. Similarly, the value of **City** depends on what is selected in **Region**.

1. Create a new key-LOV that contains all the desired keys and options. For interdependent key-LOVs, you must assign a key that is unique within the key-LOV.

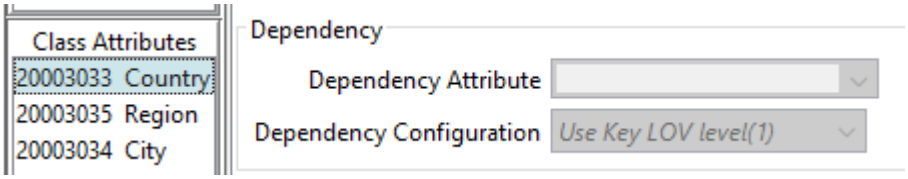


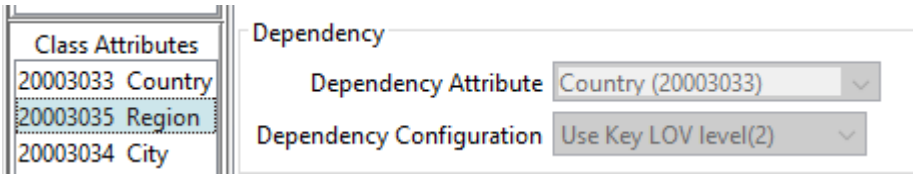
2. Create an attribute to represent each level of the key-LOV. The example requires three attributes: **Country**, **Region**, and **City**.

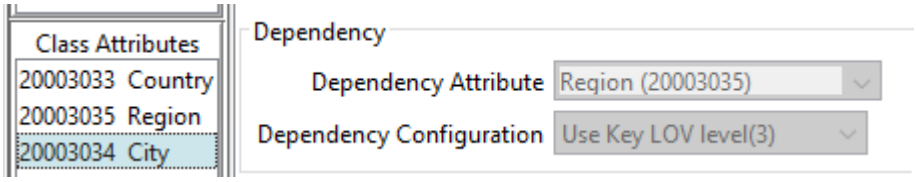
Level 1 Country	Level 2 Region	Level 3 City
1 USA		
	1.1 California	
		1.1.1 Los Angeles
		1.1.2 Cypress
		1.1.3 San Francisco
	1.2 Minnesota	
		1.2.1 Minneapolis
		1.2.2 St Paul
2 Germany		
	2.1 Bavaria	
		2.1.1 Munich
		2.1.2 Passau
	2.2 North Rhine-Westphalia	
		2.2.1 Cologne
		2.2.2 Duesseldorf

3. In the **Unit** box of each attribute, select **Format**.
4. In the **Format Dialog** box, click the **key-LOV** tab, enter the key-LOV ID, and click **OK**.
5. On the **Class Attributes** tab, add all the attributes that reference the key-LOV to the class.
6. For each interdependent attribute, assign the appropriate dependency.

   The top-level attribute does not have a dependency attribute value.

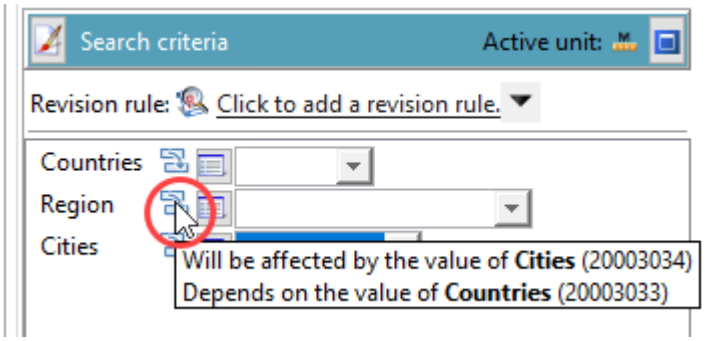






Tip:

Check the tool tip in the user interface to discover which interdependencies exist.



Preview a key-LOV entry

As you create a key-LOV, it is useful to view the list as it will appear on the Classification form. To preview a key-LOV, click the **Preview** button on the **Detailed Key-LOV Definition** pane.

Save a key-LOV entry

After all entries are added to the key-LOV, click **Save** .

Teamcenter displays the key-LOV in the **Existing Key-LOV** list and can be used, and reused, as an attribute format.

Remove a key-LOV entry

You can remove entries from a key-LOV, but you cannot delete the key-LOV definition itself.

You must be in create or edit mode to remove entries from a key-LOV.

Caution:

Deleting entries from a key-LOV that has been instanced compromises data integrity.

1. Choose the entry in the key-LOV that you want to delete.

2. Click the **Delete** button.

Teamcenter removes the entry from the key-LOV tree.

Modify a key-LOV entry

> Caution:
>
> While you can modify the properties of a key-LOV, you should approach such changes with caution. Changing entry keys or key-LOV IDs can compromise the integrity of your Classification data and is not recommended. Additionally, changing an entry description in a key-LOV that has been instanced changes the value in each and every instance in which it occurs.

1. Click the **Key-LOV** tab.

   Teamcenter displays the **Key-LOV** pane.

2. Choose the key-LOV from the **Existing Key-LOV** list that you want to modify.

   The system displays the key-LOV in the **Detailed Key-LOV Definition** pane.

3. Complete the necessary modifications (adding or inserting entries, removing entries, or modifying the description of an entry) to the key-LOV.

   > Caution:
   >
   > Key-LOV IDs must not be modified.
   >
   > You cannot modify a deprecated key-LOV entry.

4. Click **Save** .

   Teamcenter saves the modified key-LOV in the database. If the description of an entry is changed for a key-LOV that is instanced, the description is changed in each and every instance in which that key-LOV entry occurs.

Deprecate a key-LOV entry

You may need to change a key-LOV entry over time to respond to changing business situations. You may want to prevent a Classification user from using a particular key-LOV when classifying or updating. To support these scenarios, you can deprecate key-LOV values.

1. Click the **Key-LOV** tab.

   Teamcenter displays the **Key-LOV** pane.

2. Select the key-LOV from the **Existing Key-LOV** list that you want to deprecate.

Teamcenter displays the key-LOV in the **Detailed Key-LOV Definition** pane.

3. Click the **Edit** button .

4. Select the value you want to deprecate.

5. Click **Deprecate**.

6. (Optional) Set the following preferences to specify the behavior for deprecated key-LOVs.

Set to TRUE	To
ICS_allow_deprecated_lovs_on_update	Allows you to save a classification object that uses deprecated key-LOVs when you update that object.
ICS_allow_deprecated_lovs_on_copy	Allows you to use the **Save As** command on an item or item revision to create a copy of a classification object already classified using deprecated key-LOV values.

Note:

- Deprecated key-LOV entries are not visible to the Classification user when classifying a workspace object.
- Deprecated key-LOV entries are only visible to the Classification user when editing a classification instance if the **ICS_allow_deprecated_lovs_on_update** preference is set to **True**.
- Deprecated key-LOV entries are visible to the Classification user when editing a classification instance, but are preceded by the ⛔ symbol to show that the value is deprecated.
- Deprecated key-LOV values are available for searching.

Undeprecate a key-LOV entry

1. Click the **Key-LOV** tab.

   Teamcenter displays the **Key-LOV** pane.

2. Choose the key-LOV from the **Existing Key-LOV** list containing the value that you want to undeprecate.

   The system displays the key-LOV in the **Detailed Key-LOV Definition** pane.

3. Click the **Edit** button .

4. Select the deprecated value you want to undeprecate.

5. Click **Undeprecate**.

6. Click .

Creating and managing the attribute dictionary

Attribute dictionary overview

You use the Classification **Dictionary** pane to define and manage attributes. Attributes store values that distinguish one instance of a class from another. For example, within the **Sheet Metal Screws** class, the **length**, **diameter**, and **thread** attributes are used to distinguish one sheet metal screw from another. The goal of the attribute dictionary is to provide a repository for all defined classification attributes. Once defined in the dictionary, attributes can be used and reused within the classification hierarchy.

To avoid creating ambiguous Classification data, each attribute in the dictionary should uniquely describe a particular characteristic of an object. For example, consider the **diameter** generic attribute. While **diameter** may apply to a number of objects, it does not necessarily describe the same characteristic for each object. When applied to a screw, **diameter** can refer to the diameter of the thread, and when applied to a drill, it may refer to the diameter of the cut. To avoid such ambiguity, use concise, descriptive names when creating attributes.

Create attribute dictionary definitions

1. Click the **Dictionary** tab.

   Teamcenter displays the **Dictionary** pane.

2. Click the **Create** button to add a new attribute ID to the dictionary.

   Teamcenter displays the **New Attribute** dialog box.

   Use this dialog box to create (reserve) an ID for the new attribute within the dictionary.

   > Warning:
   >
   > Attribute values between -999 and 999 are reserved for Siemens Digital Industries Software functionality, such as the unit definition class. Do not create or modify any of these internal attributes, as these can get overridden by future features. When creating new attributes, it is advised to use positive integers greater than 999 for attribute IDs.

3. Perform either of the following steps to define the attribute ID:

   - Type a unique numeric identifier for the attribute in the **Attribute ID** box and click **OK**.

- Click **Assign** to generate the next available attribute ID and click **OK**.

  To determine the next available attribute ID, the program searches the database for the highest attribute ID that has previously been assigned and then generates an attribute ID that is one number higher than the highest number found in the database. If attributes have been deleted from the database, leaving a gap in the attribute ID numbering scheme, those numbers are not reassigned. If you cancel the attribute creation process, the assigned number is released for future use.

  > Note:
  >
  > Once an attribute ID is assigned, it cannot be modified.

  The new attribute ID is now displayed in the **Attribute Definition** pane. You define the remaining properties of the new attribute within this pane.

4. Type a descriptive name for the attribute in the **Name** box.

   The attribute name can be a maximum of 63 alphanumeric characters in length and is case sensitive. Attribute names do not have to be unique.

5. (Optional) Type a short descriptive name in the **Short Name** box.

6. (Optional) Type a default annotation in the **Default Annotation** box.

   > Caution:
   >
   > The annotation value must be unique within the class.

7. Define the units and formats for the attribute.

   > Tip:
   >
   > You can copy the format of an existing attribute by searching for it in the attribute dictionary and clicking **Assign Format** in the **Attribute Search** pane.

8. (Optional) Type the text in the **Help Text** dialog box that is displayed for this attribute in the Classification application.

9. (Optional) Type notes for internal use in the **Comments** box.

10. (Optional) Type text in the **User 1** and **User 2** boxes. Any values entered in the **User 1** or **User 2** boxes are shown in the attribute's tool tip in the Classification application.

11. Select **Encrypted** if you want to encrypt the value in the database. Selecting this option makes the attribute encrypted for all classes and views where the attribute is used. Therefore, you can no longer use this attribute as a criterion for a search.

    You can only encrypt values that have a string format.

12. (Optional) In the **Multi-Site Collaboration** pane, choose the sites with which you want to share this attribute definition.

13. If the new attribute should reference properties stored with the workspace object instead of with the ICO, select **Reference Attribute**.

14. To provide the most readable unit with the least number of leading or trailing zeros, select **Optimized Display**.

15. Click **Save** to add the attribute definition to the dictionary. Teamcenter saves the attribute definition and available for use when defining the Classification hierarchy.

Defining units and formats

Overview of units and formats

Use formats to enforce consistency of the attribute values you enter into the Classification database. Without formats, values can be entered in a variety of ways depending on the preference of an individual user, thus compromising data integrity. Classification allows you to define two formats for each attribute—a metric and a nonmetric format. This is useful when a single attribute needs to be rendered in different ways depending on, for example, the geographic location of the user (for example, Germany and the United States). In such cases, units are used to determine which format is applied to the attribute.

Because the goal of the attribute dictionary is to provide a repository of attributes that you can reuse, it is important to define two format/unit combinations for each numeric attribute. For example, to express the **diameter** attribute value in both nonmetric and metric units, you specify a separate format for each unit. The types of unit formats that you can choose from in the **Dictionary** pane are saved in the **Unit Definition Class** class in the classification hierarchy.

Use caution when using key-LOVs in combination with both systems of measure. When you convert from one system to another, Teamcenter does not convert the values. It displays the value that is contained by the equivalent key. For example, the following table displays key-LOV values for two systems of measure.

Metric key/value pair	Nonmetric key/value pair
01-2.5	**01-4.3**
02-3.0	**02-5.4**
03-3.5	**03-6.1**

If **3.0** is displayed as an attribute value and you switch to the nonmetric measurement system for that attribute, Teamcenter looks for the value assigned to the key **02** and displays that (in this case, **5.4**).

You must be careful when assigning values to the same key in two different measurement systems that these values accurately represent the converted values.

Attribute formats

Attribute formats define the configuration of values that can be entered in the Classification form for an attribute. Teamcenter provides five attribute formats: key-LOV, string, integer, real, and date and are described in the following table.

Attribute format	Description
Key-LOV	Associates a list of allowable values with an attribute. If a key-LOV value displays the ⛔ symbol, it is deprecated and cannot be used for classifying objects (although you can use it when searching for classified objects).
String	Defines an attribute as an alphanumeric string. You can define the following characteristics for a string attribute: string length and case-sensitivity. For string length, the system treats the value **99** as a special case. If you enter **99**, the resulting string length is 256 characters.
Integer	Specifies the numeric format that is appropriate for attributes that are usually quantified as whole units. You define the following characteristics for an integer attribute: • Maximum number of digits allowed • Whether the plus (**+**) symbol is displayed with positive numbers
Real	Numeric format that is appropriate for attributes that can be quantified as fractional units. You define the following characteristics for a real attribute: • Number of digits before and after the decimal point • Whether negative numbers are valid values

Attribute format	Description
	• Whether the plus (+) symbol is displayed with positive numbers
Date	Defines the configuration of date values entered for an attribute.

Define units and formats

1. Click the **Format** button in the **Metric Unit** and/or **Non-Metric Unit** sections of the dialog box.

   Teamcenter displays the **Format** dialog box. You perform the next steps within this dialog box.

2. Choose a format type for the attribute by clicking one of the following format tabs:

   - **KeyLOV**
   - **String**
   - **Integer**
   - **Real**
   - **Date**

3. Define the parameters for the attribute format.

Element	Description
Key-LOV tab	Displays the options used to define a key-LOV format for the attribute.
ID	Specifies the ID of an existing key-LOV to be associated with the attribute.
Preview window	Displays a preview of the selected key-LOV is displayed in this area.
Key-LOV Panel button	Moves to the key-LOV pane that is used to create and modify lists.
String tab	Displays the options used to define a string format for the attribute.
String Length	Defines the maximum number of alphanumeric characters to be allowed for the attribute value. The system treats the value **99** as a special case. If you enter **99**, the resulting string length is 256 characters.
Case-Sensitivity Selection dialog box	Defines the case sensitivity format for the attribute value. The following options are available:

Element	Description
	• **Upper and Lowercase** Allows mixed-case, uppercase, or lowercase values to be entered for the attribute. • **Uppercase** Causes the value entered for the attribute to be converted to uppercase. • **Lowercase** Causes the value entered for the attribute to be converted to lowercase.
Integer tab	Displays the options used to define an integer format for the attribute.
Number of Digits	Defines the total number of digits to be allowed for the attribute.
Display Format Selection window	Selects the display format for the integer attribute. The following options are available: • **Force Positive Number** Prevents the user from entering a negative number value for the attribute. • **Accept and Display + or –** Allows both positive and negative values to be entered for the attribute and causes the plus (**+**) or minus (**–**) symbol to be displayed as part of the value. • **Accept + or – but display – sign only** Allows both positive and negative values to be entered for the attribute. However, only the minus (**–**) symbol is displayed. Values without a corresponding symbol are assumed to be positive.
Real tab	Displays the options used to define a real format for the attribute.
Number of Digits	Defines the number of digits to be allowed before and after the decimal point.

Element	Description
Display Format Selection dialog box	Selects the display format for the integer attribute. The following options are available: • **Force Positive Number** Prevents the user from entering a negative number value for the attribute. • **Accept and Display + or –** Allows both positive and negative values to be entered for the attribute and causes the plus (**+**) or minus (**–**) symbol to be displayed as part of the value. • **Accept + or – but display – sign only** Allows both positive and negative values to be entered for the attribute. However, only the minus (**–**) symbol is displayed. Values without a corresponding symbol are assumed to be positive.
Date tab	Displays the options used to define a date format for the attribute.
Display Format Selection dialog box	Selects the display format for the date attribute. Several options are available for formatting dates and date/time combinations. All formats use some combination of the following elements: • **D** – Day • **M** – Month • **Y** – Year • **H** – Hour • **M** – Minute • **S** – Second
OK button	Accepts the format definition and dismiss the **Format** dialog box.
Cancel button	Closes the **Format** dialog box without applying any changes to the attribute format.

4. Click **OK** to accept the format definition and dismiss the dialog box.

> Caution:
>
> Teamcenter always converts between the metric and nonmetric formats that you specify in the attribute dictionary. If, during conversion, the converted value's number of digits is larger than the specified target format allows, Teamcenter truncates the excess digits and displays an erroneous value. For example, an attribute has a nonmetric format with **REAL(3.5)** and a metric format of **REAL(4.3)**. If you try to convert 622 inches to a metric value, the correct converted value is 15,800 millimeters. Teamcenter, however, truncates the converted value to 1580 to fit the defined format (**REAL(4.3)**) and displays this incorrect value in the Classification application.
>
> Make sure that the integer and real formats that you define here display a sufficient number of digits before and after the decimal point for all conversion needs.

5. Select a unit for the measuring system you are using. The units available in this list are stored in the **Unit Definition Class** class. The label that is shown is displayed next to the attribute value box in the Classification form.

6. (Optional) Type a default value in the **Default Value** box. This value is then shown for this attribute anywhere it is used in a class.

   - This value can be overwritten in the **Attribute Details** pane on the **Class Attributes** and/or **View Attributes** tabs.
   - If you select a default value that contains a key-LOV, deprecated key-LOV values are not shown in the list.
   - If minimum and/or maximum values are specified for this attribute, Teamcenter immediately validates the default value against these values.

7. (Optional) Set a minimum and/or maximum value for the attribute.

   This value limits the range that a Classification user can use when entering numerical attribute values (integer or real format). Use the following preferences to determine whether a Classification user must adhere to these ranges.

Set this preference to true	To
ICS_enforce_min_max_ constraints_on_update	Prevent Classification users from saving or updating classification instances or classified objects whose attribute values are beyond the specified range.
ICS_enforce_min_max_ constraints_on_copy	Prevent Classification users from creating a copy of an existing classification instance or classified object containing attribute values beyond the specified range. This includes revising or using the **Save As** command

Set this preference to true	To
	on an item or item revision outside the Classification application.
ICS_allow_min_max_ range_override	Allows the Classification administrator to specify maximum or minimum values in the class or view that are outside the range set in the attribute dictionary.

Assigning reference attributes

Overview of reference attributes

When you create an item or item revision in Teamcenter, you store different kinds of information as properties in the item, item revision, or a form related to the item or item revision. When you classify the workspace object, you may want to search the classification hierarchy for objects with these properties. Classification provides you with reference attributes to access this information. Reference attributes can refer to information stored in:

- The classified workspace object.
- The master form of the classified workspace object.
- An object related to the classified workspace object.

You define reference attributes in the Classification Admin application and display them in the Classification application.

There are two locations where you can define reference attributes:

- When you create a new attribute in the **Dictionary** tab, you can designate it to be a reference attribute.
- You can designate an attribute to be a reference attribute when you are defining it in the **Class Attributes** pane.
- A reference attribute that refers to a date format always displays the date and time based on the time zone in effect on the server.

Caution:

- When creating a reference attribute in Classification Admin, make sure you format this attribute to match the Teamcenter property which it references. For example, if you create a reference attribute for the Teamcenter **Alias_ID** property that contains multiple values, you must select the **Array** property when creating the classification attribute so that the reference attribute can also show the multiple values that the Teamcenter property contains.

- If you turn an existing dictionary attribute into a reference attribute, you must also specify the attribute to be a reference attribute in the class containing it. This is only possible if the attribute has not yet been assigned any values.

By default, the value field of reference attributes is adjusted to the length of the value. If the reference attribute is empty, the attribute value field is displayed as follows:

To avoid this when searching, set the **ICS_reference_attribute_use_format_for_display** preference to **true**. The format definition of the attribute is then used to determine the size of the value field in the **Search criteria** pane.

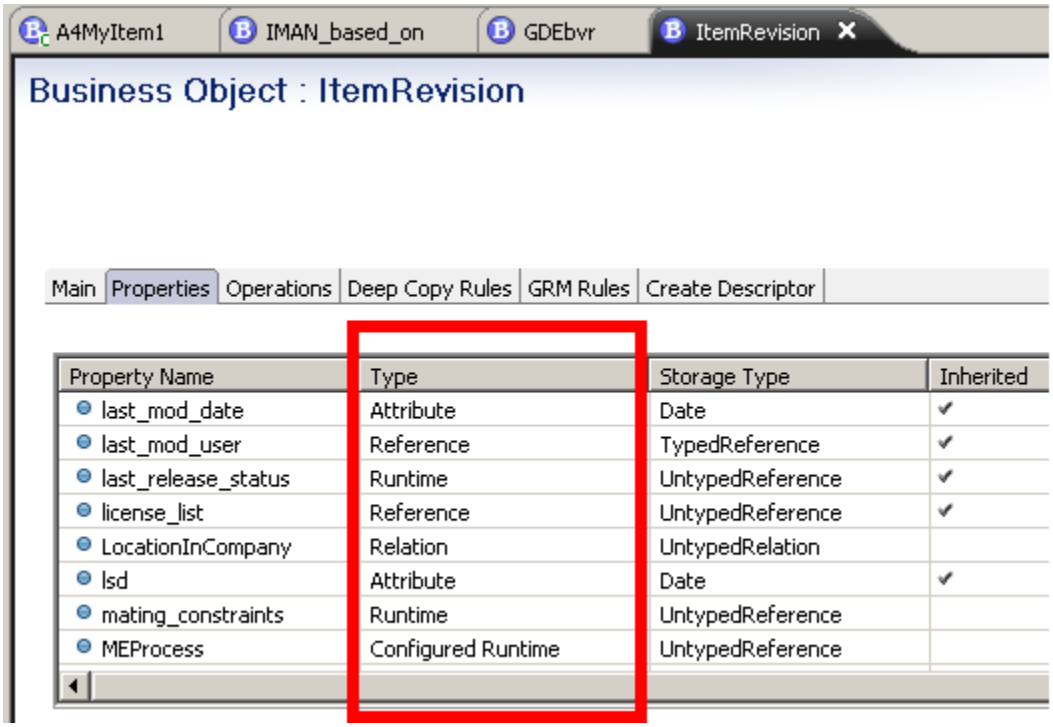
Reference attributes are read-only in the Classification application. You can search for them only if the attribute:

- Is a database attribute or references a database attribute.
- Has a searchable format such as number, date, or string.

Note:

- Run-time properties are not searchable attributes.
- The autofilter is not available when displaying reference attributes in the Classification application.
- Teamcenter ignores an attribute's format when you designate the attribute to be a reference attribute.

When configuring reference attributes in the **POM Attribute** pane, you can only configure those attributes whose type in the Business Modeler IDE are marked as **Attribute**. All others types (for example, **Runtime**, **Configured Runtime**, **Reference**, or **Relation**) can only be configured using the **Type Properties** pane and therefore cannot be used for searching.



Configure reference attributes

1. Select the **Reference Attribute** check box and click **Configure**.

   The **Configure Reference Attribute** dialog box appears. For each of the reference attribute types listed on the left, you can select either a POM attribute or a type property. When you do so, the available POM attributes/type properties appear in the list at the right. Only those type properties that reference a POM attribute are searchable attributes.

   You can select from the following reference attribute types:

Classified Object	Selects a POM attribute or a type property of the classified object, typically item or item revision, as the source of the reference attribute. For example, assume it is your company's policy to classify items and you save an object description with each item you create. By selecting **Item** from the **Type Property** list, you can select from all the available properties in the **Type Property** list. If you select **current_desc**, the object description that you saved with the item appears in the reference attribute's name box in Classification.
Master Form Attribute	Selects a POM attribute or a type property of the master form, typically item master form or item revision master form, as the source of the reference attribute. For example, you want to see the serial number of the item in Classification. By default, this information is stored in the item master form. Select **Type→Item**

	to reference this information. When you select **serial_number** from the **Type Property** list, the reference attribute shown in the class attributes in the Classification application displays the serial number that is stored in the master form of the classified item.
Related Object	Specifies an attribute of an object that is related to the classified workspace object as the source of the reference attribute. You can specify which particular relation the object should have in the **Relations List** or check the **Any relation** option to find all objects in the specified POM class.

2. Select the type of reference attribute. Depending on the type you select, the pane on the right provides you with the POM hierarchy or type hierarchy (or both) in which to find the attribute to which you want to refer.

3. Find the POM class or type property in the hierarchy.

4. Select the desired attribute from the list.

5. (Optional) Select **Ignore the selected type** or **Ignore the selected POM class**. Names of attributes or type properties are not unique. It is possible, for instance, to have two attributes with the same name but belonging to different POM classes. when you select **Ignore the selected POM class**, the system searches for the attribute of a specific name in the loaded object, regardless of which POM class it is in. The same behavior occurs with type properties.

6. (Optional) Select **Select attribute from item**. If, for instance, your company has a policy of classifying item revisions, you can still show an attribute found in the item revision's item by checking **Select attribute from item**.

7. If you choose to refer to an attribute found in an object related to the classified workspace object, you can select which relationship from the **Relations** list or disregard the relationship by selecting the **Any relation** check box.

8. Click **OK**.

Example: show release status in Classification

You can use a reference attribute to display the release status of a workspace object in Classification.

1. Create a new attribute with a **String** format (80 characters in length).

2. Select the **Reference Attribute** check box and click **Configure**.

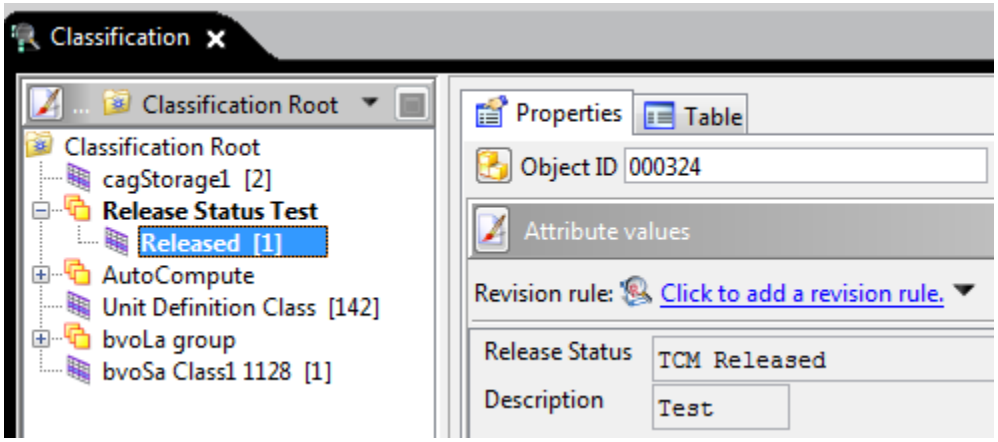
3. In the **Configure Reference Attribute** dialog box, do the following:

   - In the **Types of Reference Attribute** section, select **Classified Object**.

- From the **Type Property** list, select **Types→Item Types→Item**.
- In the **Properties** list, select **release_status_list**.
- In the **Advanced Options** section, select **Ignore the selected type**.

4. Assign the new attribute to a class and classify a released item or item revision in this class.

Teamcenter displays the status of the new attribute in the Classification application.



Note:

You cannot use this attribute as a search criteria.

Reveal all types for selection

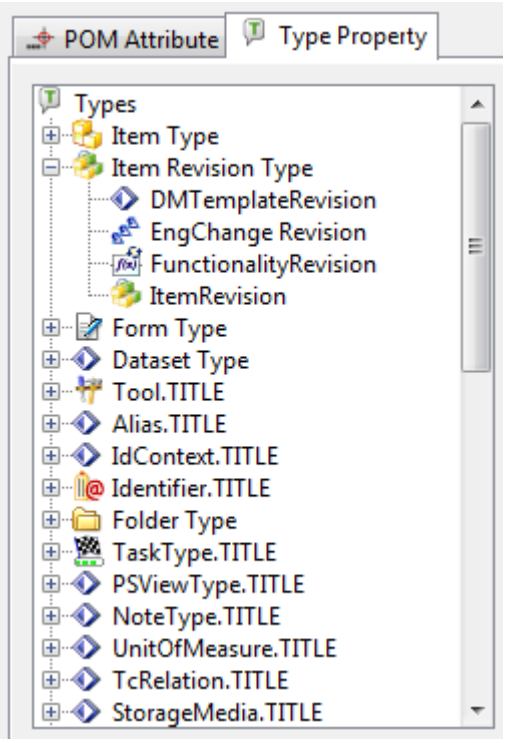
The **Type Property** list does not always contain all business object subtypes. In particular, it does not list custom business object subtypes.

To make a more complete list of business object subtypes available for selection:

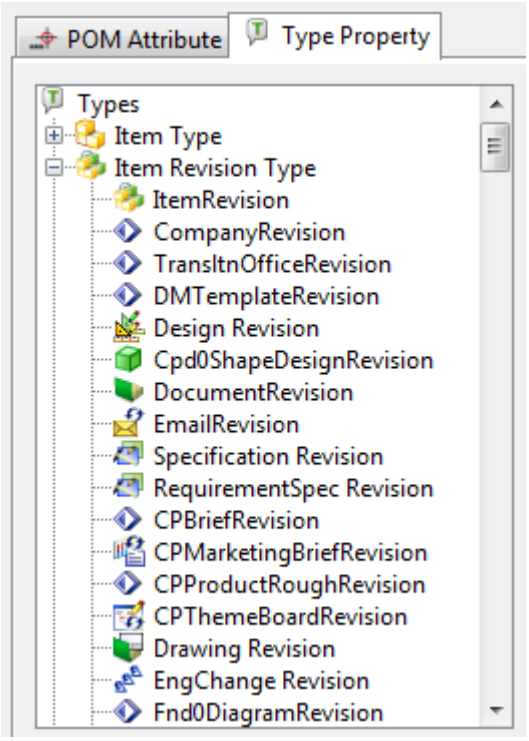
- Add the business object type to the **TYPE_DISPLAY_RULES_list_types_of_subclasses** preference.

  For example, to see the extended list of item revision business object subtypes, add the **ItemRevision** entry to the list of preference values.

Before



After



Optimizing attribute values

You can display *optimized* attribute values in Classification. When Teamcenter optimizes an attribute value, it provides the most readable value with the least number of leading or trailing zeros. For example, 1 km is easier to read than 1000 m.

Attribute value entered	Optimized attribute value
1723000 µF	1.723 F
9640000 pF	9.64 µF
4372.3 Ω	4.3723 kΩ
3756794 mm	3.756794 km

> Note:
>
> The number of decimal places actually shown is dependent on the number of decimal places specified in the **Number of decimal places** attribute in the unit definition.

The optimization is based on the unit definitions assigned to the attribute. These unit definitions are managed in the **Unit Definition** class in the Classification application. You specify whether a unit is available for optimization in the **Ignore for Optimization** attribute in the unit definition ICO.

Specify whether Teamcenter should display optimized units in the Classification Admin application in one of two ways:

- At the attribute level in the **Dictionary** pane—anywhere this attribute is used, its unit is optimized.
- At the class level in the **Class Attributes** pane—the value of this attribute is optimized in this class only.

Teamcenter always takes the storage unit into consideration for optimization even if the **Ignore for Optimization** option is turned on for that unit in the **Unit Definition** class.

When you display the attribute in the storage unit (either because optimization did not take place or you selected the storage unit from the unit list),Teamcenter shows the number of decimal places specified in the attribute definition and not the number specified in the unit definition. This is done to ensure that the value you enter is saved by the system and not rounded by the optimization process.

If you want to view the full attribute value in different units without it being rounded to the number of digits specified in the unit definition, you can set the **ICS_override_unit_definition_precision** preference to **true**.

Modify attribute dictionary definitions

Siemens Digital Industries Software recommends that you modify only attributes that have not been instanced. While it is possible to modify the properties of an attribute, with the exception of the attribute ID, such changes should be approached with caution. Changing an attribute property can compromise the integrity of your Classification data.

1. Click the **Dictionary** tab.

   Teamcenter displays the **Dictionary** pane.

2. Locate the entry in the attribute dictionary that you want to modify.

3. Select the row in the attribute table that contains the attribute that you want to modify. The properties of the selected attribute are displayed in the **Attribute Definition** pane.

4. Click the **Edit** button . You must be in edit mode to modify an attribute.

   > Note:
   >
   > If you are not in edit mode, you can modify the properties but you cannot save the modifications to the attribute dictionary.

5. Change the properties of the attribute definition.

> Note:
>
> The attribute ID cannot be modified.

6. Click **Save** to save your changes.

   Teamcenter saves the modified attribute in the attribute dictionary.

Delete attribute dictionary definitions

> Note:
>
> You can only delete an attribute that has not been referenced.

1. Click the **Dictionary** tab.

   Teamcenter displays the **Dictionary** pane.

2. Locate the entry in the attribute dictionary that you want to delete.

3. Select the row in the attributes table that contains the attribute that you want to delete. The properties of the selected attribute are displayed in the **Attribute Definition** pane.

4. Click the **Edit** button. You must be in edit mode to modify or delete attributes.

5. Click the **Delete** button.

   Teamcenter displays the **Delete Confirmation** dialog box.

6. Verify that you are deleting the correct attribute and click **Yes** to complete.

   Teamcenter removes the attribute from the database.

Add user-defined data to an attribute

You can add metadata to a dictionary attribute that is displayed as a tool tip on the attribute. You can subsequently edit the user-defined attribute values as you would any other attribute values.

1. In the Business Modeler IDE application, add a persistent property on the **unct_dict** business object.

2. Make the property modifiable.

3. Shut down the server and deploy the schema.

4. If you are using Multi-Site, update the schema of all other sites that exchange classification data.

5. In the **Dictionary** tab in the Classification Admin application, click the **Extended Properties** tab.

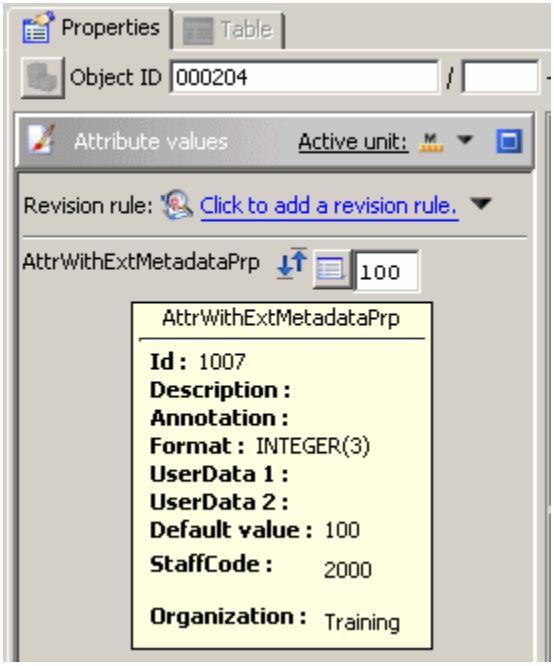
   This tab is displayed only after you add the metadata attributes to the **unct_dict** business object and deploy it.

   Teamcenter displays the new properties that you created in the Business Modeler IDE application.

6. Add the desired data to the property boxes.

7. Add the name of the attribute to the **ICS_attribute_displayable_properties** preference.

Teamcenter displays these values in a tool tip on the attribute in the **Properties** pane of the Classification application.

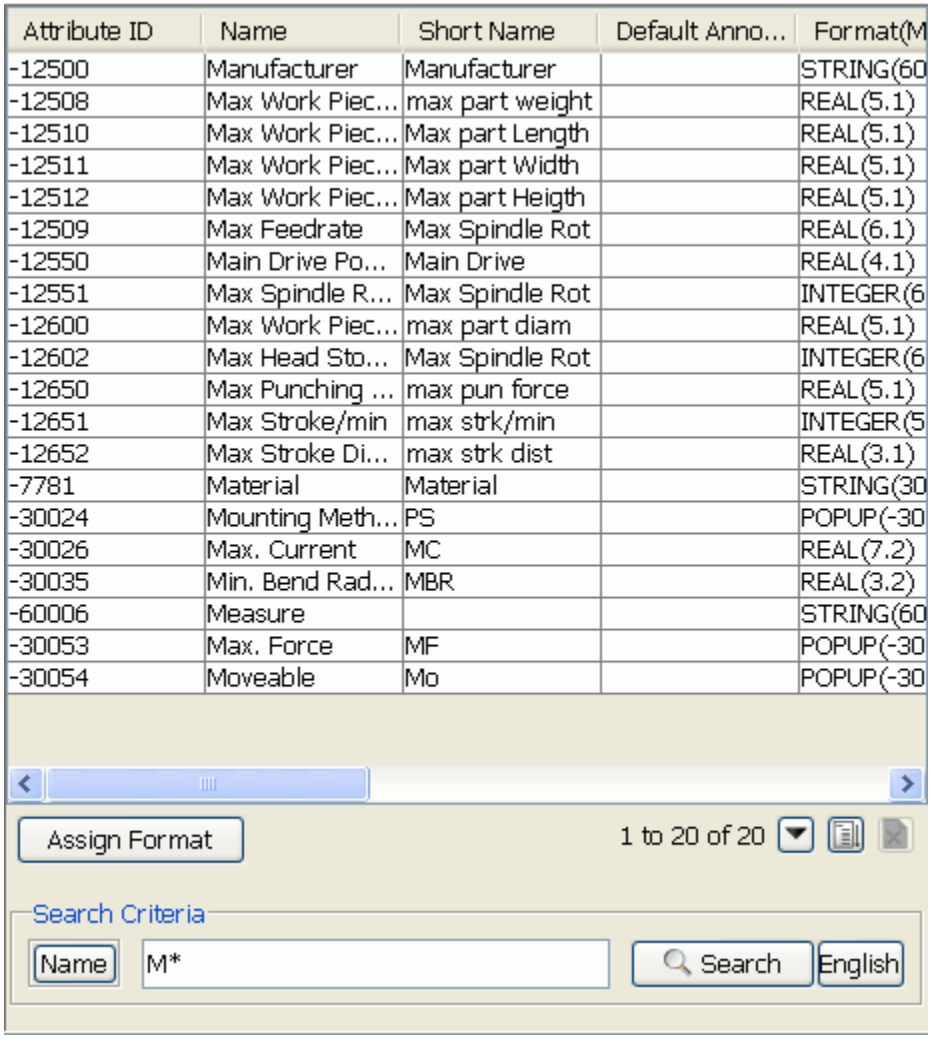


In the figure, **StaffCode** and **Organization** are user-defined attributes.

> Tip:
>
> You can search for the newly created attributes.

Search the dictionary for an attribute

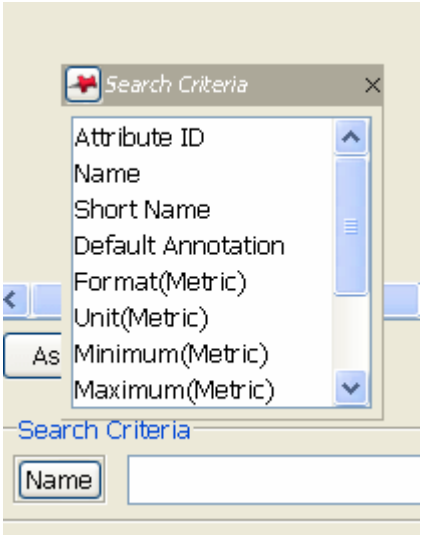
You can use the search function in the **Dictionary** pane to quickly locate and display an attribute. The attributes are listed in the **Attribute search pane**.



Define the search criteria as follows:

1. Click the **Search Criteria** button to display a list of the available search criteria in the **Search Criteria** dialog box.

> Caution:
>
> The use of a wildcard (*) is not permitted for searching by attribute ID. To search by attribute ID, you must enter the exact ID or leave the field blank to search for all attributes.



2. Choose the attribute property to narrow your search.

3. Type search criteria text that corresponds to the property selected. For example, if you are searching by name, enter the first letters of the attribute name followed by a wildcard character.

4. If localization is enabled at your site, select the language in which to search.

   Caution:

   Selecting a language in this search sets the **TC_language_search** preference interactively, which affects all Teamcenter localization.

5. Click the **Search** button.

   The search result statistics are displayed below the table. If more than 20 attributes are found, the first 20 attributes are displayed in the table.

   1 to 20 of 72

   When you perform a search of the attribute dictionary, Teamcenter displays the first page of results in the Attributes table. Click the **Load Next Page** button ▼ or the **Load All** button to load additional pages of results into the table.

6. (Optional) To copy the format of an attribute to a new attribute that you create in the **Dictionary** pane, select the attribute and click **Assign Format**.

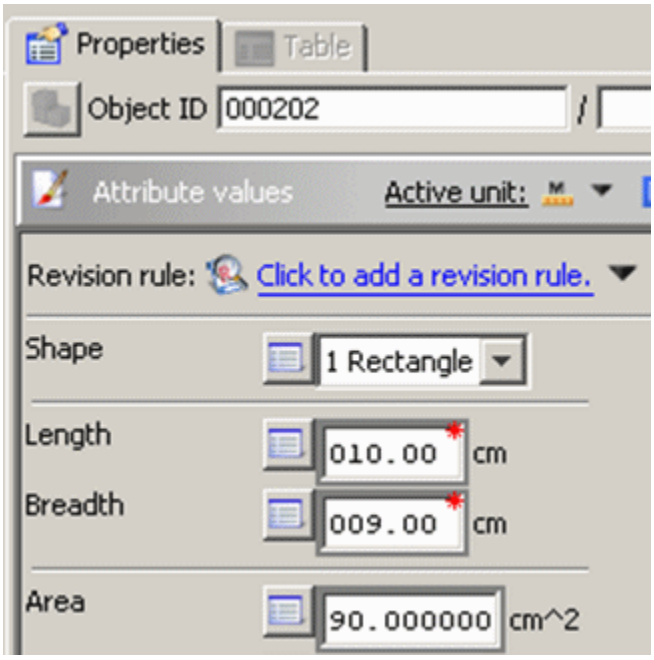
Using custom logic to automatically compute attribute values

Automatically computing attribute values

Teamcenter can automatically compute attribute values based on other attribute values within the class or view or based on attribute values of the object being classified. It uses custom logic that you assign to a predefined operation in the Business Modeler IDE application.

Using this logic, you can set attributes to be mandatory, protected, or invisible to the Classification user. For example, you can select from a variety of shapes and, based on the shape, enter attribute values required to calculate the area. If you select **Circle**, you must enter a diameter and Teamcenter can automatically calculate the area. If you select **Square**, you must enter a length and width to enable the area calculation. **Diameter**, **Length**, and **Width** are attributes that are only displayed when you select the corresponding shape.

In the following figure, the **Area** value is automatically computed after you enter **Length** and **Breadth** values. In this example, if you select **Rectangle** as the **Shape** value, the **Diameter** box is hidden.



If an attribute is designated as mandatory, you can only hide it if at least one of the following is true:

- It is marked as **Auto Computed**.
- It has a default value defined for the selected format.

Teamcenter displays automatically computed values before saving so you can verify them before committing them to the database. Note that you must press the Enter key for the values to automatically compute.

Classification users can override automatically computed attribute values unless these values are also protected.

> Caution:
>
> You can modify the layout of attributes in a view. When doing so, you can associate the attribute layout to specific attributes. If you make extensive use of autocomputation and turn off visibility on attributes, this can create unwanted results, especially when you use frames and separators in the view layout. You must be careful when creating layout elements using attributes that can be hidden.

Register custom logic

Using the Business Modeler IDE application, you can attach custom logic to a predefined operation.

1. Create a new extension definition.

   When you create a new extension, you must supply the following parameters to the Business Modeler IDE in the **New Extension Definition** dialog box:

   Name: *function_name-of-the-custom-logic*
   Language: ANSI_C
   Library: *library-where-the-custom-logic-is-created*

   When you define the availability of the extension to the business object, you must use the following parameter values:

   Business Object: ICS
   Operation Name: BMF_ICS_exec_custom_attribute_logic
   Extension Point: BaseAction

2. Attach the extension definition to the user exit.

   The name of the user exit is **ICS** and the operation to select is the **BMF_ICS-exec_custom_attribute_logic** operation.

3. Save and deploy the data model.

Creating custom logic

The custom logic for automatic computation of attribute values consists of three parts:

- Includes statements
- Main program

Teamcenter uses the main program when creating the Business Modeler IDE extension definition. This becomes the entry point for the computation logic. The program registers all the attribute dependencies and computing functions.

- Computing functions

  The computing functions contain the logic for manipulating attributes. Each function is registered against an automatically computed attribute and, once registered, is called directly from Teamcenter during normal execution.

Controlling access to classification objects

Classification access control overview

The Classification Admin access control feature allows you to control access to Classification objects (ICOs) and hierarchy components (groups and classes). This feature is an extension of the Access Manager (AM) application and employs the AM tree of rules and access permissions to define access to objects. Rules created for Classification groups and classes are inserted into branches of the rule tree.

The basic concepts of access control are the same for both the Classification Admin and Access Manager applications.

> Caution:
>
> To maintain consistency, Classification rules should not be edited in the Access Manager application. Additionally, the Classification access control feature and the Access Manager application cannot be used simultaneously.

You can suppress the display of individual groups and classes in the hierarchy based on group, role, or user-specific controls. This enables you to customize the display of the Classification hierarchy tree, providing users with only the Classification data that is relevant to their tasks. Component display suppression affects the display of the hierarchy tree in both the Classification and Classification Admin applications. When visibility of a hierarchy component is suppressed, the display of the component's children is also suppressed.

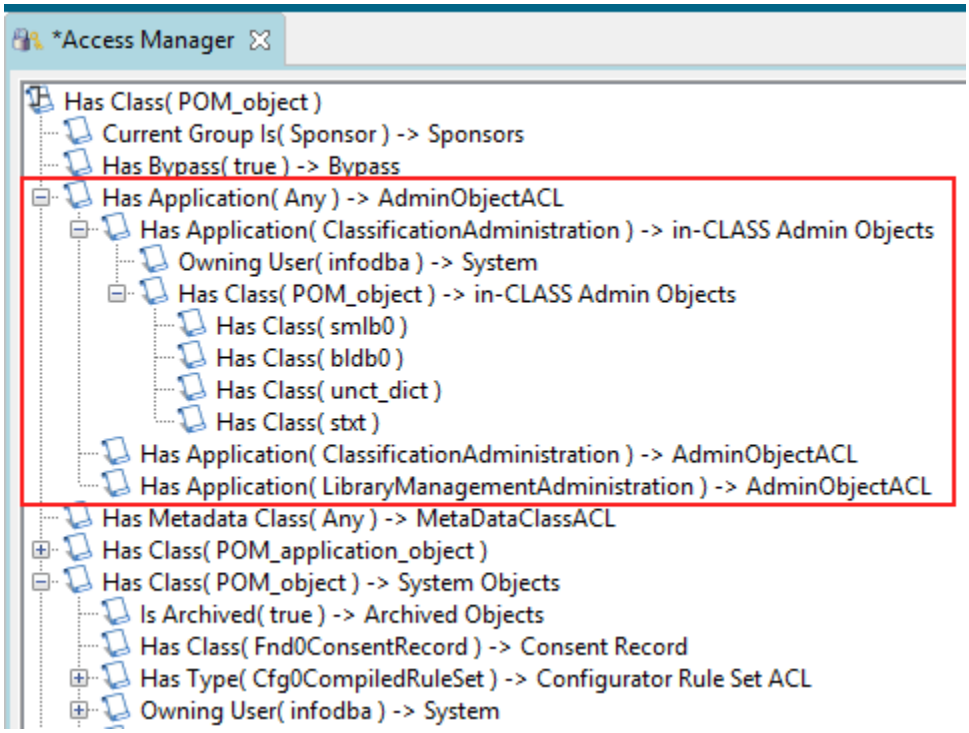
Creation and modification of groups, classes, and subclasses is controlled, enabling different individuals to be responsible for maintaining different parts of the hierarchy.

The following restrictions apply:

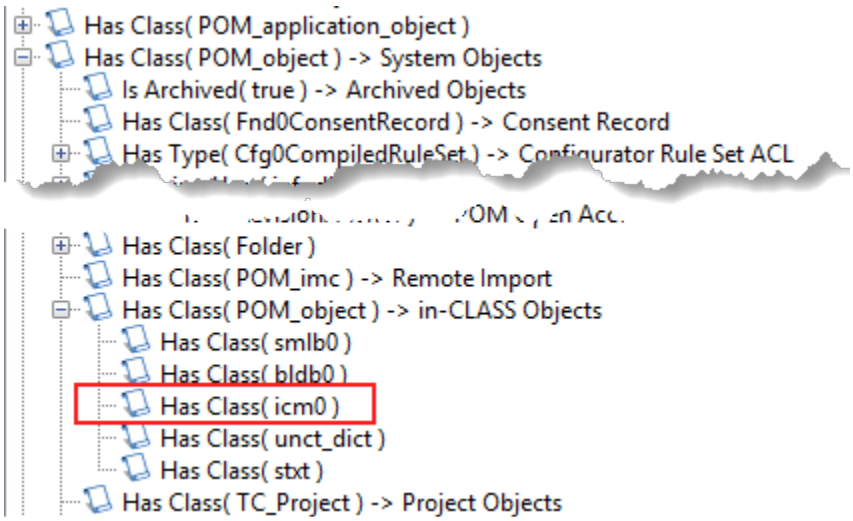
- The IDs of classes and groups to which you apply privileges must not contain spaces.
- To maintain consistency, Classification rules should not be edited in the Access Manager application. Additionally, the Classification access control feature and the Access Manager application should not be used simultaneously.

- Only members of the system administrator group can modify access privileges.

All access rules created in the Classification Admin application except those pertaining to classification objects (ICOs) are stored in the following location in the Access Manager rule tree:



Access rights to stand-alone ICOs are stored in the following location:

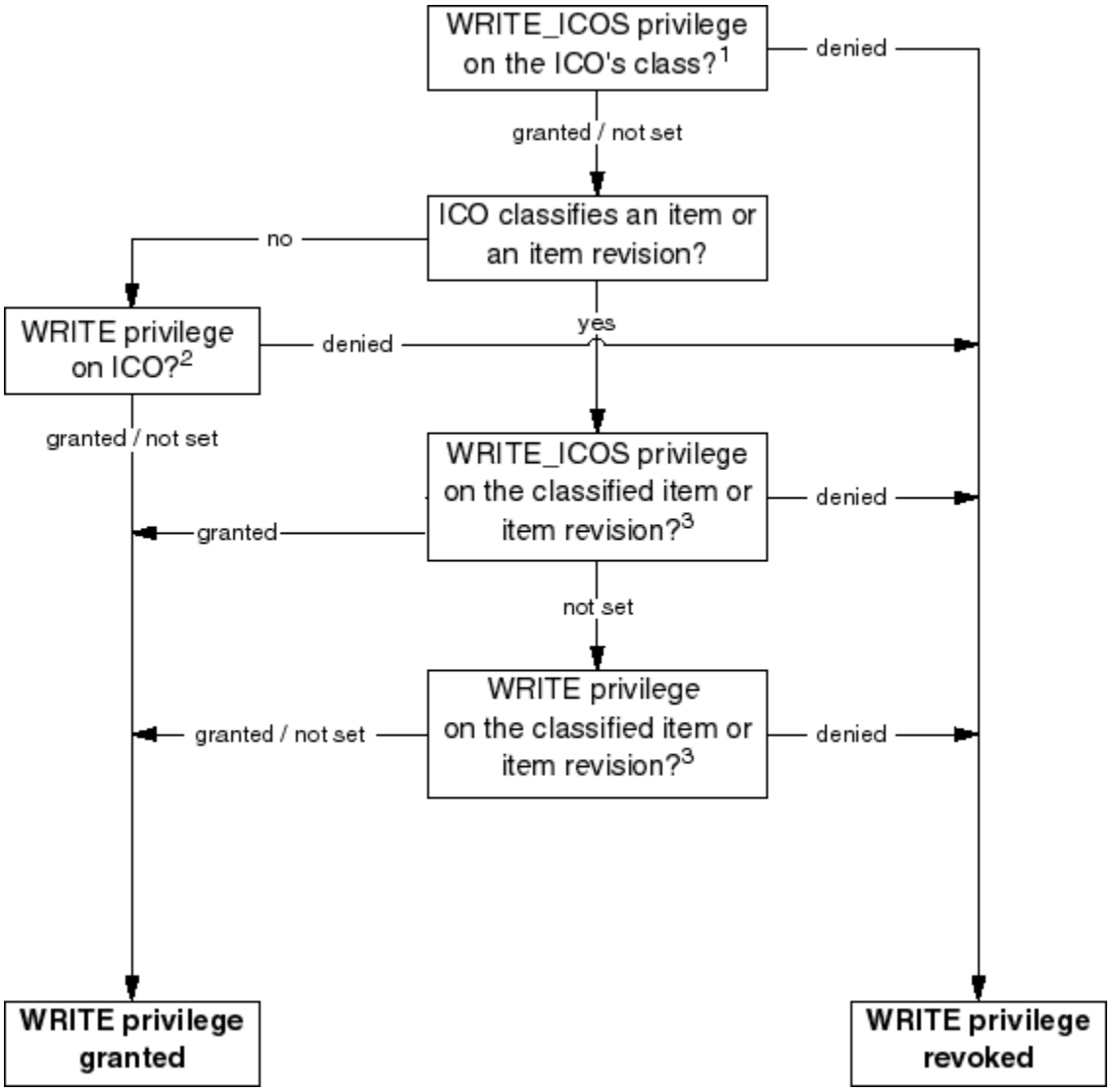


Caution:

Do not manually create access rules for administrative objects in the **Has Class(POM_object)** group as they are overwritten by entries higher in the tree.

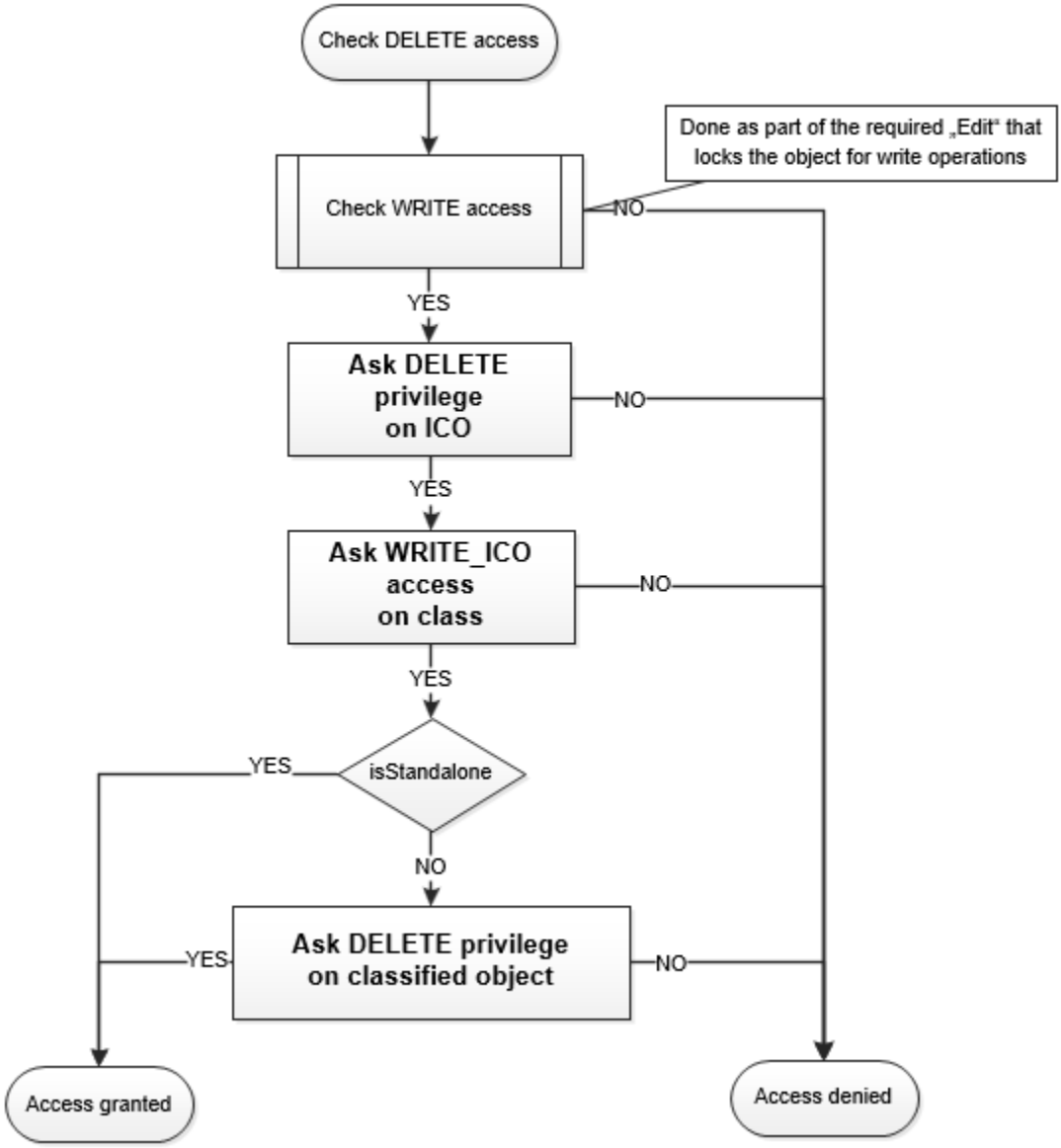
ICO protection

Privileges applied to groups and classes determine which user, or groups of users, can view, add, or modify the Classification objects (ICOs) associated with the group or class. The following figure illustrates how privileges are evaluated to determine if a user has the privileges required to update an ICO.



1. Set in Classification Admin, **privileges on class definition**.
2. Set in Classification Admin, **privileges on ICOs**.
3. Set in Access Manager application or as **Object ACL** on the workspace object (for example, in My Teamcenter).

The following figure illustrates how privileges are evaluated to determine if a user has the privileges required to delete an ICO.



Classification access privileges

The following table describes how access privileges apply to Classification objects, and whether the privileges are inherited by children of the selected object within the hierarchy.

Privilege	Used by Classification access control	Purpose	Inherited?
Read (R)	Yes	Controls visibility of a group or class in the hierarchy tree. When read access is denied, the object is not displayed in the tree. This privilege overrides privileges set for the Classification objects (ICOs) of a class. If read privileges are denied at the class level, but granted for the ICOs of the class, the ICOs are inaccessible. Note: Revoking the read privilege cannot be overridden in a subclass; however, granting the Read privilege can be overridden in a subclass.	Yes
Write (W)	Yes	Controls whether a group or class can be modified, and when applied to a class, controls whether subclasses and views can be added to the class.	Yes
Delete (D)	Yes	Controls whether a group or class can be deleted from the hierarchy. Restrictions on deleting groups and classes may prevent you from deleting an object to which you have delete privileges. For example, you cannot delete a class that has been referenced, regardless of the privileges granted.	Yes
Change (C)	Yes	Controls the right to define access control privileges.	Yes
Promote (p)	No	Not applicable.	Not applicable.
Demote (d)	No	Not applicable.	Not applicable.
Copy (c)	No	Not applicable.	Not applicable.
Export (X)	No	Not applicable.	Not applicable.
Import (I)	No	Not applicable.	Not applicable.
Transfer-out (x)	No	Not applicable.	Not applicable.
Transfer-in (i)	Yes	Controls whether ownership of an object can be transferred from one site to another.	No
Change Ownership	No	Grants or revokes the privilege to select shared sites for sharing classification data in Multi-Site Collaboration.	No
Publish	Yes	Controls whether the group or class and its children can be shared to other sites. Additionally, if this privilege is denied, the user will not be able to modify the list of shared sites in the class or group definition.	No
Subscribe	No	Not applicable.	Not applicable.
Write ICOs	Yes	Controls whether objects can be classified and stored within a class. Also controls whether existing Classification objects (ICOs) can be modified. Attributes of the ICOs associated with part family members cannot be modified unless write access is granted to the part family template.	Yes

Note:

ICOs that classify workspace objects are subject to further restrictions.

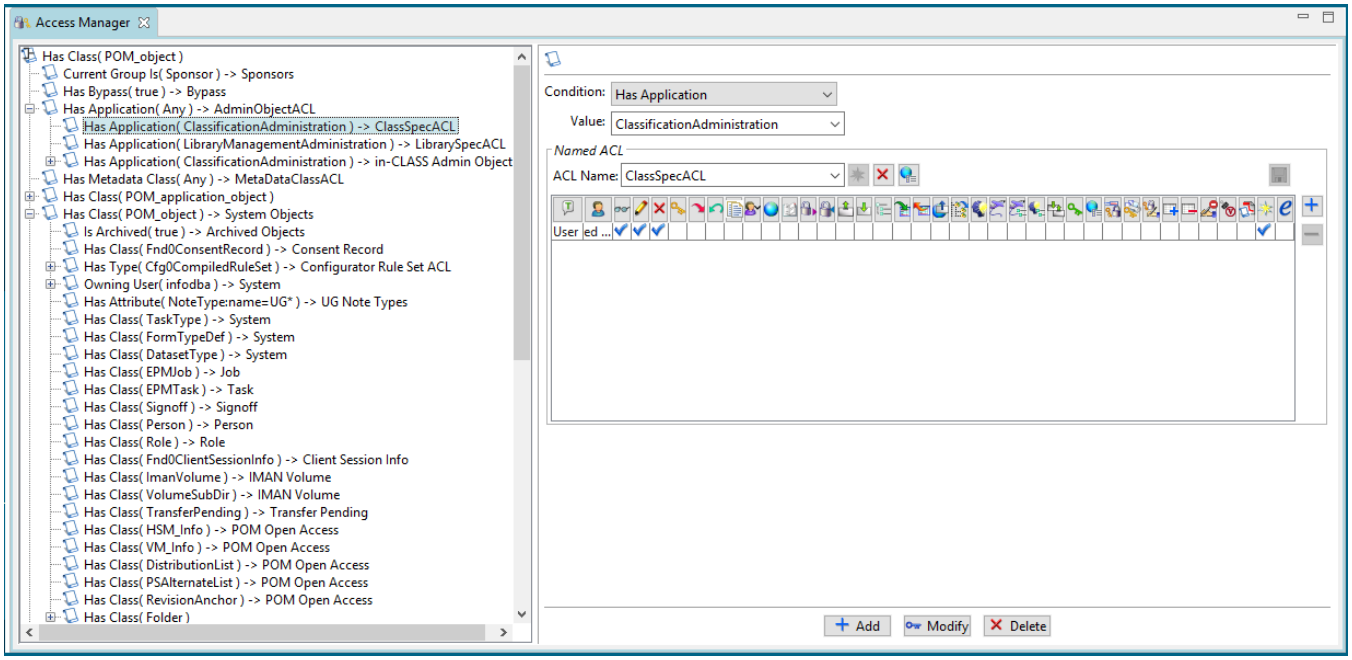
Granting access to administrative objects to non-dba users

For security purposes, only dba users have access to classification administrative objects. However, there is often need for users without dba permissions to modify the classification hierarchy. A system administrator can set up an access rule that allows such users full access to the classification hierarchy. These rules must be created in the **Has Application(Any) → AdminObjectACL** group. In this group, there are several conditions that apply to the classification hierarchy admin objects:

- **ClassificationAdministration**
- **LibraryManagementAdministration**

All permissions to admin objects are stored in these locations.

A typical rule for granting access to admin objects may look like this:



With this rule in place, this user has access to all classification administration objects. To restrict that access to specific branches of the hierarchy, you must **create a classification access rule**.

Create a classification access rule

To control and protect your data,various conditions or rules are applied. These rules are global, they affect your entire Teamcenter site. Rules that apply to Classification data specify a named access control list (ACL) that is applied to the object.

The AM rule tree displays the rules in force at your site. Each rule is assigned some level of relative importance. The rules near the top of the tree take precedence over rules lower in the tree. The Classification Admin **Access Control** pane displays only the portion of the tree that applies to Classification objects.

1. Click the **Hierarchy** tab.

   Teamcenter displays the **Hierarchy** pane.

2. Choose the group or class in the hierarchy tree that is affected by the rule.

   Teamcenter displays the definition pane for the group or class.

3. Click the **Edit** button on the toolbar to activate the definition pane.

4. If defining rules for a class, click the **Access Control** tab to display the **Access Control** pane. When working with groups, the **Access Control** pane is displayed on the definition pane when you select the group.

   The **Access Control** pane displays the two AM rule tree roots related to Classification, along with the standard Teamcenter **Named ACL** dialog box.

5. Choose the AM Rule tree root node that represents one of the following types of rule you want to define:

   - Privileges on class/group definition
   - Privileges on ICOs

   Privileges on class/group definition applies rules to the group or class and its descendants. Privileges on ICOs applies controls to the ICOs that are created within the group or class.

6. Choose a named ACL from the list or create a new ACL.

   Teamcenter displays the access control entries (ACEs) that comprise the named ACL in the table.

7. (Optional) Modify the access control entries.

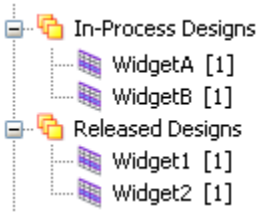
8. Click the **Add** button located at the bottom of the **Access Control** pane. Teamcenter adds the ACL to the rule tree.

9. Order the rules in the tree, as required, by using the up-arrow and down-arrow buttons next to the tree. These rules are evaluated in order from top to bottom when a user attempts to access an object. Thus, a rule directly beneath the root takes precedence over one further down the tree.

10. Click **Save** on the toolbar to save the new rule.

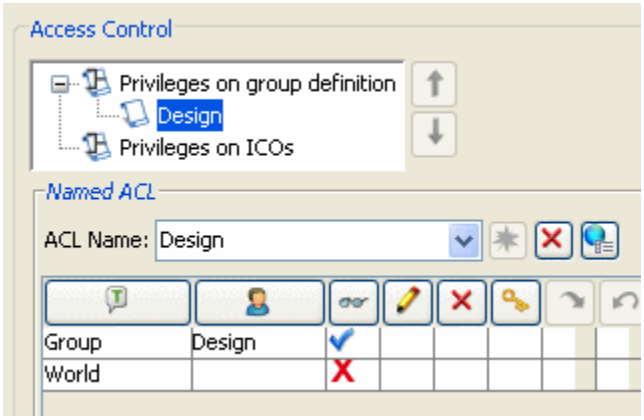
Example: controlling the display of the hierarchy tree for Classification users

ABC Corporation manufactures widgets and uses Classification to classify their design data, using the following hierarchy structure.



In-process designs are considered to be strictly confidential and only the Design work group is allowed to view them prior to release.

To suppress the display of this Classification hierarchy data for all users except those in the Design work group, protections are applied to the In-Process Designs Classification group, as shown next.



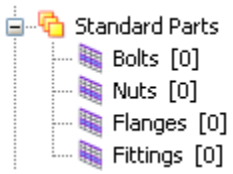
By granting read privileges to users in the Design work group and denying read privileges to the world, the data in the In-Process Design Classification group and all of its child classes are only visible to users who have a role in the Design work group. In addition, the protected objects are only returned as query results to users in the Design work group.

> Note:
>
> Read privileges to child classes of a read-protected Classification group cannot be granted unless the user is one to whom read privileges are also granted at the Classification group level.

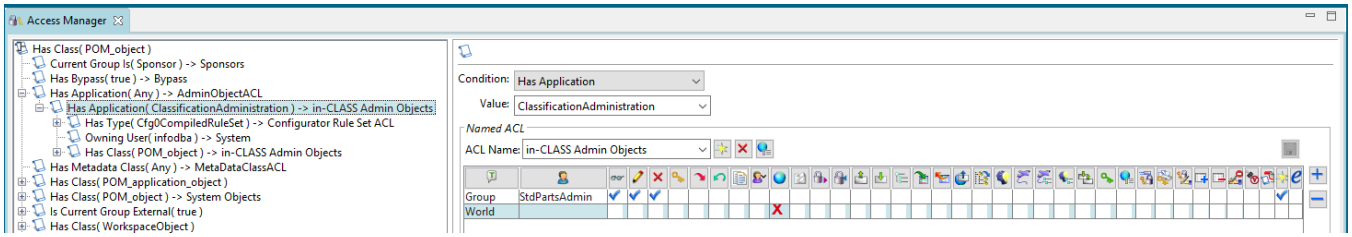
For example, John Smith, a member of the Marketing work group, cannot be granted read privileges to the Widget A class because it is a child class of the In-Process Design group. As a child, it inherits the privileges of the parent group, and according to the rule defined in the figure above, only users with a role in the Design work group can be granted read privileges to the Widget A class. If, however, John Smith maintained dual roles in both the Marketing and Design work groups, he could be granted read privileges on the Widget A class.

Example: controlling access to hierarchy definitions

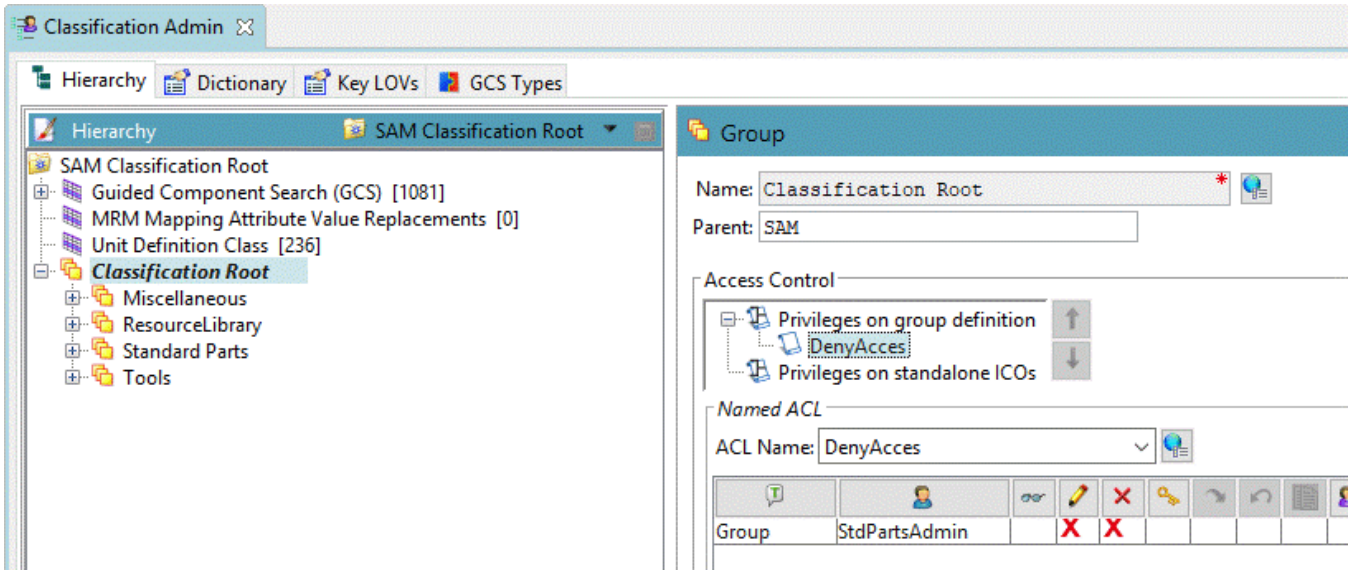
ABC Corporation also maintains a library of standard parts that are accessible to users throughout the organization. These parts are classified according to a hierarchy that includes storage classes for different types of parts.



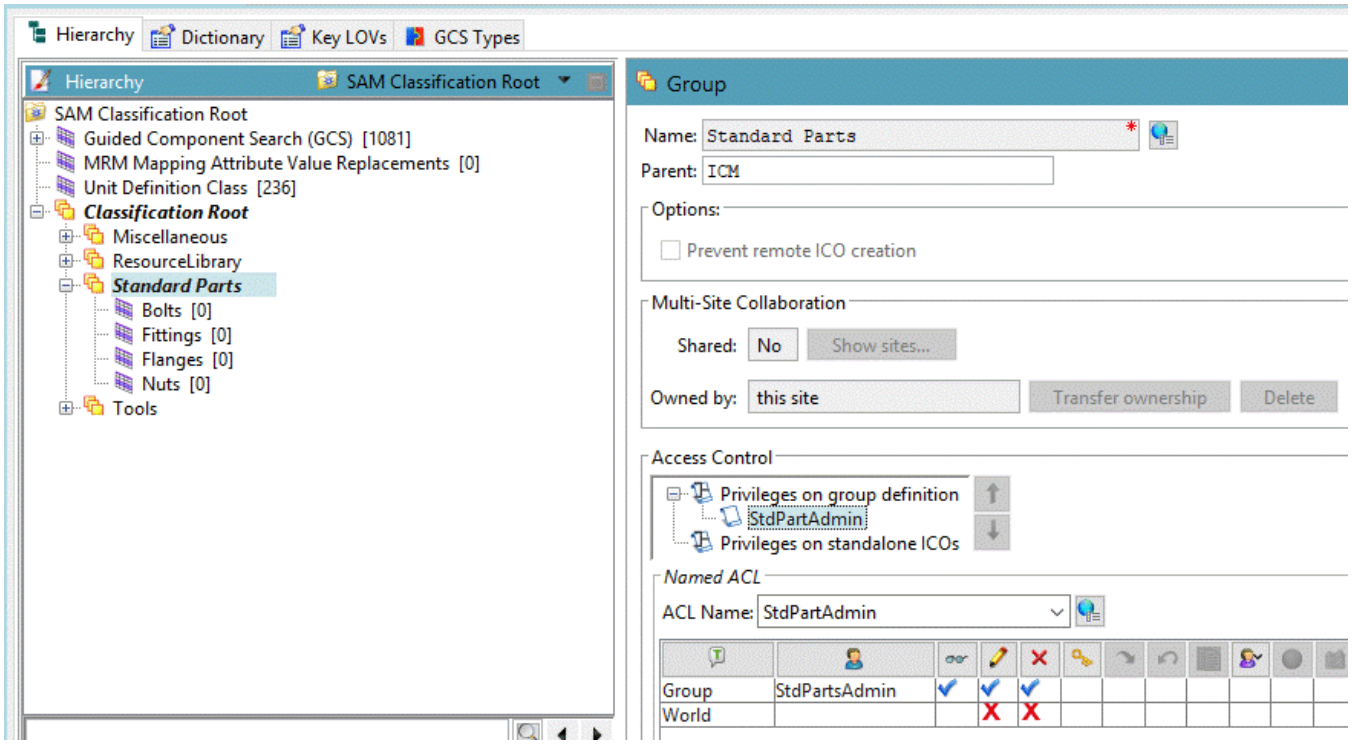
Only users in the **StdPartsAdmin** work group are allowed to perform maintenance tasks on this portion of the hierarchy (beginning with the Standard Parts group as the root). By default, access to classification administration objects is denied to non-DBA users. To grant a group of non-DBA users, the group must first be granted access to classification administration objects in the rule tree.



Now, the **StdPartsAdmin** group has access to the complete classification hierarchy. This must be restricted to the **StdPartsAdmin** work group only. Although there may be different approaches to assigning access rights depending on your business needs, this example begins by restricting access to the entire hierarchy in **Classification Admin**.



Subsequently, the **StdPartsAdmin** group is granted access to the **StdPartsAdmin** branch of the hierarchy only.



By granting write privileges to users in the **StdPartsAdmin** work group, the definitions of the **Standard Parts** classification group and all of its child classes are only modifiable by users who belong to the **StdPartsAdmin** work group.

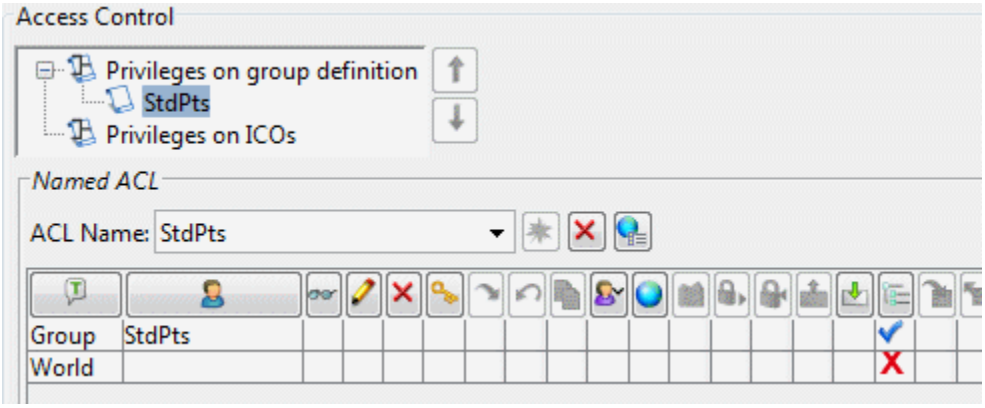
Note:

Write privileges can be granted on child classes of a write-protected group.

Example: controlling access to ICOs

Classifying the standard parts used to produce various models of widgets is a task assigned to the **StdPts** work group at ABC Corporation. While other work groups within the organization must be able to view the data associated with the Classification objects (ICOs), such as the physical attributes of the part or cost data, only the **StdPts** group is allowed to classify new parts or modify the attribute values of existing part classifications.

To restrict write access to standard part ICOs for all users except those in the **StdPts** work group, protections can be applied to the Standard Parts Classification group, as shown below.



By granting write privileges on the ICOs that are created under the Standard Parts group to users in the **StdPts** work group and denying those privileges to the world, the ability to create and modify ICOs is restricted to the **StdPts** group. However, all other users are able to view the ICOs and their data.

Note:

Creating an access rule under **Privileges on ICOs** controls access to stand-alone (nonclassifying) ICOs only.

Create a named access control list (ACL)

Note:

You must be in edit mode to create named ACLs.

1. Type a name for the new ACL in the **ACL Name** box.

2. Click the **Create** button located next to the **ACL Name** box.

Teamcenter creates the ACL. However, there are no entries associated with it.

3. Click the **Add New ACL** button .

   A blank line appears in the ACL table.

4. Double-click in a blank cell in the **Type of Accessor** column to display a list of predefined accessor types.

5. Select the accessor type that you want to use for this entry.

6. Double-click in a blank cell in the **ID of Accessor** column to display the **Select Accessor** dialog box. This dialog box contains a list of predefined roles corresponding to the type of accessor you selected in step 4.

7. Double-click the role that you want to apply to the accessor. You can also select the role in the dialog box and clicking **OK**.

   Teamcenter displays the role of the accessor you selected in the **ID of Accessor** column.

8. Define privileges for the accessor by double-clicking in the **Privilege** column and choosing one of the following options:

   - Grant privilege
   - Deny privilege

   Blank entries are also valid. Using blank entries enables rules to accomplish focused objectives by allowing objects and accessors to *fall through* rules that do not apply to them.

9. To add additional entries to the named ACL, repeat steps 1 through 7.

10. Click **Save** located to the upper right of the ACL table.

Modify access control list entries

The following task is performed in the **Access Control** pane in Classification Admin.

> Note:
>
> You must be in edit mode to modify named ACLs.

1. Choose the named ACL you want to change from the **Named ACL** list.

   Teamcenter displays the details of the ACL in the table.

2. Modify privileges by double-clicking the column corresponding to the privilege and choosing one of the following options:

   ✔ Grant privilege

   ✘ Deny privilege

   Blank entries are also valid. Using blank entries enables rules to accomplish focused objectives by allowing objects and accessors to *fall through* rules that do not apply to them.

3. Repeat steps 1 and 2 until all desired privileges have been granted or denied for this ACL.

4. Click the **Modify** button located at the bottom of the **Access Control** pane.

5. Click **Save** located to the upper right of the ACL table.

Delete access rules

The following task is performed in the **Access Control** pane in Classification Admin.

Access control rules can be removed from the rule tree and deleted from the database.

> Note:
>
> You must be in edit mode to delete access rules.

1. Click the **Hierarchy** tab. Teamcenter displays the **Hierarchy** pane.

2. Choose the group or class in the hierarchy tree that is affected by the rule.

   Teamcenter displays the definition pane for the group or class.

3. Click the **Edit** button on the toolbar to activate the definition pane.

4. If you are deleting rules relative to a class, click the **Access Control** tab to display the **Access Control** pane. When working with groups, the **Access Control** pane is displayed on the definition pane when you select the group.

   The **Access Control** pane displays the two AM rule tree roots related to Classification, along with the standard Teamcenter **Named ACL** dialog box.

5. Choose the rule in the tree that you want to delete.

6. Click the **Delete** button located at the bottom of the **Access Control** pane. The rule is removed from the tree.

7. Click **Save** on the toolbar to save the change and delete the rule from the database.

Searching the classification hierarchy

Classification hierarchy search overview

The Classification Admin search features, quick search and group and class search, save you time by enabling you to focus your search of the classification hierarchy using familiar criteria, such as name, class ID, and attribute name or ID.

The quick search method is a simple search of the hierarchy using the **Name** property as the search criteria. Results are displayed in the hierarchy tree, one entry at a time. You can navigate through the results using the left and right arrow buttons.

The group and class search feature is an advanced search of the hierarchy using any of the following properties as search criteria:

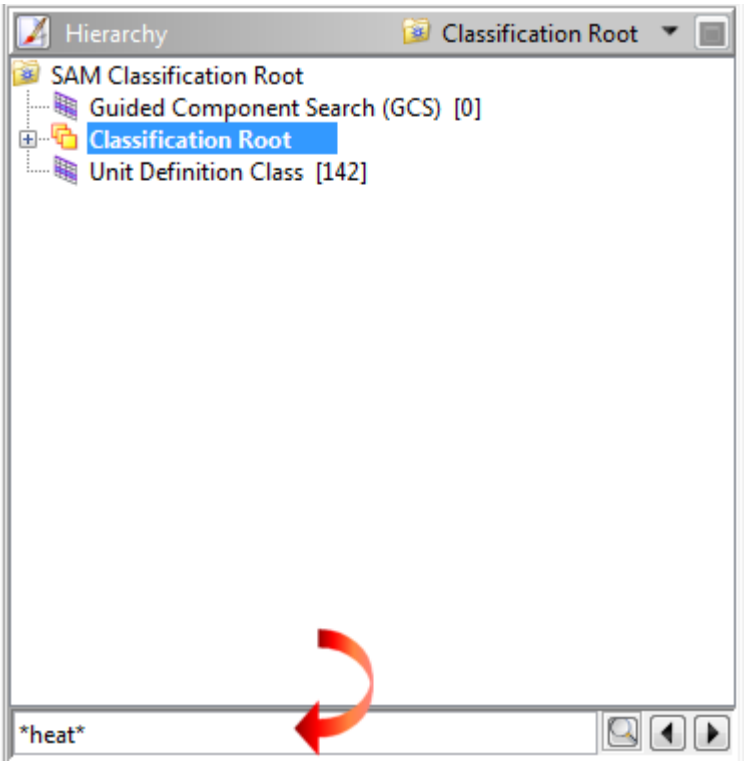
Class ID
Name
Alias Names
Name & Alias Names
Attribute ID
Attribute name
User data 1
User data 2

Results are listed in the **Search Class** dialog box. Choose a result from the list to expand the tree and indicate the class and/or group.

You can specify that the search does not return ICOs for which the user does not have read access by setting the **ICS_search_filter_by_read_access** preference.

Search using the quick search feature

1. In the search box located beneath the hierarchy tree display, type text corresponding to the name of the group or class that you want to locate.



You can also search by ID by entering **id=***xxx* in the text box, where *xxx* is the class ID.

The search text can either be the exact name or ID of the object you are looking for or you can use character strings combined with wildcard characters. The default multiple-character wildcard is an asterisk (*****) and the default single-character wildcard is a dot (**.**). Using a wildcard is available for string attributes only.

2. Press Enter to start the search.

   The hierarchy tree expands to display the first object in the hierarchy that matches the search criteria. The path of the group or class is indicated in bold text (see the figure in *Search using the Search Class dialog box*). If multiple objects are found, the arrow buttons at the bottom of the hierarchy tree are enabled.

3. Click the left-arrow and right-arrow buttons to display the matching objects, one at a time. The right-arrow button moves down the hierarchy tree; the left-arrow key moves up the hierarchy tree.

   > Note:
   >
   > If you prefer to view a list of the results, you can display the **Search Class** dialog box (see the figure in *Search using the Search Class dialog box*) by clicking the **Magnifying Glass** button located below the hierarchy tree.

Search using the Search Class dialog box

1. Click the **Class Search** button located below the hierarchy tree. Teamcenter displays the **Search Class** dialog box. To move the dialog box, double-click the title bar and drag the dialog box to another location on your desktop.

2. Define the search criteria by performing the following steps:

   a. Choose a property from the list at the upper left corner of the dialog box. The available properties are:

      Class ID
      Name
      Alias Names
      Name & Alias Names
      Attribute ID
      Attribute Name

      > Note:
      >
      > You can use the **Name** and **Class ID** properties to search for groups or classes. When searching by attribute, the results include the class in which the attribute is defined and any subclasses in which the attribute is used. Classes that inherit the attribute are not included in the results.

   b. Enter search text corresponding to the selected property.

      The search text can either be the exact name or ID you are looking for or you can use character strings combined with wildcard characters. The default multiple-character wildcard is an asterisk (*****) and the default single-character wildcard is a dot (**.**). Using a wildcard is available for string attributes only.

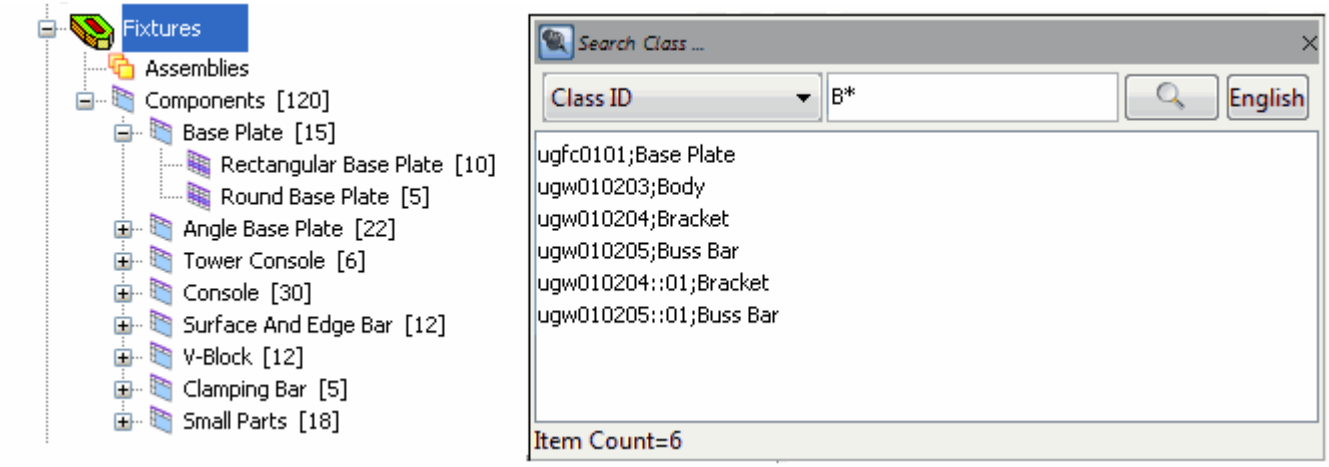
      > Note:
      >
      > The search text box is case sensitive.

3. To start the search, either click the **Magnifying Glass** button located in the upper-right corner of the dialog box or press Enter.

   Teamcenter displays the results of the search in the message area of the dialog box, sorted in the same order as the hierarchy tree display.

4. To display an object in the tree, double-click the entry in the results list.

   The hierarchy tree expands to display the selected group or class. The path to the object is indicated in bold text.



Viewing files associated with to groups, classes, and views

Adding images to groups, classes, and views

Adding images to groups, classes, and views provides visual points of reference within the classification hierarchy. These images are displayed for the Classification user in the class viewer at the top right of the **Properties** pane when the group or class is selected in the hierarchy tree. Images help identify the contents of a group or class and should be generic enough in nature to adequately represent the variations in the contents of the group or class. You can add multiple images to a class which then appear in the viewer when you click the tabs at the side of the viewer.

The following preferences control the behavior of class images:

- **ICS_presented_class_documents** contains a list of those named references you want to have available for the viewer to display. The system searches through the **ICSClassFiles** dataset of the class for these named references and presents a tab for each one.

- **ICS_default_class_document** controls which of the available documents is displayed by default when you view a class.

By default, you can present 10 images: **ICS-ClassImage**, **ICS-ClassImage1**...**ICS-ClassImage9**. If you want to alter this number, or the names of these images, you must modify the **ICSClassFiles** dataset in Business Modeler IDE.

Add an image to a group, class, or view

1. Select the group, class, or view to which you want to add an image and choose **Edit**.

2. Right-click the node in the hierarchy tree and choose **Add Image**.

   Teamcenter displays the **Select Image File** dialog box.

3. Locate and select an appropriate image file to be associated with the group or class.

4. Choose the appropriate reference for this file from the **Reference** list. If you want the selected image to be the default image, choose **ICS-ClassImage**.

5. Click **Import**.

   Teamcenter displays the image in the class viewer.

   > Note:
   >
   > If the image does not immediately load in the class viewer, reselect the node in the hierarchy. This action initiates the image loading process.

6. (Optional) Repeat the preceding steps to add multiple images to a class.

Remove an image from a group, class, or view

1. Select the group, class, or view from which you want to remove an image and click **Edit**.

2. Do one of the following:

   - To remove an image from a group, click  at the bottom of the **Group** pane.
   - To remove an image from a class, click  at the bottom of the **Class Details** pane.
   - To remove an image from a view, click  at the bottom of the **View** pane.

3. Click **Save**  .

View GIF images

1. Open the **com\teamcenter\rac\classification\common\common_user.properties** file in a text editor.

2. Modify the file to contain *one* of these lines:

   ```
   Imager.VIEWPANEL=com.teamcenter.rac.util.viewer.TwoDViewer
   ```

   -or-

   ```
   Imager.VIEWPANEL=com.teamcenter.rac.classification.common.G4MTwoDViewer
   ```

TwoDViewer displays the markup tree and the toolbar. With **G4MTwoDViewer**, no markup tree or toolbar is visible in the viewer, but you can use the usual context menu commands for manipulation, such as zoom, pan, or fit.

Viewing files associated with an ICO in the viewer

To view documents associated with ICOs in the viewer, you must ensure that the file type is added to the following preferences:

- **ICS_presented_class_documents** contains a list of those named references you want to have available for the viewer to display. The system searches through the **ICSClassFiles** dataset of the class for these named references and presents a tab for each one.
- **ICS_default_class_document** controls which of the available documents is displayed by default when you view a class.

By default, you can present 10 images: **ICS-ClassImage**, **ICS-ClassImage1**...**ICS-ClassImage9**. If you want to alter this number, or the names of these images, you must modify the **ICSClassFiles** dataset in Business Modeler IDE.

If you use 32-bit browsers, you see each image type as a tab to the right of the image. If you use 64-bit browsers, a **Launch** button is displayed in the viewer.

Generating graphics for classification objects

Graphics generation overview

There are three ways in which Teamcenter can automatically create part files and JT graphics for an ICO:

- Based on part family templates

  Part family templates are used in NX to define a set or *family* of parts that share similar form, fit, and function but differ based on parameter values (for example, length, width, or diameter) that typically control the physical characteristics of the part (or tool). The part families are created with the help of a Microsoft Excel file that holds a list of all *part family members*.

- Based on template parts

  Any NX part can be used as a template part.

- Using Tcl scripts

  You can generate ICOs based on Tcl macro files. This is generally used with legacy Genius4000 data.

> Note:
>
> When you create graphics for a tool component for instance, either with part family template or a template part, the geometry and CSYS are generated in NX. The graphics and CSYS are also displayed in Teamcenter. If the CSYS does not display, unload and then reload the component.

Both part family templates and template parts contain expressions that describe a part parametrically. For example, **L1** represents the length of a drill. If you change the value of **L1**, you can quickly create many drills (*part family members* or *member parts*) of different lengths. Although the behavior in Teamcenter when using part family templates or template parts is very similar, the mechanics of how graphics are created for the members in the background varies.

> Note:
>
> Revisioning is supported with the template part method only.

An administrator must perform the following steps to configure graphics building for ICOs:

1. Associate the part family template or template part with a specific class.

2. Map the template expressions to class attributes.

A Classification user creates a new ICO by entering attribute values. When the user starts the process to create graphics, Teamcenter starts the graphics builder executable that communicates with NX in the background and generates a new part family member or member part using the new attribute values. The graphics builder executable also creates a 3D model and, optionally, a JT file. These are stored in the database, and the JT file is displayed in Classification.

Configure the graphics builder

Classification Admin uses a separate graphics builder program that communicates with the Teamcenter server to generate graphics. The graphics builder uses NX libraries; therefore, NX must be installed on the same computer as Teamcenter.

- You can configure the graphics builder on a Windows or Linux server platform only.

- If you are using the graphics builder with Teamcenter Security Services single sign-on (SSO) capability, you must configure the Teamcenter integration to use this.

  For more information, see the *Single Sign-On* documentation in the NX Help (*Introduction to Teamcenter Integration for NX→Advanced management of data*).

1. Start Teamcenter Environment Manager.

2. Select **Configuration Manager** and click **Next**.

3. Select **Perform maintenance on an existing configuration** and click **Next** until you reach the **Feature Maintenance** pane.

4. Select **NX Graphics Builder** under **NX Part Family Classification Integration** and click **Next**.

> Note:
>
> This option is available only if you have previously installed **NX Rich Client Integration**.

5. Enter the following information:

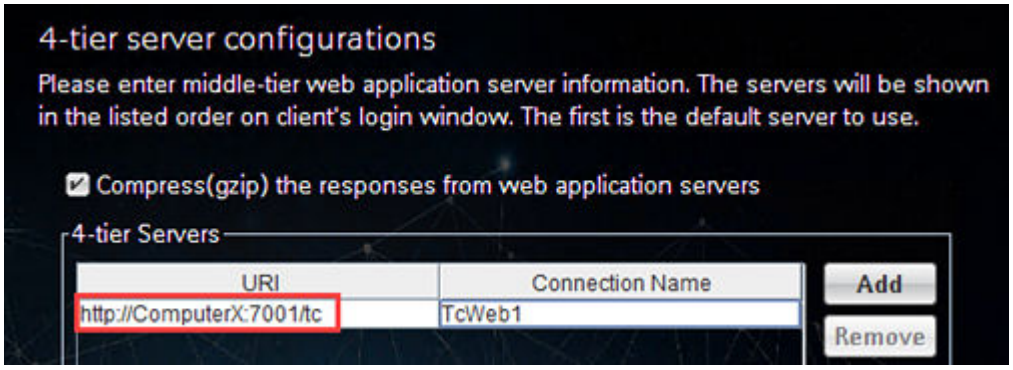
Value	Description
NX Install Location	Specify the path to the NX installation.
Graphics Builder Base Port	Specify the port used for the HTTP graphics builder server. Any free port is valid.
Number of Ports	Specifies the maximum number of graphics builder instances to be started.
Server Manager URL	Specify the Teamcenter web server URL. This is the Teamcenter 4-tier URL.

The Teamcenter web server URL has the following syntax:

http://*my_hostname***:***my_port***/***my_path*

Where:

my_hostname	Specifies the computer name of the machine hosting the Teamcenter server. To find this, open a command prompt on the machine hosting the Teamcenter server and type **hostname**. The response is used as *my_hostname*, for example, **ComputerX**.
my_port/*my_path*	Specifies the port on the host machine used to communicate with the Teamcenter server. You can find this entry as follows: a. Open TEM and proceed through the wizard until you reach the Feature Maintenance page. b. Select **Modify 4-tier server settings** from the **Client Communication System** section.



In this example, the Teamcenter web server URI is:

http://ComputerX:7001/tc

The example displays *my_port* as **7001** and *my_path* as **tc**. Your entry in TEM may be different.

This step does the following:

- Writes the path to the NX installation into the **start_nx_graphicsbuilder.bat** file as the value for the **UGII_BASE_DIR** variable. This file is stored in the *Teamcenter_root***bin\nx_graph** directory.

- Writes the following environment variables:

  UGII_UGMGR_HTTP_URL

  Specifies the Teamcenter web server and the application name of the web server in four-tier environment.

  TC_GB_BASE_PORT

  Sets the variable to the base port for the graphics builder server. For example, if you enter 7007, the first graphics builder server starts on this port and subsequent servers start on ports 7008, 7009, and so on up to the number of ports specified by **TC_GB_NUMBER_OF_PORT**.

  TC_GB_NUMBER_OF_PORT

  Sets the variable to the number of graphics builder instances. For example, if this variable is set to **10**, a maximum of 10 graphics builder users is supported.

Note:

This configuration works only if Teamcenter and NX are installed in the same location on both Teamcenter server and client.

Understanding part family templates

You can create part family templates that define geometry and certain properties of the geometry as variable properties, for example, lengths and angles, in NX. When you assign values to the properties

in the part family template, you create part family members. You can create members in NX or in Teamcenter Classification by creating an ICO and propagating its values to the part family template.

For more information about how to create part family templates, see the *NX Help Library*.

> Caution:
>
> For graphic generation to work correctly, you must choose **Importable Part Family Template** in the **Part Families** dialog box in NX. This ensures that the two columns, **DB_PART_NO**, and **OS_PART_NAME** appear in the part family spreadsheet.

An administrator attaches part family templates to class definitions in Classification Admin and maps the attributes of the class to the variable properties of the part family template. When you assign values to an instance of a class to which a part family template is attached in the Classification application, you can create a part family member by clicking the **Create/Update graphic from ICO** button after saving the ICO.

> Note:
>
> Multiple templates can be attached to a single class. In this situation, when you create a part family member for an ICO, you must decide which template to apply. When you create the part family member, a relationship is established between the ICO and the template. Once this relationship is established, you cannot create a part family member for this ICO using a different template.

Understanding template parts

Template parts are parametric NX parts whose expressions are mapped with class attributes. Graphics creation based on template parts is very similar to that based on part family templates with the following advantages:

- The template and the instances have their own status and revisions. This allows you to revise template and members individually.
- You can reclassify a component (move it from one class to another) and re-create the graphics based on the template part in the new class.
- You can manually modify the part file of a specific instance.
- No write access to the template is required.
- You can easily update or replace an existing template part file (for example, to fix incorrect geometry in the template).

To use template parts, you must perform all the setup and configuration steps required for using part family templates.

When working with template parts, there are two scenarios possible when you update a member part. You can refresh member parts by updating with new classification attribute values, or you can re-create the geometry from the template part.

Associating part family templates or template parts with class definitions

Associating part family templates or template parts with class definitions and mapping the class attributes to expressions in the part family template or in the template part allows you to associate CAD graphics files with Classification instances. This process consists of the following steps:

- Attaching the part family template or template part to the class.
- Mapping the part family attributes to the class attributes.

In addition to associating a part family template or template part to a particular class, you can choose to have the system look upwards in the hierarchy to find part templates that are associated to parent classes by clicking **Use hierarchy**. If these parent classes have stipulated that the template can be used in child classes, you are offered a selection of all available templates when you create ICOs for the class. These templates come from both from the current class and from any parent classes whose templates are available for child classes.

Attach a part family template or template part to a class

You can attach either a part family template or a template part to a class. When doing so, you can choose to attach the template item or the item revision.

- When you attach the item, the latest revision is used for graphics creation.
- When you attach the item revision, you must select the desired revision with which to create graphics.

Choosing an item rather than an item revision allows you to revise the item in response to changes to the part. You can make modifications to the part family template or template part, which, in most cases, leads to the creation of a new item revision.

1. Do one of the following:
    - Use the clipboard in the My Teamcenter application.
        a. In the My Teamcenter application, choose an item or item revision containing the part family template or template part.
        b. Copy the item to the clipboard.
        c. Switch to Classification Admin.
        d. Choose the class to which the template will be attached.

Teamcenter displays the class details in the pane on the right.

e. Click the **Edit** button on the Classification Admin toolbar.

f. Click the **Graphics Builder** tab.

Teamcenter displays the attributes of the selected class.

g. Click the **Paste** button next to the **Template** box.

Teamcenter pastes the template into the **Available Templates** list; however, the template expressions are not displayed until the list is refreshed.

h. Click the **Refresh List** button.

The system displays the template expressions in the unmapped attributes list.

Because this is the first point at which Teamcenter starts the graphics builder server in the background, this step may take some time.

- Select an item or item revision from your home folder.

a. Select the class to which the template will be attached.

The system displays the class details in the pane on the right.

b. Click the **Edit** button on the Classification Admin toolbar.

c. Click the **Graphics Builder** tab.

The system displays the attributes of the selected class.

d. Click the **Select from Home Folder** button beside the **Template** box.

The system displays a navigation tree containing your home folder.

e. Navigate to the item or item revision containing the desired part family template and click **Attach**.

Teamcenter pastes the template into the **Available Templates** list; however, the template expressions are not displayed until the list is refreshed.

f. Click the **Refresh List** button.

The system displays the template expressions in the unmapped attributes list.

Because this is the first point at which Teamcenter starts the graphics builder server in the background, this step may take some time.

- Select a part from the operating system.

  a. Select the class to which the template will be attached.

     The system displays the class details in the pane on the right.

  b. Click the **Edit** button on the Classification Admin toolbar.

  c. Click the **Graphics Builder** tab.

     Click the **Graphics Builder** tab.

     Click the **Graphics Builder** tab.

  d. Click the **Import a part family template from the OS** button.

     Teamcenter displays your operating system.

  e. Navigate to the part file.

     When you import a part file from the operating system, Teamcenter automatically creates an item and attaches the part to the item revision.

     The unit system in the **Part Family** tab is displayed as **unknown**.

  f. Click the **Refresh unit system information from the template** button.

  g. Teamcenter displays the unit system of the part file and displays the template expressions in the unmapped attributes list.

     Because this is the first point at which Teamcenter starts the graphics builder server in the background, this step may take some time.

2. (Optional) Do any of the following:

   - Select **Use as default template**.
   - Select **Use for child classes**.
   - Type a value for **Type name**.

Mapping part family template attributes or template part expressions to class attributes

Template expressions can be mapped to class attributes either automatically or selectively. Alternatively, you can map an attribute simply by dragging it from the unmapped attributes list to the appropriate cell of the **Template Columns** in the mapping table.

Mappings are enabled by default. They can be disabled. This allows you to map attributes for large part family templates or classes over multiple sessions, restricting availability until the mapping is complete. To disable the mapping, select the **Enabled** option on the **Graphics Builder** tab.

> Note:
>
> Modifying part family templates may invalidate the mapping between part family template and class definition. If this occurs, you can modify and validate the mapping and update the ICOs after the item revision is created.

When working with template parts, you may choose whether to load all or only a subset of expressions into Teamcenter.

- If some expressions are marked as **TC** in the **Comments** column of the expressions in NX, only these expressions are displayed in Teamcenter.
- If the **Comments** column remains empty, all expressions are loaded into Teamcenter.

Table 8-3. Expressions in NX

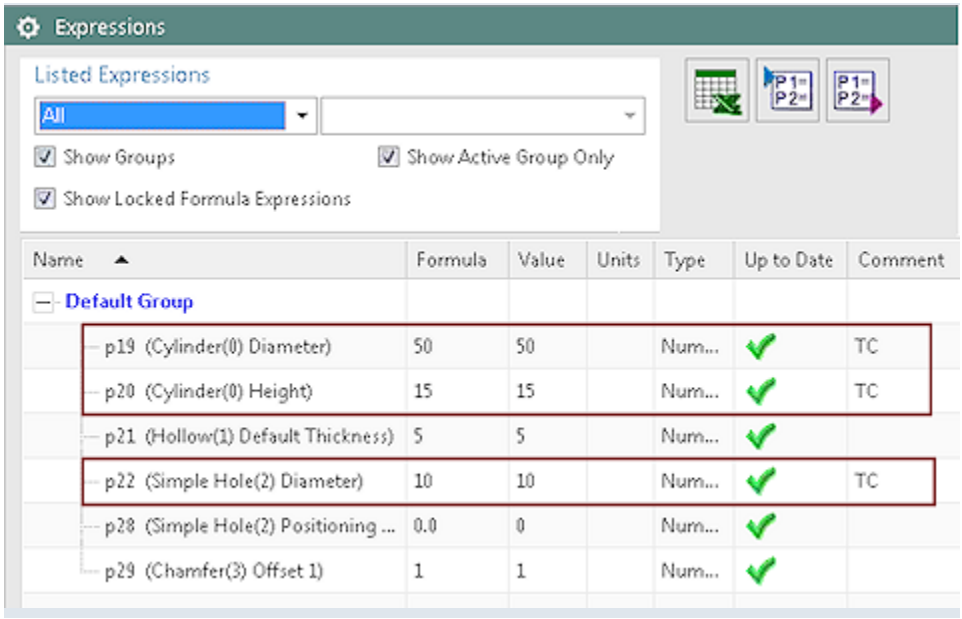


Table 8-4. Expressions loaded into Teamcenter

Attributes	Template Columns
-7504 Diameter **p19**	
-7503 Height **p20**	
-7701 Diameter Hole **p22**	

p19 EXPRESSION
p20 EXPRESSION
p22 EXPRESSION

Automatically map attributes

> Note:
>
> To use the automatic mapping feature, the name of the expression in the template must match the value in the annotation box of the attribute definition.

1. Select one or more expressions in the unmapped attributes list (on the **Graphics Builder** tab), and click <<*.

   The system displays the part family expressions and their corresponding class attributes in the **Mapping** table.

2. Click **Save** on the toolbar.

   Teamcenter saves the mappings in the database.

3. (Optional) Validate the mapping by clicking the **Validate mapping** button. The validation feature provides information about expressions that no longer exist in the template.

Selectively map attributes

1. Select an attribute from the class attributes column of the mapping table.

2. Select the corresponding attribute from the unmapped attributes list.

3. Click the **Map Selected Attribute** button <.

   Teamcenter displays the part family attribute with its corresponding class attribute in the mapping table.

4. Repeat the preceding step until all the attributes are mapped.

5. Click **Save** on the toolbar.

   Teamcenter saves the mappings in the database.

6. (Optional) Validate the mapping by clicking the **Validate** button. The validation feature provides information about expressions that no longer exist in the template.

Delete a template from a class

When you remove a template from a class, you remove only the association between the template and the class. The template still exists as an item in the database and can subsequently be associated with a different class.

1. Click **Edit** in the class containing the undesired template.

2. Select the template from the **Template** list.

3. Click the **Delete** button.

   The name of the template disappears from the list.

   You cannot change the association of a part family template after ICOs are created. If you are working with part family templates and already created ICOs for the part family members in a previous class, and you associate that part family with a new class, when you create ICOs for the new class, the system informs you that ICOs already exist for the given part family members.

Create classification instances (ICOs) for part family members

If a part family template includes member information in the attached spreadsheet, you can create classification instances for these members. This is an administrative task that is normally executed only one time after attaching the part family template to a specific Classification class. Do not confuse this procedure with creating ICOs for graphics.

> Note:
>
> When created, ICOs are automatically named. This name is made up of **member name/template revision**, for example, for part family member 000025 and revision A, the ICO name would be 000025/A. ICOs cannot be created if an ICO with the same name as the part family member already exists in the database.

1. Select the class in the hierarchy tree.

   The system displays the details of the class in the pane on the right.

2. Click the **Graphics Builder** tab.

   The system displays the part family template to class definition mapping information.

3. (Optional) Validate the mapping by clicking the **Validate** button.

The validation feature checks whether the column names in the part family template still match those mapped to attribute names in the mapping table.

4. Click the **Create ICOs** button  .

   The system displays a message confirming that ICOs have been created.

   If you are reusing a template that had previously belonged to another class, you can do either of the following:

   - Move the ICOs from the original class to the current class (by dragging them from one class to another in the Classification application). ICOs corresponding to these part family members then exist in the current class, but are not associated to the part family template. You can choose to reassociate these moved ICOs with the part family template by clicking **Connect existing ICOs**.

   - Create new ICOs in the new class causing the items to be classified in both the old and the new class.

After creating ICOs, you can switch to Classification and navigate to the class in which you were working. This class now contains new ICOs—one for each part family member from the template. The object ID of these ICOs is comprised of the name that you entered as **Part_Name** or **DB_PART_NAME** in the part family template appended with **/template revision**.

Create graphics using legacy Genius4000 Tcl scripts

If you have migrated from Genius4000, you may have created Tcl scripts that you used to create graphics from ICOs. You can use these to create graphics in Teamcenter. The scripts use the attribute values of the ICOs as input and produce graphic output, such as an NX part file or JT file.

> Note:
>
> Siemens Digital Industries Software recommends creating graphics using the part family template or template part method. The Tcl evaluation method is generally used for dealing with legacy data.

1. Set up the Teamcenter preferences as follows.

   a. Add the following values to the **NXGraphicsBuilder** preference:

      - **ScriptEvaluationPath:***the_path*

        the_path is the path to the directory where the Tcl scripts are stored.

      - **evaluateScript:yes**

      - **evaluateScriptWhenNotSet**

This overwrites the **not set** values for Tcl script evaluation in your classification classes. If you set this preference, Tcl scripts are evaluated even if **Tcl Script Evaluation** is set to **not set** in the individual classes.

2. If the scripts are not already in this folder, copy them there. You must maintain the directory structure you used in Genius4000.

3. (Optional) Set the default value for script evaluation when creating new classes. To do this, specify the following preference:

   evaluateScript:*your_setting*

   your_setting can be **yes**, **no**, or **not set**.

4. Copy the following files from *TC_ROOT***sample\in-CLASS\Genius4000Tcl** to the script evaluation path specified in step 1:

   unclib.ucl
   uncuflib.ucl
   ics_graphicsbuilder.tcl

   Note:

   You must install the **sample** directory using Teamcenter Environment Manager.

   When evaluating Tcl scripts, the system expects to find a file with the name **ics_graphicsbuilder.tcl** in the folder specified by the **ScriptEvaluationPath** preference. The system sources the file and then calls the procedure. The script programmer designates what the file and procedure accomplish. To see an example, look at the following files found in *TC_ROOT***sample\in-CLASS\Tcl**:

   ics_graphicsbuilder.tcl
   myclass.tcl

   This code creates/updates a part file containing a block for a classification instance in class **myclass** using the values of attribute IDs **–200000**, **–300000**, **–400000**, representing height, width, and length.

   The attribute values are set as Tcl array variables **GRAPHICSBUILDER_PARAMS** (*attribute-id*). The system assumes that the **createGraphics** procedure creates a part file with a specific name. The script can ask for this name by calling the **UGB_ask_partfile_name** procedure. If this part file exists, the classification instance for which this script is being evaluated classifies this object.

   You can use all the functions for Tcl script evaluation that were available to you in Genius4000.

Create graphics when template parts or part family templates have a multifield key identifier

To create graphics when Teamcenter is set up to use multifield key unique identifiers, NX must also be configured to use multifield keys.

To configure NX, set the following preferences in Teamcenter.

- **TC_MFK_DEFAULT_DOMAIN**

  This specifies the Teamcenter default domain. This is set to **Item** by default to use the standard default domain.

  > Note:
  >
  > To change the default domain to a value other than **Item**, contact your Siemens Digital Industries Software representative for assistance in setting up NX.

- **TC_NX_Supports_MFK**

  If set to **True**, NX supports items in all Teamcenter domains. If this is not set, then NX can only access those item types that belong to the default domain. This is not set by default.

For more information, see the NX documentation.

The following points pertain to part family templates.

In Classification Admin:

- When attaching an multifield key part family template to a classification class, all multifield key attributes from this template must be mapped to classification attributes.

In Classification:

- When creating graphics, all classification attributes mapped to (mandatory) multifield key properties must contain a value so that NX is able to successfully create the member.

- When updating graphics, all classification attributes mapped to multifield key properties are ignored (as the information is already available from the classified object).

Setting template or script priority

You can attach a part family template to any class in the classification hierarchy. Also, at any point in the hierarchy, you can specify that graphics for a particular class should be created using a Tcl script. Classification Admin offers you various options to dictate which graphics creation method should be available for use by a specific class.

Use the **Show graphics information** button to see which method will be used to create graphics for the selected class. An information window opens showing you the status of each part family template. It tells you why a template is or is not available for graphics creation.

The following table provides you with an overview of the options available to influence part family template or script priority:

If you want to	Select
Search upward in the hierarchy for templates attached to parent classes that are available for graphics creation.	**Use hierarchy**
Have the current template appear first in the list of available templates when you create graphics in other applications.	**Use as default template**
Enable inheritance of the part family template for all child classes.	**Use for child classes** If the child classes have activated **Use hierarchy**, you see the template from the parent class in the list of available templates when creating ICOs for these child classes in Classification.
Override the use of a template higher up in the hierarchy.	**Type name** By giving different templates in the hierarchy the same type name, you can control which template is available when you create graphics.

The following example shows the interrelationship between these options:

```
Class A + PFT 1 & PFT 2
  :
  :…  Class B has no template assigned
        :
        :…  Class C + PFT 3
  :
  :…  Class D + Tcl Script evaluation=yes
        :
        :…  Class E has no template and Tcl script evaluation=not set
              :
              :…  Class F + PFT 4 and Tcl script evaluation=no
```

In this example:

- Class A has two part family templates attached—PFT 1 and PFT 2.

- Class B is a child class of Class A and has no part family template. Tcl script evaluation is not set.
- Class C is a child class of Class B and has one part family template attached—PFT 3. Tcl script evaluation is not set.
- Class D is a child class of Class A. It has no part family template assigned. Tcl script evaluation is set to **yes**.
- Class E is a child class of Class D and has no part family template. Tcl script evaluation is not set.
- Class F is a child class of Class E and has one part family template attached—PFT 4. Tcl script evaluation is set to **no**.

The following tables show various option combinations and their effect on which template is available to create graphics:

Classes	Tcl script evaluation	Use hierarchy	Use in child classes	Type name	Default template
Class A – PFT 1	Not set	✓	✓	Master	
Class A – PFT 2	Not set				✓
Class B – no pft/script	Not set				
Class C – PFT 3	Not set	✓		Master	
Class D – Tcl Script	Yes		✓		
Class E – no pft/script	Yes	✓			
Class F – PFT 4	No	✓			

If you switch to Classification and generate graphics for ICOs in each class, the following methods for graphic generation are available:

Class	Available templates	Reason
Class A	PFT 2 PFT 1	You have the choice between PFT 2 and PFT 1 as both are assigned to this class. PFT 2 appears first in the list as it is the default template.
Class B	PFT 1	As this class has no part family template of its own, the system looks higher up in the hierarchy for a template. Template PFT 1 has **Use in child classes** active, so it is available for graphic generation.
Class C	PFT 3	Although PFT 1 has **Use in child classes** active, it has a type name **master** specified. Because PFT 3 also has

Class	Available templates	Reason
		a type name **master**, PFT 3 overrides PFT 1 and is the only template available for graphic generation.
Class D	Tcl Script Evaluation	Class D has a Tcl script attached and is therefore able to use this script for graphic creation. Because it does not have **Use hierarchy** active, it does not look any further in the hierarchy for another method of graphics creation.
Class E	Tcl Script Evaluation	Class E has neither a Tcl script nor part family template attached. It does, however, have **Use hierarchy** active and therefore looks upward in the hierarchy. When it reaches Class D, it sees that there is a Tcl script attached that can be used for child classes. Class D does not have **Use hierarchy** set, so the system stops looking upward in the hierarchy.
Class F	PFT 4	Class F has Tcl script evaluation set to **no**. It has PFT 4 attached, so it can use this to create graphics. It also has **Use hierarchy** activated, so it looks upward. Class E has no part family template attached, but has **Use hierarchy** active, so the system continues to look upward. When it reaches Class D, **Use hierarchy** is not set, so it stops looking.

Sharing template data using Multi-Site Collaboration

If you are using Multi-Site Collaboration to share classification information that includes part family templates or template parts, there are several points to consider:

For part family templates:

- After you attach a part family template item or item revision to a Classification class and want to share this item/revision with a remote site, you must share the class with the remote site using the Classification mechanism. If you attempt to replicate the item using the Multi-Site ODS, the class information does not get shared with the item.

- You can only create part family member ICOs at the master site. These are then shared with the local sites if the **ICS_share_related** preference is set to **partFamilyTemplates**.

For template parts:

Add **templateParts** to the value of the **ICS_share_related** preference.

Sharing classification hierarchy data

Overview of sharing with Multi-Site Collaboration

Classification hierarchy data, class, view, attribute, and key-LOV definitions can be shared with one or more remote sites, using basic Multi-Site Collaboration concepts.

> Note:
>
> This feature is used to share hierarchy data, not Classification instances. Classification instances are shared as attachments to item and item revision workspace objects, via Multi-Site Collaboration.

To share Classification data, Multi-Site Collaboration must be installed at each site, the list of shared sites must be defined using the **ICS_synchronize_sites** preference, and the **IDSM** service must be running at both sites. If you want to share data automatically, the **ICS_synchronize_automatically** preference at the local site must be set to **1** and the **subscriptionmgrd** process must be running. For example, to share data from site A to site B and from site B to site A, the **subscriptionmgrd** process must be running at both sites.

> Note:
>
> - The Classification Multi-Site Collaboration feature does not use Object Directory Services (ODS).
> - Errors due to the transfer of classification objects are listed in the Subscription Manager log file.
> - Other error messages are listed in the **IDSM** log file.
> - Before sharing, you cannot have any duplicate IDs at any of the sites.
> - If you use custom logic, you must synchronize this manually.
> - Do not share any classes or attributes that are delivered with the software, such as the unit definition classes. If you do, you cannot share any classes that you create that contain any of these attributes.
> - If you change a site name, you must run the **smlutility** utility with the **–update_shared_sites** option.

Multi-Site Collaboration sites

A Multi-Site Collaboration site is considered to be a single Teamcenter database and its users, rather than a physical location such as a plant or engineering facility.

Object ownership

Objects can be shared with multiple sites, but in order to maintain data integrity, ownership of an object is restricted to a single site. When data is shared, a read-only replica is created for each site on the shared site list. Classification uses the concept of a *Classification master site* to track ownership of the hierarchy. A site is the master site for a class or group if all descendants are owned at this site. Therefore, different parts of the hierarchy can be mastered by different sites.

You cannot add children to parents that are not locally owned. Therefore, you must transfer the ownership to the site that is allowed to add children. By tracking ownership, the master site tracks where classes are added locally. You can use Access Manager to protect the transfer of the mastership.

Before you add or remove attributes from a shared class, you must ensure that the intended master site has access to and owns all children of this node. You must manually confirm that a site has been set up as a master site by setting the **ICS_classification_master_site** preference to **true**.

> Warning:
>
> You cannot perform a change to the class hierarchy that requires a change to its ICOs (for example, removing attributes) if the class is already shared.

If the classification hierarchy is exchanged between sites using XML or SML files instead of Teamcenter Multi-Site Collaboration, you must apply the same change to all sites.

If a site does not support Classification, you can specify a remote site that can classify objects for that site using the **ICS_classification_site_for** preference.

Object dependencies

Dependencies between classification hierarchy objects require that the associated objects also be shared. For example, to share a view definition, you must also share the class definition and its parent classes, the dictionary objects (attributes), and the key-LOVs associated with the view. You are asked if you want to share the class definitions when selecting sites with which to share the view. Dictionary objects and key-LOVs are shared automatically if this is necessary to share a view/class.

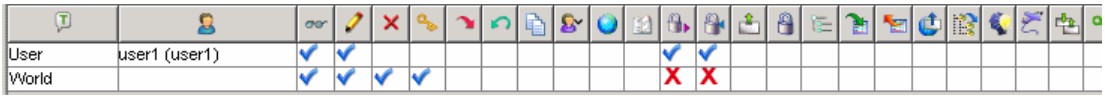
The following table shows the dependencies between Classification objects.

Classification object	Dependent objects	Are dependent objects shared automatically?
Key-LOV	None.	Not applicable.
Dictionary object (attribute)	If the dictionary object format is defined as key-LOV, then the key-LOV is also shared.	Yes.

Classification object	Dependent objects	Are dependent objects shared automatically?
Group definition	All group definitions that are parents of the selected group definition. The **SAM** and **ICM** groups exist at all sites, and therefore are not shared.	No.
Class definition	All class and group definitions that are parents to the selected class definition. The **SAM** and **ICM** groups exist at all sites, and therefore are not shared.	No, class and group definitions are not automatically shared.
	All dictionary objects and key-LOVs, if applicable.	Yes, dictionary and key-LOV objects are automatically shared.
View definition	Class definitions for which the view is defined. For details on class objects that must be shared, see Class definition in this table.	No.

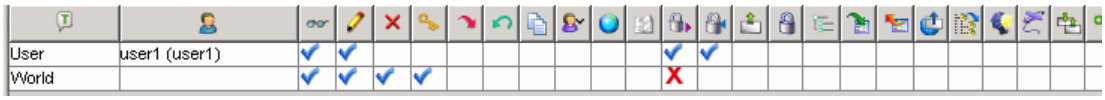
Understanding access rights

Assume a class hierarchy is already shared from site 1 to site 2, and you set the access rights as follows.



When exporting objects, at any exporting site, the importing sites are first evaluated to see that they can import from this site. Because all potential sites are available in this organization, when the **Import** privilege is revoked on **World**, it evaluates to site 2 not being privileged to import.

Instead, to allow only the selected users to import or export, use the following access control list.



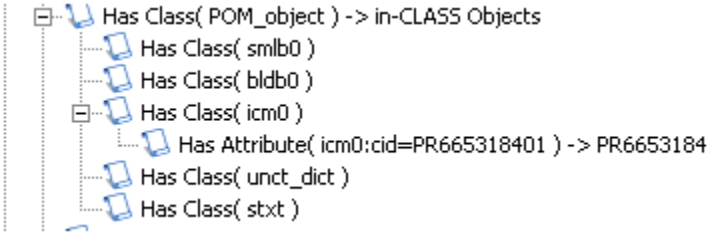
Example: configuring access rules for Multi-Site use cases

Administrators can use Access Manager to block classification objects (ICOs) from being exported when sharing a classified object (any classifiable workspace object) using a Multi-Site operation.

When configuring access rules for Multi-Site use cases in Classification:

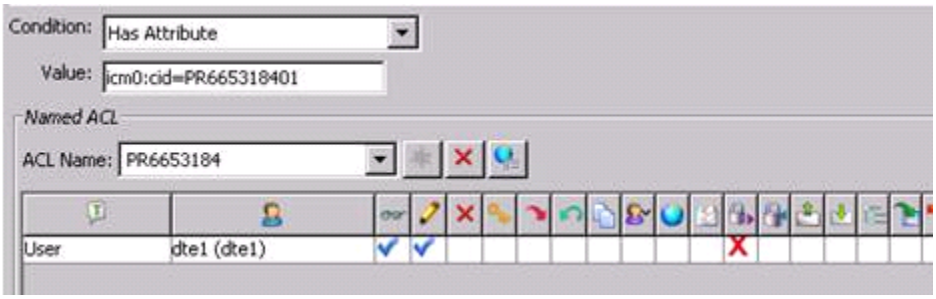
- Access checks are configured and checked at the exporting site only.
- To successfully export an object, the target site must be allowed to import from the given source site.

In this example, the Access Manager rule tree is configured as follows to identify Classification data from a specific class.



To achieve the necessary access control, configure the access control list as follows:

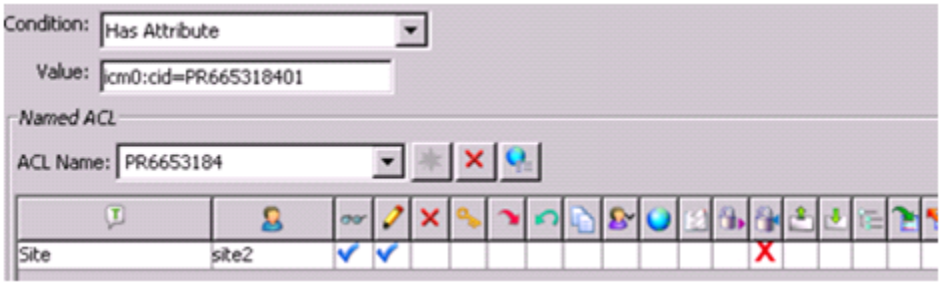
- To revoke access to exported ICOs from site 1 for a specific user (in this case, **dte1**) when performing the remote export operation, set the access rights as follows.



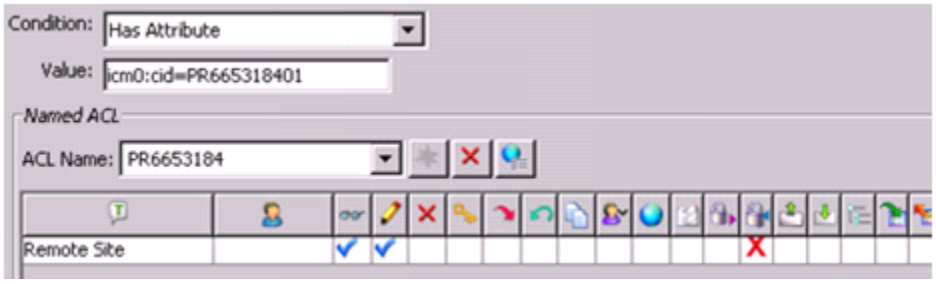
  To be able to perform access checks at a granular level instead of simply evaluating the remote site when remotely importing from any other site (for example, site 2), you must additionally set the **TC_check_remote_user_priv_from_sites** preference to **site1** and it must contain **site2** as one of the values. This indicates to the IDSM server on site 1 to evaluate the remote user from site 2.

  > Note:
  >
  > As this preference is honored only on the IDSM site, it is used only in a remote import operation.

- To revoke access to import ICOs from site 1, configure the ACLs as follows:
  - To stop objects from being exported to a specific site (for example, site 2), disallow site 2 to import from this site.



- To stop objects from being exported to all remote sites, disallow all remote sites to import from this site.



Synchronizing classification data

If the **ICS_synchronize_automatically** preference is set to **1** and the **subscriptionmgrd** process is running, when changes are made to a shared Classification definition, the data is automatically synchronized with the sites defined on the shared sites list, via the **subscriptionmgrd** process running at the local site and the **IDSM** service at the remote sites. Data can also be synchronized immediately using the **smlutility**.

Share classification hierarchy objects

1. In the **Class Details**, **View Details**, **Hierarchy**, **Dictionary**, or **Key-LOV** pane, click either the **Edit** button or the **Create** button, and then click **Select Sites**.

   The **Shared Sites** dialog box displays the possible sites with which you can share Classification data, as defined by the **ICS_synchronize_sites** preference.

2. From the **Available sites** list, choose the sites with which you want to share the object.

3. Click the right-arrow button.

   The sites are now displayed in the **Shared sites** list.

4. Click **OK** to select the sites and close the **Shared Sites** dialog box.

Each time that the object is saved, it is automatically updated to each of the shared sites, provided that automatic synchronization has been enabled.

Remove sharing of classification hierarchy objects

1. In the **Class Details**, **View Details**, **Hierarchy**, **Dictionary**, or **Key-LOV** pane, click either the **Edit** button or the **Create** button, and then click **Select Sites**.

   Teamcenter displays the **Shared Sites** dialog box.

2. In the **Available sites** list, remove the site with which you no longer want to share the object by selecting the site name and clicking the left-arrow button.

3. Click **OK** to close the **Shared Sites** dialog box.

4. **Manually remove the object** from the classification hierarchy at the remote site.

   Alternatively, you can delete a class using the **smlutility** utility.

Delete a class at the master site

1. Remove the sharing of the class at the master site.

2. **Manually remove the object** from the classification hierarchy at the remote site.

   Alternatively, you can delete a class using the **smlutility** utility.

Transfer object ownership

1. Ensure that the **TC_directly_transferable_classes** preference contains the following values:

   icm0
   icm1
   smlb0
   smlb1
   bldb0
   bldb1
   stxt
   unct_dict
   smllabel

2. In the hierarchy tree, select the object for which you want to claim ownership.

3. In the definition pane, located on the right, click the **Transfer Ownership** button. The system displays a confirmation dialog box.

4. Click **OK** to transfer ownership to your local site.

Add or remove attributes in shared classes

1. **Copy the class** to be modified along with all its children. This creates new class IDs for each class.

2. **Add or remove attributes** in the copied hierarchy.

3. Share the new hierarchy.

4. On all remote sites, move the local ICOs from the old classes to the new.

5. Delete the old hierarchy from all sites.

6. To retain the original class IDs, copy the new class hierarchy back to the old IDs and copy the ICOs back to the appropriate classes.

Transitioning from text file sharing to Multi-Site Collaboration

Overview of transitioning from text file sharing to Multi-Site Collaboration

If you share Classification information using text files, such as **sml** files or PLM XML files, and now want to share this information using Multi-Site Collaboration, you must perform certain steps to ensure that the classification hierarchy and all corresponding ICOs are complete and accurate. You can do so without having to delete the ICOs that you create at each site. Before you begin using Multi-Site Collaboration, you must perform the following steps:

1. Create a Classification master site that includes all classes from all sites.

2. Remove the classification hierarchy at all remote sites.

3. Share the classification hierarchy from the site containing the master hierarchy to all the remote sites using the Multi-Site Collaboration features found within Classification Admin. After this step, the ICOs that remained at the remote sites are correctly housed in the new hierarchy.

Creating a master site

When using Multi-Site Collaboration with Classification, you must define one site that contains the master classification hierarchy. The selection of this site depends on your business factors. After you select an appropriate site, you must import those classes from the remote sites that do not already exist in the master hierarchy to ensure that the master hierarchy is complete.

To export classes without their ICOs at the remote site, use the PLM XML **Export** option and choose the **ICSExportSubtree** transfer rule.

You should only export those classes not found in the master hierarchy. If you create guided component search data at remote sites, remember to export this data.

To import the missing classes into the master hierarchy, use the **Import** option and choose the **incremental_import** transfer rule. The master hierarchy is updated with the classes from the remote sites.

> Warning:
>
> If you add attributes to classes at remote sites, you must check the values of the ICOs belonging to these classes after the import. Depending on where you inserted attributes in the class, there may be errors in the ICO values.

Remove classification hierarchy at remote sites

To repopulate a site with a classification hierarchy using Multi-Site Collaboration, you must first remove the existing classification hierarchy at all remote sites.

> Warning:
>
> This task carries a multitude of risks to the integrity of your classification data. Only an experienced Classification administrator should perform this procedure.

1. Back up the hierarchy structure at the local sites by backing up your database. Also, export the hierarchy using PLM XML. To do this, select the **SAM** node, and choose the **ICSExportSubtreeWithICOS** transfer rule for the export. This transfer rule includes all attributes, key-LOVs, and ICOs used in the classes.

2. Remove the class hierarchy at each site with the **smlutility** using the following command:

   ```
   smlutility -delete -u=user_name -p=password -g=group_name
    -id=SML_CLASS -recurse -force
   ```

   SML_CLASS represents the class that you want to delete from the hierarchy. This command deletes the class *and* all its child classes.

   > Warning:
   >
   > The Classification root class is **ICM**. *Do not delete this class*. Delete each class that is directly underneath this class in the hierarchy.

   The **-recurse** option ensures that everything beneath the specified class is removed.

   The **-force** option causes classes to be deleted even if they contain ICOs. The ICOs are not deleted.

> Note:
>
> Remember to remove the guided component search class (GCS) from the hierarchy.

It may be necessary to enter this command multiple times as it sometimes leaves a residual hierarchy. To check which classes still exist in the hierarchy, you can enter the following command:

```
smlutility -list user_name password group_name class
```

Only the **SAM** and **ICS** classes should remain.

3. Delete all attributes from the Classification dictionary with the following command:

```
smlutility -delete -u=user_name -p=password -g=group_name -attribute
-id=*
```

4. Delete all key-LOVs with the following command:

```
smlutility -delete -u=user_name -p=password -g=group_name -keylov -id=*
```

Sharing classification hierarchy to remote sites

Share classification hierarchy using the Multi-Site Collaboration dialog box from within Classification Admin. When you share a class to a site, all dependent objects such as parent classes, attributes from the dictionary or key-LOVs are shared automatically. Therefore, you only need to share each child node.

> Note:
>
> If you use guided component search, remember to share this data.

Customizing the hierarchy tree display

Hierarchy tree customization overview

You can customize the images used to represent groups and classes in the hierarchy tree, providing a means to easily identify the contents of a particular group or class.

Custom symbols are stored in the database and are propagated to the client machine when the tree node corresponding to the symbol is rendered for the first time. Synchronization between the database and client machines is maintained to ensure that the images are consistent and up-to-date.

Associate a custom symbol with a group or class

Custom symbols provide a means of easily identifying classes and groups in the hierarchy structure. You can depict the purpose and contents of an individual class or group by associating a descriptive image with it.

1. In the hierarchy tree, choose the group or class with which you want to associate the custom symbol.

2. Click the **Edit** button . You must be in edit mode to modify group or class definitions. The definition pane displays information about the group or class on the right.

3. Click the **Add Icon** button.

   Teamcenter displays the **Select Image File** dialog box.

4. Using the **Select Image File** dialog box, locate and choose the image that you want to associate with the group.

   You must use standard Java graphic file formats (GIF and JPG).

5. Click the **Open** button.

   Teamcenter closes the **Select Image File** dialog box.

6. Click **Save** on the Classification Admin toolbar.

   The new image displays in the hierarchy tree and Teamcenter creates a dataset corresponding to the group or class in the database. The image file is a named reference of this new dataset.

Remove a custom symbol from a group or class

Images that have been associated with groups and classes to create custom symbols in the hierarchy tree can be removed and replaced by the default symbols for the respective object.

1. In the hierarchy tree, select the group or class from which you want to remove the custom symbol.

2. Click the **Edit** button . You must be in edit mode to modify group or class definitions. The definition pane displays information about the group or class on the right.

3. Click the **Remove Icon** button.

4. Click **Save** on the Classification toolbar.

   The custom symbol is replaced by the default symbol for the group or class in the hierarchy tree.

Configuring Resource Manager features

Configuring the guided component search

Guided component search overview

A guided component search (GCS) accelerates the search for matching components within an assembly. This search allows you to choose from a list of only those components that physically fit into the first component of your search. Before you can use the GCS, you must define the following:

- Global connection types

  This is a list of all possible connection types you use in your classification hierarchy. Each possible connection type contains a set of Classification attributes and comparison criteria. Each connection type is stored separately in the database.

- Connection point definitions for classes

  These are the elements that define what kinds of components match components of this class. Each class can have any amount of connection point definitions. Each definition has a direction (upward or downward), a physical connection shape (plug or socket), a quantity, and a connection type. For each class, you must map the class attributes to the connection type attributes. Based on the connection point definition, the system automatically creates connection points containing direction, shape, and individual attribute values for each ICO in the class.

Based on this information, the system checks for a matching component. The system finds a match if the following match:

- GCS connection type

  Both components must have the same connection type.

- Physical shape of GCS connection point

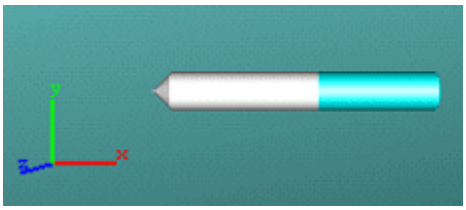
  There are two possible physical shapes: plug and socket. Components must have opposite connection point shapes to match. For example, component A with a plug shape fits into component B with a socket shape.

- Values of GCS connection point attributes

  For each component in a class, the system creates a connection point containing mapped attribute values for each connection point definition contained in the class. Each time an ICO is saved, the system updates the connection point attributes. The values of the connection point attributes have to match with respect to the connection type comparison operators.

Connection definition is done on a class level, not for specific instances. This means that all ICOs in one class use the same comparison criteria to search for matching components.

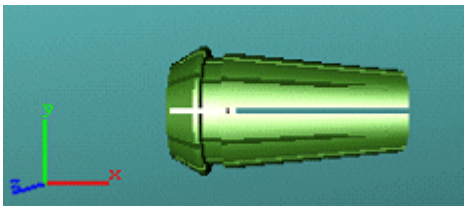
Consider the following example. Determine if the components shown match according to the principles of the guided component search.



Drill with a *cylinder* connection type and *plug* physical shape and the following pertinent attribute values:

–4110 Shank Diameter = 10.0

–4100 Parallel Shank Type = 'A'



Collet with a *cylinder* connection type and *socket* physical shape and the following pertinent attribute values:

–4120 Seat Diameter = 10.0

The connection type comparison criteria are as follows:

Attribute ID	Attribute name	Comparison criteria
–4121	Chuck Clamping Range – min	Plug ≥ Socket
–4122	Chuck Clamping Range – max	Plug ≤ Socket
–4100	Parallel Shank Type	Plug = Socket

In each class (drill and collet), the class attributes are mapped to the connection type attributes as follows:

Connection type attribute	Mapped class attribute for drill (plug)	Mapped class attribute for collet (socket)
–4121	–4110 Shank Diameter	–4120 Seat Diameter – 0.5
–4122	–4110 Shank Diameter	–4120 Seat Diameter
–4100	–4100 Parallel Shank Type	A

Evaluating the connection type attributes for these two components, their values are as follows:

Connection type attribute	Connection point attribute values for drill (plug)	Connection point attribute values for collet (socket)
–4121	10.0	10.0–0.5 = 9.5
–4122	10.0	10.0
–4100	A	A

The final comparison performed by the guided component search is as follows:

Connection type attribute	Attribute values for drill (plug)	Connection type comparison criteria	Attribute values for collet (socket)	Results
–4121	10.0	≥	9.5	Matches
–4122	10.0	≤	10.0	Matches
–4100	A	=	A	Matches

The guided component search finds that these two components match.

There are two things to remember about this type of search:

- All attributes must match.
- The system differentiates depending on the connection point from which you begin the search. In the previous example, when the search begins at the drill (plug-side), the system searches for a component that has the following:
  - A minimum clamping range value that is less than or equal to 10.
  - A maximum clamping range value that is greater than or equal to 10.
  - A parallel shank type equal to A.

  If you begin the search at the collet (socket-side), the system searches for the following:
  - A minimum clamping range value that is greater than or equal to 9.5.
  - A maximum clamping range value that is less than or equal to 10.
  - A parallel shank type equal to A.

Note:

You can search for matching components in Active Workspace. See *Search for matching components* in the Active Workspace help.

Enable array attribute for guided component search

You can perform comprehensive and flexible searches within the guided component search (GCS) using the array attributes. Array attributes allow you to include multiple values within a single attribute.

Example:

Consider an array string attribute *Color* with three values in the class A and class B components.

Class A component values

- **ClassAComp1:** Values: Red, Yellow, Green
- **ClassAComp2:** Values: Red, Yellow, Green
- **ClassAComp3:** Values: Red, Yellow
- **ClassAComp4:** Values: Red
- **ClassAComp5:** Values: Yellow, Green

Class B component values

- **ClassBComp1:** Values: Red, Yellow, Green
- **ClassBComp2:** Values: Red, Yellow, Green
- **ClassBComp3:** Values: Red, Yellow
- **ClassBComp4:** Values: Red
- **ClassBComp5:** Values: Yellow, Green

When performing a GCS for any of these components, the system identifies matching components based on the array attribute values as below:

- Search for *ClassAComp1* will find *ClassBComp2*, *ClassBComp3*, *ClassBComp4*, and *ClassBComp5*.

- Search for *ClassAComp2* will find *ClassBComp1*, *ClassBComp3*, *ClassBComp4*, and *ClassBComp5*.
- Search for *ClassAComp3* will find *ClassBComp1*, *ClassBComp2*, *ClassBComp4*, and *ClassBComp5*.
- Search for *ClassAComp4* will find *ClassBComp1*, *ClassBComp2*, and *ClassBComp3*.
- Search for *ClassAComp5* will find *ClassBComp1*, *ClassBComp2*, and *ClassBComp3*.

As a rule, if at least one of the source elements is in the target list, at any position, then it is a candidate for matching components. When you create a new GCS connection type that should be used for array attributes, you must enable array attribute support.

Procedure

1. Create a new global connection search **connection type** and add the desired array attributes.

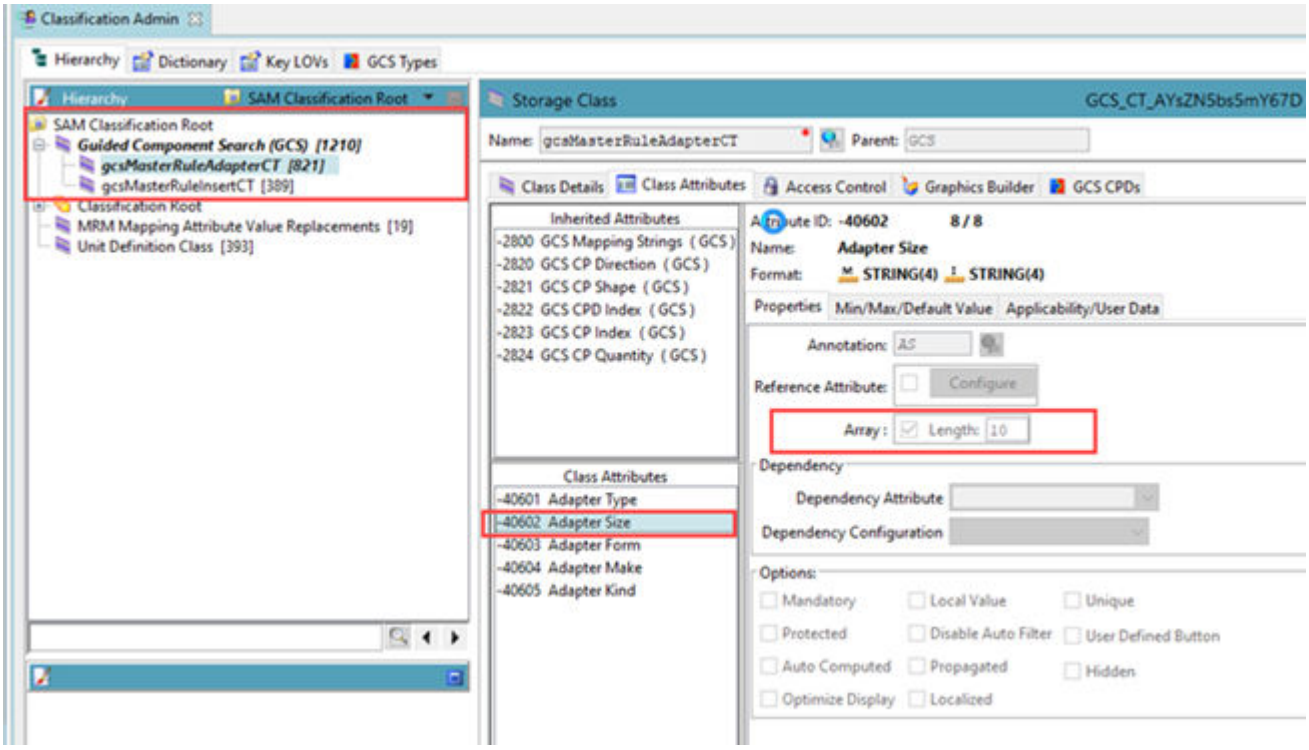
   For the newly created GCS connection type, a corresponding class is automatically created.

2. In Classification Admin, navigate to the newly created class under **SAM Classification Root** > **Guided Component Search**. The class has the same name as the GCS connection type.

3. Edit the class to configure the attributes.

4. Select the class attribute that must support guided component search with array attributes.

5. In the **Properties** pane, select **Array** and specify the required **Length**.



6. Save the class.

Define alternate values for guided component search

Guided component search finds matching components based on predefined rules that consider the classification connection attributes. By default, the system searches for all objects with the same connection. However, the **MRMGCSAlternates** preference allows you to define alternative mappings for the GCS rules to find matching components. The system finds objects with the specified connection as well as those using the alternative connections.

Example:

Consider that you have classes for clothes and hangers. In the *T-Shirt* class, there is a *Size* attribute with values *XS*, *S*, *M*, *L*, and *XL*.

In the *Hanger* class, there is a *Size* attribute with values *extra small*, *small*, *medium*, *large*, and *extra large*.

You can define alternates to match these attributes. For example:

- *XS* and *extra small* are the same.
- *S* and *small* are the same.

- *M* and *medium* are the same.
- *L* and *large* are the same.
- *XL* and *extra large* are the same.

You can start a GCS search or display matching components for an *XL T-Shirt* and find an *extra large hanger*.

Procedure

1. Define the alternate values for GCS in the preference as per the syntax below:

   Preference name: **MRMGCSAlternates**

   Syntax: `[connection_type:]original_value::alternate_value1[ | alternate_value2]*`

   For a specific `original_value`, one or multiple alternate connections can be specified.

   For example:

   ```
   MRMGCSAlternates
   PIN::SOCKET
   SOCKET::PIN
   ```

   The optional `connection_type` checks for alternates for the specified connection types only. If `connection_type` not specified, the alternates apply to all connection rules.

   For example:

   ```
   MRMGCSAlternates
   FKB::FKS | FKBS
   FKS::FKBS | FKB
   FKBS::FKB | FKS

   MRMGCSAlternates
   my_CT:orig::alternate
   my_CT:X::Y | Z
   my_CT:Y::X | Z
   my_CT:Z::X | Y
   ```

   You can specify one matching alternate value or more using the "**|**" separator. Which means, in the above example, `FKB::FKS | FKBS`, FKB matches with FKS as well as with FKBS.

By default, the system searches for all matching objects with the same connection. This preference defines alternative mappings for the GCS rules to find components that match. In this case, the system still finds objects with the given connection but also other objects using the alternative connections.

Examples

For the T-shirt example above, the **MRMGCSAlternates** preference can be defined as below so that the alternate values are also considered during the GCS.

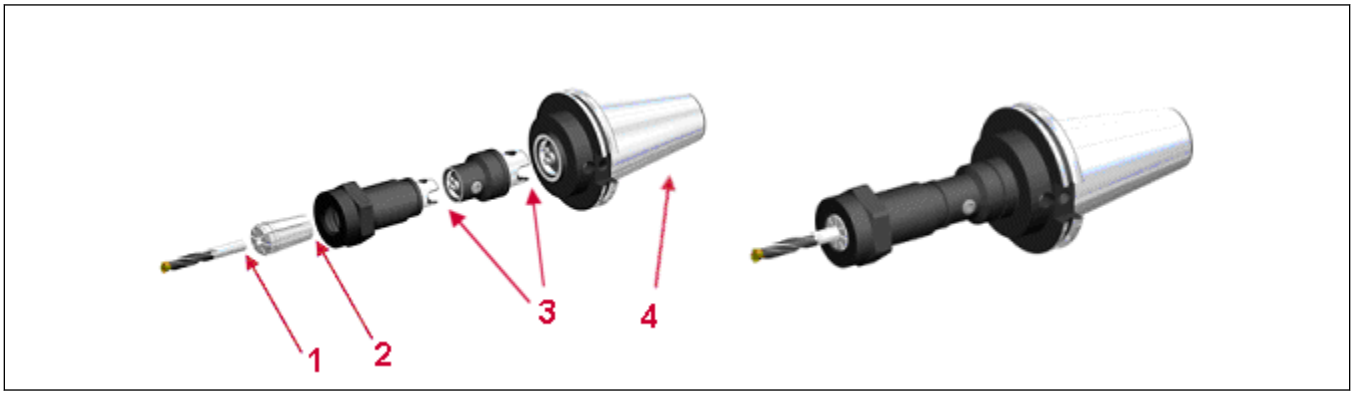
```
MRMGCSAlternates
XS::extra small
S::small
M::medium
L::large
XL::extra large
extra small::XS
small::S
medium::M
large::L
extra large::XL
```

Working with global connection types

Overview of global connection types

Before you can use the guided component search, you must create a set of global connection types from which to choose when you create connection point definitions within the individual classes. These are stored in the database.

Connection types represent the junctions between two parts that must fit together, such as a holder, a round insert, a rhombic insert, a square shank, or a tapered shank. In the following figure, there are four different connection point types: cylinder (1), collet (2), KM (3) and taper (4).



Connection point types

You can create as many connection types as you need to describe the connection shapes of your assemblies.

Each connection type has:

- A name.
- A set of attributes (optional—if no attributes exist, the system simply checks whether the connection types match).
- A comparison criterion for each existing attribute.

You manage connection types in the **GCS Types** tab in the Classification Admin application. This tab contains a list of all existing GCS types, sorted alphabetically. The right pane shows the attributes and their comparison criteria belonging to the GCS type selected on the left.

To update the connection type list at any time, click the **Refresh/Reload** button at the bottom of the **Existing GCS Connection types** pane.

Create connection types

You can create connection types manually or use the **gcs_import** utility to create multiple connection types at once.

To create connection types manually:

1. Click the **GCS Types** tab.
2. Click the **Create** button on the toolbar. The system displays the **New GCS Connection Type** dialog box.
3. Enter a unique name for the connection type and click **OK**. The new connection type is added to the list of connection types in the left pane. You can now add attributes to the new connection type.

   > Note:
   >
   > You can only add attributes to those connection types that have not yet been used in connection point definitions.

4. Click the **Add Attribute** button at the bottom of the attribute pane. The **Add Attribute** dialog box opens providing you access to all available dictionary attributes.
5. Select the desired attributes and click **OK**. The attributes are displayed in the **Attributes & GCS Comparison Criteria** pane.

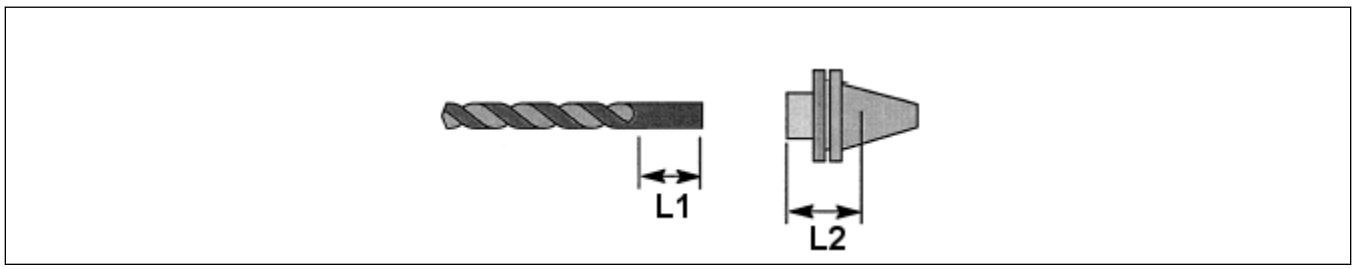
6. In the **Operator** box, select from the values in the following table.

Operator	Meaning
n.a.	Not applicable
=	Equal
<>	Not equal
<	Less than
<=	Less or equal
>	Greater than
>=	Greater or equal
Fact.	Relative range
Diff.	Absolute range

7. Click **Save** to save the connection type.

Using the factor and difference operators

If you use the factor (**Fact.**) and difference (**Diff.**) operators, you can type values in the **Plug-Sided Value** and **Socket-Sided Value** boxes in the **Connection Points Definition** pane. If, for instance, you create a connection type to represent a drill fitting into a holder, there are certain restrictions on the length of the drill and how far it fits into the holder. In the following figure, at least 90 percent of **L1** must fit into the holder and the **L2** distance must be at least 80 percent filled by the drill that fits into it; otherwise, the drill is not held correctly be the holder.



Tool and holder parameters

Using the **Fact.** operator, you can express this in the connection point definition as follows:

Attribute ID	Attribute name	Plug-side value	Operator	Socket-side value
–10030	Length	90	**Fact.**	80

Expressed mathematically, the range is:

(90*L1)/100 .... (L1*100)/80

If L1 = 10 cm, the holder value required in this example must fall between 9 cm and 12.5 cm. Conversely, if you begin your search from the holder, which has an L2 value of 15 cm, you must find the tool whose L1 length is as follows:

(80*L2)/100 ... (L2 * 100)/90

Therefore, any tool with an L1 length between 12 cm and 16.7 cm will fit into the holder.

The **Diff.** operator functions in a similar fashion using an absolute range instead of a relative range. For example:

Attribute ID	Attribute name	Plug-side value	Operator	Socket-side value
–10030	Length	1	**Diff.**	2

If you begin from the drill whose L1 value = 10cm (the plug-side value), the holder value L2 must be in the range of 10 – 1 to 10 + 2, or 9 to 12 cm.

Delete connection types

> Warning:
>
> If you delete a connection type, the system deletes this connection type and all connection point definitions and connection points using this connection type. This could seriously impair the guided component search.

1. Click the **GCS Types** tab.
2. Select the GCS type from the list in the **Existing GCS Connection Types** pane.
3. If you are not in edit or new mode, click the **Edit** button .
4. Click the **Delete** button . The connection type is removed from the list of existing GCS connection types.

Create a connection type report

You can create a report that lists the following information about the selected connection type:

- The name, attributes, and comparison criteria of the connection type.

- The number of classes containing this connection type in their connection point definitions and how many connection point definitions of this type they possess.
- A listing of the classes in which the connection type is used, along with the connection point definition index, the direction, shape, quantity and number of connection points.

1. Click the **GCS Types** tab.
2. Select the connection type from the **Existing GCS Connection Types** pane.
3. At the bottom of the **Existing GCS Connection Types** pane, click **Connection Type Report** . A dialog box displays this report.
4. (Optional) Print this report by clicking **Print** .
5. (Optional) Save the report by clicking **Save** .

Working with connection point definitions

Overview of connection point definitions

Connection point definitions are the elements that designate which connection points are created for all components in a class. Each class can have zero, one, or multiple connection point definitions. These definitions contain:

- An index.

  The system numbers the connection point definitions in ascending order.

- A **Defined in Class** entry specifying in which class the definition was created.

  Connection point definitions are inherited from a parent class to all subclasses.

- A physical connection shape.

  A connection point can have one of the following shapes:

  – **Plug** means that this shape will fit into a component.

  – **Socket** means that this shape will have something fitted into it.

  When searching for a component to fit into a plug component, the system searches only for socket components, and vice versa.

- A specific direction: upward or downward.

You can select from **up** ▲ or **down** ▼. The system uses direction to position the new component in the assembly tree structure in Resource Manager. If you are building a tool assembly, the connection points on a component that point to the machine side correspond to the upward direction. Connection points that point to the workpiece (or cutter) side correspond to the downward direction. You should build cutting tools with the components on the machine side at the top down to the components on the workpiece side to allow the usage of propagated attributes in Resource Manager. When a matching component is found for an upward connection point, the new component is added above the current as a parent. This means that the current resource slips one level down. When a matching component is found for a downward connection point, the new component is added below as a child of the current component.

> Note:
>
> Because of the parent-child relationship in the hierarchy, a class can have only one connection point definition pointing in the upward direction.

- A quantity.

  A component can contain a number of identical connection points, such as in the case of a helical milling tool.

- A specific connection type.

  You can select from one of the connection types that you already defined in the database.

- An attribute mapping.

  Each connection type has its own set of attributes. The attributes belonging to the class for which you are creating the connection point definition need to be mapped to these connection type attributes.

When you save connection point definitions, Teamcenter creates a connection point for each component in the class. If the class contains multiple connection point definitions, Teamcenter creates a connection point for each definition.

> Note:
>
> Updating the connection point data for each ICO in the a class can be time-consuming, depending on the number of instances in the class.

At any time while you are working with connection point definitions, you can update connection points derived from the selected connection point definition, or update all connection points of all objects pertaining to the selected class using the **Update** buttons .

Create connection point definitions

You can create connection point definitions manually or use the **gcs_import** utility to create multiple connection point definitions at once.

To create connection point definitions manually:

1. In the **Hierarchy** tab, select the class to which you want to add the connection point definitions.
2. Click the **GCS CPDs** tab.
3. Click .
4. In the **Connection Point Definitions** pane, click the **Add Connection Point** button . A new row appears in the pane.
5. Enter values for the shape, direction, connection point quantity, and connection type.
6. For each connection type attribute, you can define the mapping method. You can choose from these options:

   - Map to an absolute value. In the **Mapped Attributes** section, enter the value in **Mapping** box, for example, **10**. You must place text within single quotation marks, such as **'Released'**.
   - Map to a specific class attribute value. In the **Mapped Attributes** section, enter the attribute into the **Mapping** box using the arrow buttons as follows:
     - > maps the selected class attribute to the selected connection type attribute.
     - *> checks if any connection type attribute IDs match class attribute IDs and automatically maps these.
     - < removes the attribute mapping for the selected connection type attribute.

     Note:

     You can also enter attribute IDs manually into the box.

   - Map using a mathematical expression. Enter the expression in the **Mapping** box. You must put a hash sign before the attribute number, for example, **#-2503**. You can use one attribute and a value, such as in the following example:

     ```
     #-4120+0.5
     ```

     The absolute value 0.5 is added to the value of class attribute - 4120 and the result stored in the connection point attribute.

     The following four mathematical operators can be used:

+, -, *, /

When you define expressions, ensure that the result complies to the given attribute format, for example, **real** /**int** or number of digits.

- Map using the **SUBSTR** operator.

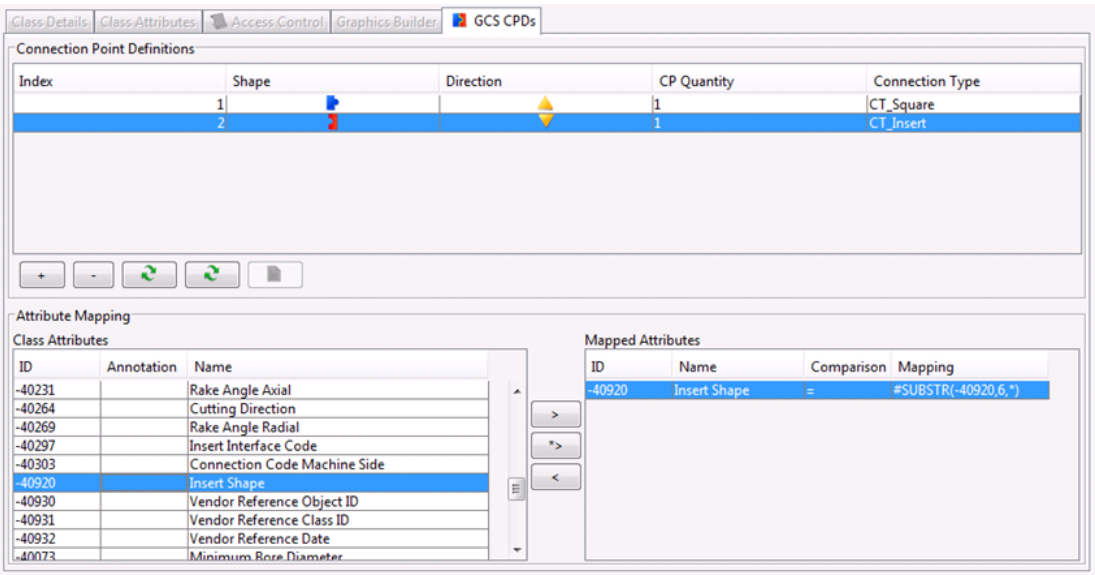
You must ensure that the format of the mapped attribute values match. You cannot, for example, map a string to a float.

7. Click **Save**.

Teamcenter saves a connection point for each component in the class. If you created multiple connection point definitions, Teamcenter creates a connection point for each definition.

Using the SUBSTR operator when mapping attributes

When mapping attributes during the guided component search, you can use a substring operator in the **Mapping** column that specifies that only specific characters in a string are mapped.



The entry must have the following format:

#SUBSTR(*attribute-ID***,** *first-character-index*,*last-character-index***)**

Teamcenter looks for the specified attribute, and then copies the characters from *first-character-index* to *last-character-index* to the target attribute.

The table demonstrates which characters are mapped to the target attribute from a source attribute with an ID of **-1102** and a value of **AXR15739**.

This entry	Maps these characters
#SUBSTR(-1102,1,3)	**AXR**
#SUBSTR(-1102,4,*)	**15379** Note: If you use the wildcard symbol as the second index, Teamcenter copies all characters from the first index to the end of the string.
#SUBSTR(-1102,2,4)	**XR1**

Modify connection point definitions

1. Select the class that contains the connection point definition in the class hierarchy.
2. Click the **GCS CPDs** tab.
3. If you are not in edit or new mode, click the **Edit** button .
4. Select the line containing the connection point definition that you want to modify and make the necessary changes or add or remove lines.
5. Click **Save**.

Delete connection point definitions

1. Select the class that contains the connection point definition in the class hierarchy.
2. Click the **GCS CPDs** tab.
3. If you are not in edit or new mode, click the **Edit** button .
4. Select the line containing the connection point definition that you want to delete.
5. Click the **Remove Connection Point Definition** button at the bottom of the pane.
6. Click **Save**.

Create a connection point definition report

You can create a report that lists information about the connection point definitions for the selected class. This information includes the index, connection type, direction, defined class, shape, quantity, number of connection points, and number of ICOs containing connection points.

To create a connection point definition report:

1. Select the desired class in the class hierarchy.
2. Click the **GCS CPDs** tab on the hierarchy tab.
3. At the bottom of the **Existing GCS Connection Types** pane, click the **Connection Point Definition Report** button. A dialog box opens displaying this report.
4. (Optional) Print this report by clicking **Print**.
5. (Optional) Save the report by clicking **Save**.

Allowable types for mapping attributes to connection types

You can map the following class attribute types to guided component search connection type attributes.

Class attributes	**Guided component search connection type attribute**				
	Integer	String	Date	Real	Key-LOV
Integer	✓	✓	X	✓	X
String	✓	✓	X	✓	✓
Date	X	X	✓	X	X
Real	X	✓	X	✓	X
Key-LOV	✓	✓	X	✓	✓

Note:

You can only use the **SUBSTRING** operator if the source attribute is of the **String** type.

Importing a tool vendor catalog

Using a tool vendor catalog

You can import tool vendor catalogs directly into Teamcenter. These catalogs use the Generic Tool Catalog (GTC) format based on ISO 13399 tooling standards. When tool vendors deliver their tool catalogs in this format, you can import them into a vendor tooling hierarchy in the Manufacturing Resource Library. You can then select tool components in the vendor catalogs and map them to a customer branch of the hierarchy. The tool components you choose are automatically mapped to existing Manufacturing Resource Library tool classes. If there are any attachments (for example, 3D models) with the vendor components, you can import these as well. You can build a tool assembly based on these components, create graphics for it, send it to NX, and use it to machine parts in NX. A variety of

tooling vendors provide data in the generic tool catalog format that can be used with Resource Manager. Contact your Siemens Digital Industries Software representative for more information.

Perform the following tasks to begin using a vendor catalog.

Role	Task	Application
Tooling/classification administrator	Imports the vendor catalog hierarchy into the database.	Classification Admin
Tooling/classification administrator/ Resource author/Tooling engineer	Imports vendor product data (part of or complete vendor catalog).	Classification Admin, Classification, Resource Manager, or using the **import_step_part21_files** utility.
Resource author/Tooling engineer	Imports vendor 3D models.	Classification or Resource Manager
Resource author/Tooling engineer	Creates tool assemblies using the components in the customer hierarchy.	Resource Manager
Part planner	Assigns resources to the manufacturing process.	Part Planner
NC programmer	Uses resources in the CAM area.	NX

Import a tool vendor catalog hierarchy

Before you import a tool vendor catalog hierarchy, ensure that you have installed version 3.1.1 or later of the Manufacturing Resource Library. The version must contain the generic mapping views that map from the vendor tool component classes to the Manufacturing Resource Library classes.

1. Obtain a copy of the tool vendor catalog from the tool vendor Web site.

2. Store the catalog in a directory that is accessible to the Teamcenter server machine. You can do this by:

   - Copying the catalog to a directory on the Teamcenter server machine.
   - Mapping a directory to the Teamcenter server machine.

3. Store the path to the directory created in the previous step in the **MRMGTCVendorCatalogRootDir** preference.

   This is a multivalue preference. You can specify multiple directories where vendor catalogs are stored.

4. In the Classification Admin application, select the **Vendor Catalogs** node in the hierarchy and click the **Import Vendor Class Hierarchy** button in the toolbar.

   Teamcenter displays the **Import Vendor Catalog Hierarchy** dialog box that lists all the tool vendor catalogs found.

5. Select the desired vendor catalog and click **OK**.

Teamcenter imports the tool vendor catalog hierarchy under the **Vendor Catalogs** node of the classification hierarchy.

Import tool vendor product data

Vendor product data consists of tool components such as drills, holders, inserts, and adapters. You can import vendor product data using two methods:

- Import the data from the rich client using the Classification, Classification Admin, or Resource Manager applications.
- Import the data using the **import_step_part21_files** utility.

To import using the rich client:

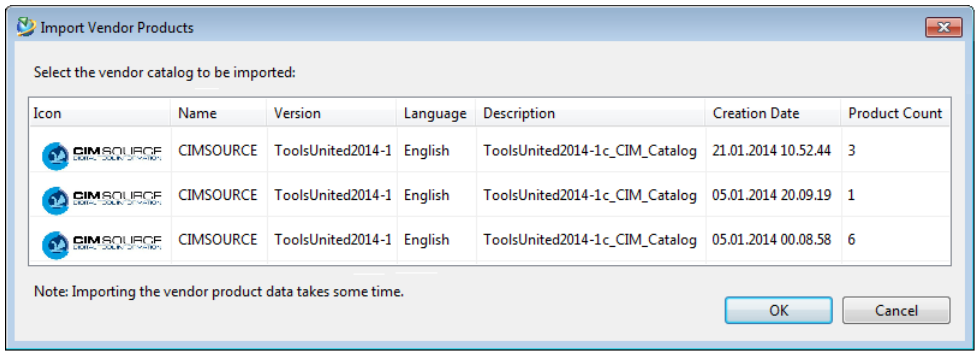
1. In the classification vendor hierarchy, select a class at any level.

   The class you select determines what packages are imported. Teamcenter imports all product data from the selected class downwards.

2. Do one of the following:

   - Choose **Import Vendor Product Data** from the shortcut menu.
   - Click **Import Vendor Product Data** in the toolbar.

3. In the **Import Vendor Products** dialog box, select the package that you want to import from the list of available Generic Tool Catalog packages.



   Teamcenter imports the product data to the selected classes.

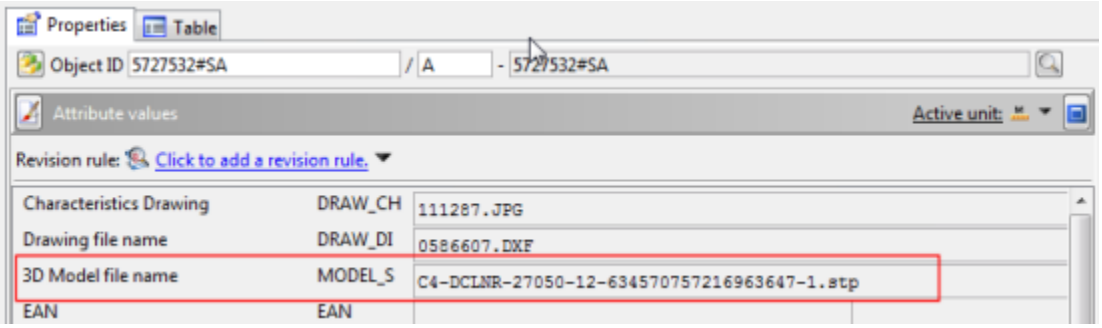
4. Click **Import**.

5. (Optional) Open the resulting log file to view the import state of individual components.

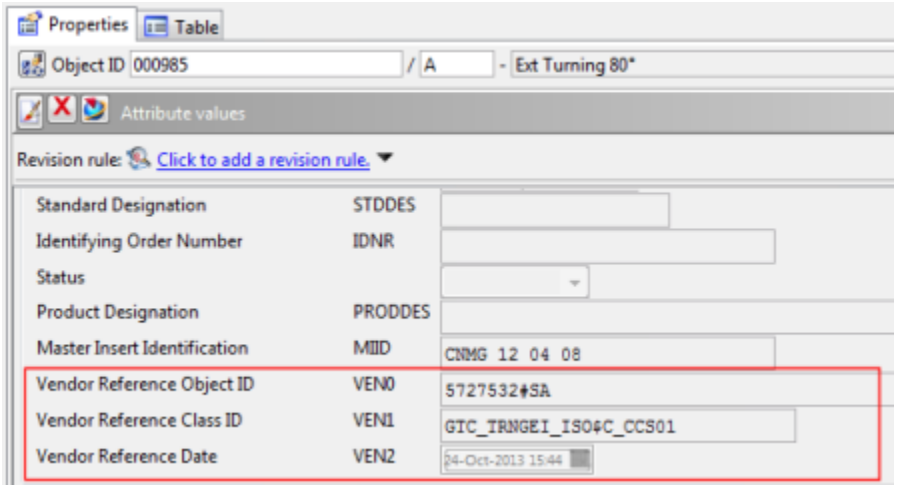
> Caution:
>
> Depending on the size of the package you select, this action can take several hours to complete.

Understanding the linking mechanism for vendor data

When you import vendor catalog data, each component has an attribute, **3D Model file name** (attribute ID **-159003**), that stores a reference to the STEP file containing the 3D data.



In turn, when you map a catalog component to a customer component, the customer component contains attributes that refer back to the vendor component.



When you open a mapped customer component and initiate the 3D model import, Teamcenter uses this reference to locate the original catalog component. Teamcenter stores the ICO modification date from the vendor tool component in the **Vendor Reference Date** attribute during the mapping.

Define advanced attribute mapping

When mapping from one ICO to another (such as when mapping from vendor components to customer components), the source class can sometimes use different key-LOVs or string attributes than the target

classes. In this case, you must create advanced mapping definitions from attribute X to attribute Y for each value.

For example:

Source attribute ID	Target attribute ID	Source value	Target value
-1000	-3000	Y	1
-1000	-3000	N	0
-1010	-3010	LEFT	L
-1010	-3010	RIGHT	R
-1010	-3010	NEUTRAL	N

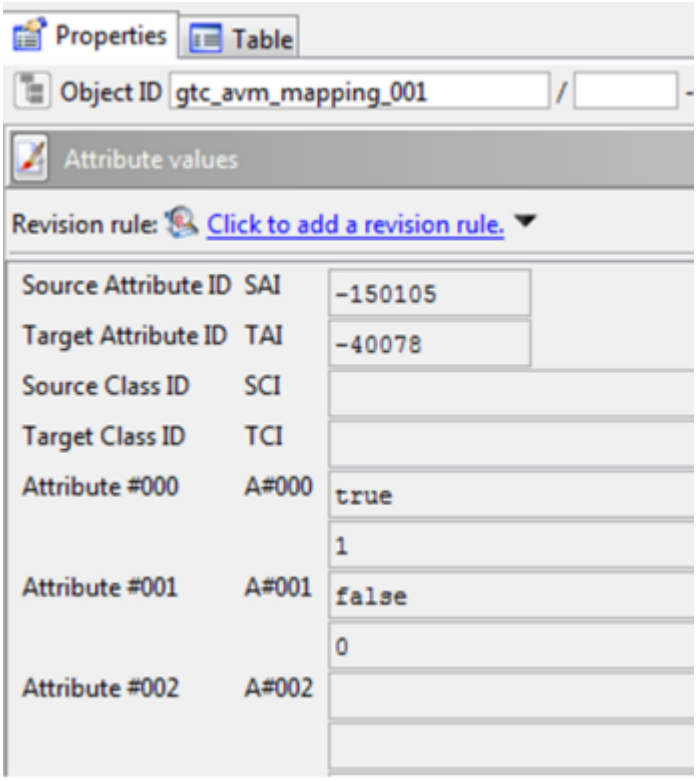
The attribute mapping definition is held in the **MRM Mapping Attribute Value Replacements** class (class ID **MRM_map_replacements**) that resides under the **SAM Classification Root** node in the classification hierarchy.

1. In Classification Admin, move the **MRM Mapping Attribute Value Replacements** class from the **SAM Classification Root** node to the **Classification Root** node using drag and drop.

2. In the Classification application, refresh the classification hierarchy.

3. Select the **MRM Mapping Attribute Value Replacements** class.

4. Add a new ICO for each new mapping definition.



The ID of the ICOs is irrelevant. If there are two or more ICOs that specify the same source and target attribute ID combination, the mappings are combined. By default, 195 attribute values can be mapped in one ICO. If more mapping definitions are needed, you can create multiple ICOs.

5. (Optional) Move the **MRM Mapping Attribute Value Replacements** class back to the **SAM Classification Root** node to protect it from inadvertent modification.

When mapping, Teamcenter first checks if an advanced mapping definition exists that pertains to the current target attribute ID. If so, it checks if the current source value is contained in the list and maps using that information.

Replace tool vendor catalog

Occasionally a vendor releases an updated version of their tool catalog. You can update your existing tool vendor catalog by first **removing** it and then **importing** the new catalog.

> Note:
>
> If there were products imported in the existing vendor catalog structure you are replacing, they will be removed when the existing structure is removed.
>
> If the user still wants to access the old products from the existing structure, they need to be re-imported into the new structure.

Remove tool vendor class hierarchy and data

Vendor product data is removed in two steps. The first step involves removing the catalog data and the second step removes the catalog class hierarchy.

1. Remove catalog data with this command using a Teamcenter shell window:

   ```
   smlutility -delete -u=<data-user> -p=<password> -g=<group> -ico
   -cid=<GTC-class-ID> -icos -wso -recurse
   ```

   This command is executed by a user in the same group as the user that imported the class structure.

2. Remove the catalog class hierarchy with this command using a Teamcenter shell window:

   ```
   smlutility -delete -u=<admin-user> -p=<password> -g=<group> -class
   -id=<GTC-class-ID> -icos -wso -recurse
   ```

Setting up site-specific properties to search resources by site

Step 1: set up site-specific properties

1. Ensure that the **MRMSiteDataEnabled** preference is set to **true**.

   This displays the **Create/update site specific data for resources** button in the Resource Manager on Rich Client application.

2. In Resource Manager on Rich Client, open a resource from the tooling library and click **Create/update site specific data for resources**.

   The first time you click the **Create/update resource site data** button, Teamcenter creates the following new classes under the **SAM** node in the hierarchy along with four new attributes.

Class name	ID	With these attributes
Resource Site Specific Data	**MRM_SITE_DATA**	**-46003 Resource Location** **-46004 Resource Status** **-46005 Resource Vendor** **-46006 Resource Machine**
Status	**MRM_SITE_DATA_01**	Inherits:

Class name	ID	With these attributes
		-46003 Resource Location
		-46004 Resource Status
Vendor	**MRM_SITE_DATA_02**	Inherits:
		-46003 Resource Location
		-46005 Resource Vendor
Machine	**MRM_SITE_DATA_03**	Inherits:
		-46006 Resource Machine

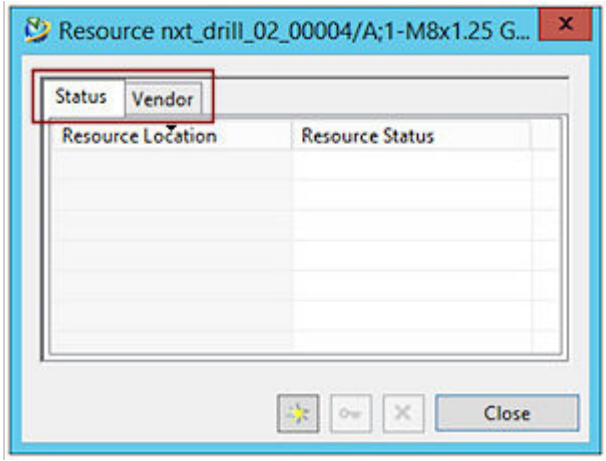
3. Add the class IDs for which you want to create the site-specific properties to the **MRMSiteData** preference.

   This preference defines the classes used to store the site-specific data for resources. For each class listed in this preference, a tab is displayed in the **Resource Site Data** dialog. For example, if the value of this preference is:

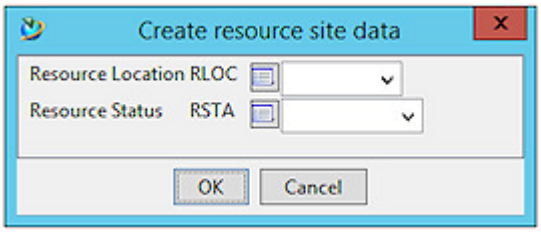
   MRM_SITE_DATA_01 (class name: **Status**)

   MRM_SITE_DATA_02 (class name: **Vendor**)

   The dialog box in Resource Manager on Rich Client displays two tabs called **Status** and **Vendor**.

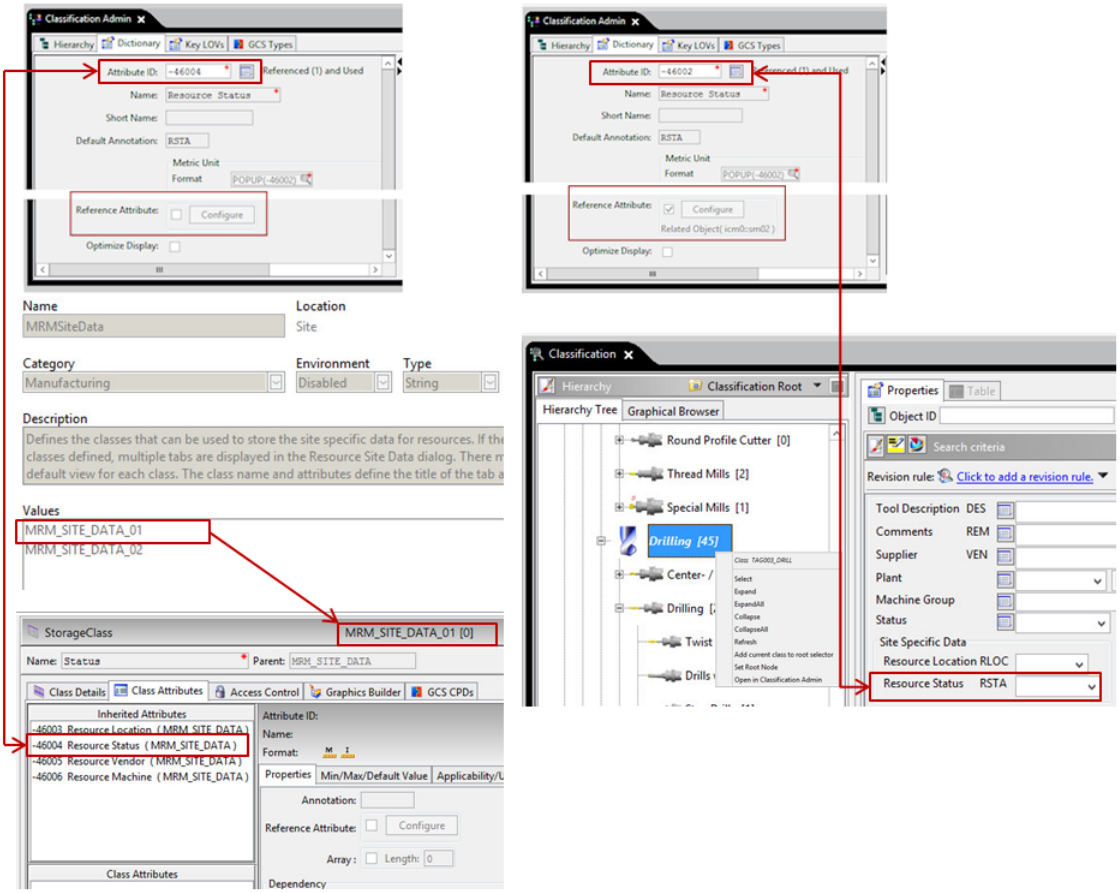


   The class name and attributes define the title of the tab and the attributes that are displayed in the tab. You can extend the **MRM_SITE_DATA** class structure to include any class that suits your needs. The attributes contained in the classes listed in this preference are the attributes that are displayed when you create new site-specific properties. For example, the **Status** class contains two attributes, and these are displayed when you create new site-specific data on the **Status** tab.



Step 2: set up the search for the site-specific properties

To set up the search for a specific property, you must create two attributes. The first attribute is added to the class listed in the **MRMSiteData** preference. The second attribute is a reference attribute that is added to the class. For example, for the **Resource Status** site-specific property, there are two attributes: a reference attribute (**-46002**) and a regular attribute (**-46004**). The regular attribute, **-46004**, is the one listed in the **MRMSiteData** preference. The reference attribute, **-46002**, is the one added to the class and for which you can search in the user interface.



1. Create reference attributes for each attribute for which you want to search.

    a. Create an attribute and select **Reference Attribute** during creation.

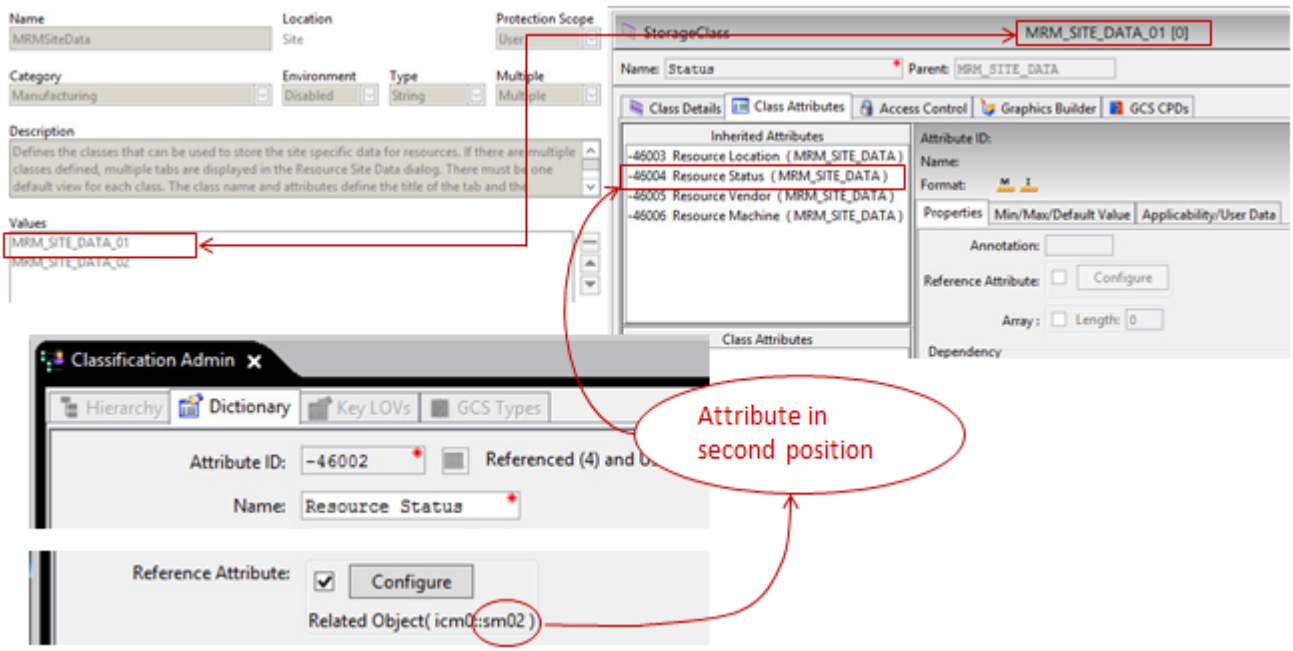
b. Click **Configure**.

c. In the **Configure Reference Attribute** dialog box, select **Related Object**.

d. Select the **icm0** class in the **POM Attribute** tree.

e. Select the correct attribute from the **POM Attribute** list.

This list contains an internal ID for every attribute. The reference attributes are hard-coded to refer to a specific position of the attributes in the classes named in the **MRMSiteData** preference. For example, if the reference attribute refers to the **icm0::sm02** attribute, Teamcenter searches for the value contained in the attribute that is in the second position of the attributes contained in the classes listed in the **MRMSiteData** preference.



> Caution:
>
> To avoid unexpected search results, when extending the site-specific data framework, create subclasses of the **MRM_SITE_DATA** class and add any new attributes to the parent **MRM_SITE_DATA** class allowing them to be inherited to the new subclasses. When you create a reference attribute referring to the new attribute, Teamcenter then searches for the new attribute value in the correct position.

f. Select the **Mfg0ResourceSiteDataRel** relation in the **Relations** list.

2. Add the reference attribute to each class in which you want to search.

3. Set the **ICS_reference_attribute_use_format_for_display** to **true**.

This preference specifies that the format of the reference attribute is used to determine the type of text field when searching in Resource Manager on Rich Client.

Working with the data dictionary

Data dictionary overview

For systems design, key building blocks or components of designs are reused to save time and effort in typical functional and logical model-structure design activities. To facilitate reuse of these building blocks or components in multiple designs across several projects or programs, Teamcenter provides a central organizational repository for them. Designers can search for and find components based on criteria and reuse them in their designs.

Typically, the organizational repository is tightly controlled and the components or blocks in it have gone through a formal, customer-specific approval process. Once approved, the blocks or components are labeled as ready to use. This central organizational repository is referred to as a *data dictionary*. It allows you to organize data by your own criteria with the goal of reuse.

You can control access to the classes and classification objects making up a data dictionary. You do this in the **Access Control** pane.

Set up a data dictionary

In Classification, a data dictionary is modeled as a *library*. A hierarchy, or section of a hierarchy, is designated as a specific library. A library contains a collection of nested classes. You can specify different sections of the classification hierarchy as different libraries.

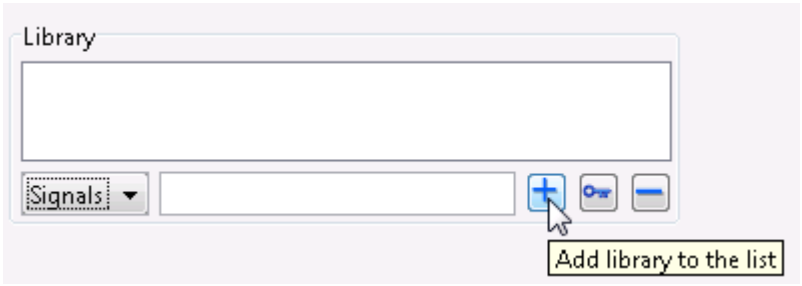
1. Create groups and classes that reflect the desired hierarchy for the data dictionary.

2. Select the topmost class in the hierarchy representing the dictionary.

3. Open the **Class Details** pane.

4. Do one of the following:

   - In the **Library** section of the pane, select the desired library from the list and add it to the box using the **Add** button.



The **Signals** value is saved in the **Fnd0LibraryType** LOV. You can extend your customer template to include additional entries to this LOV in the Business Modeler IDE.

- Type the name of a new library in the box and click +.

  Teamcenter adds the new value to the list of available libraries.

By default, you can choose a **Signals** library type. If you installed the MATLAB integration, Teamcenter also displays the **Behavior Model** library.

View library information

1. Select a classification class.
2. Click the **Library Information** button .

   Teamcenter opens the **Library Information** dialog box displaying:

   - The projects in which this library participates. A library can be associated with multiple projects.
   - The parent classes of the selected node.
   - The library type.

Deleting a data dictionary from the hierarchy

You delete a dictionary by deleting the class specified as a dictionary and all its child classes. This is done using the **Delete** button.

Importing and exporting hierarchy data

Overview of importing and exporting hierarchy data

You can import and export classification hierarchy data using either the import and export features in the Classification Admin application or using command line utilities. The following import/export methods are available:

- Interactive PLM XML method
- **plmxml_import** and **plmxml_export** command line utilities

  You can use the **transfer modes** as input parameters for these utilities.

Note the following points when importing and exporting using PLM XML:

- When importing with PLM XML, Teamcenter aborts the import if it finds errors in the data. When importing views, the import aborts if it encounters an attribute not contained in the parent class. You can change the view import behavior using the **ICS_import_view_drop_attributes_not_in_class** preference.

- For exporting, Classification Admin offers you additional transfer modes customized for classification data.

- Classification and Classification Admin differentiate between a user class (all objects exported as one PLM XML element) and an **admin** class (each object exported as an individual element). Classification Admin allows you to export **admin** classes only. To export user classes or ICOs, use the Classification application.

- Do not confuse these data import and export features with the **Export translated object** feature that imports and exports display names for translation purposes.

Export data using PLM XML

To perform these steps, ensure that you extend your data to the presentation layer.

1. In the class hierarchy, select the group, class or view that you want to use as the starting point for your export.

2. Click the **Export** button.

   The **Export** dialog box opens.

3. Click the **PLM XML** tab.

4. In the **Target Application** box, select the transfer mode that you want to use for exporting your data.

5. Enter a file name for your export file in the **Output File** box. Alternatively, you can browse to an existing file using the button.

6. If you work in a localized environment, select the language for which you want to export translations.

   > Note:
   >
   > Teamcenter exports approved translations only.

7. Click **OK**. Classification Admin exports all the data that you specified via the transfer mode to the specified file. You can view this file in a Web browser or in any XML editor.

If the exported object includes images, these are contained in a separate folder in the export directory. To re-import data on another machine, you must ensure that you copy the image directory along with the exported data.

Using transfer modes

Transfer modes are a collection of rules that assist you in traversing your data hierarchy. They specify which objects should be imported or exported. You can create your own transfer modes in the PLM XML module.

Classification Admin offers you a selection of transfer modes that meet most of your classification import/export needs. You can also modify these packaged transfer modes in the PLM XML module. The following table lists the transfer modes that you can use with Classification Admin and their purpose.

To use these transfer modes, ensure that you extend your data to the presentation layer.

Select the following transfer mode	To export
ICSExportParents	A class and all its parents. This includes attributes, key-LOVs, and views.
ICSExportParentsWithICOs	A class and all its parents, including all ICOs belonging to the exported classes. This includes attributes, key-LOVs, and views.
ICSExportParentsWithPartFamily	A class and all its parent classes, part family templates, and their member ICOs.
ICSExportParentsWithWSOs	A class and all its parents, including all ICOs and classified objects. This includes attributes, key-LOVs, and views.
ICSExportSubtree	A class and all its children. This includes attributes, key-LOVs, and views.
ICSExportSubtreeWithICOs	A class and all its children, including all ICOs belonging to the exported classes. This includes attributes, key-LOVs, and views.
ICSExportSubtreeWithPartFamily	A class and all its child classes, part family templates, and their member ICOs.
ICSExportSubtreeWithWSOs	A class and all its children, including all ICOs and classified objects. This includes attributes, key-LOVs, and views.

The Classification application allows you to export data using the ICO as the starting point. The following table lists the transfer modes that you can use with Classification and the information they export.

Select the following transfer mode	To export
ICSExportICOs	An ICO.
ICSExportICOs_User	An ICO with the user class to which it belongs.
ICSExportICOs_Admin	An ICO with its **admin** class, views, parents, attributes, and key-LOVs.
ConfiguredDataExportDefault	An ICO with the user class to which it belongs, in addition to the item it classifies (product data).

To export an ICO with its **admin** class, views, parents, dictionary attributes, and key-LOVs, as well as the object it classifies (product data), you must modify the **ConfiguredDataExportDefault** transfer mode.

> Note:
>
> If you work in a localized environment, Teamcenter exports approved translations only.

The Classification application allows you to import data using the **incremental import** or **incremental_import** transfer mode. The **incremental_import** transfer mode (with an underscore in place of a space) is used on Linux platforms. Either transfer mode can be used on Windows.

Modify the ConfiguredDataExportDefault transfer mode

To export an ICO with its **admin** class, views, parents, dictionary attributes, and key-LOVs, as well as the object it classifies (product data), you must modify the **ConfiguredDataExportDefault** transfer mode as follows:

1. In the PLM XML application, create a copy of the **ConfiguredDataExportDefault** transfer mode.

2. In the copied transfer mode, find the following closure rule clauses.

Primary Obj...	Primary Object	Secondary Obj...	Secondary Object	Relation Type	Related Property Or Object	Action Type
CLASS	icm0	CLASS	smlb0	PROPERTY	ICSUserClass	TRAVERSE_AND_PROCESS
CLASS	icm0	CLASS	bldb0	PROPERTY	ICSUserClass	TRAVERSE_AND_PROCESS

3. Modify the clauses as follows.

Primary Obj...	Primary Object	Secondary Obj...	Secondary Object	Relation Type	Related Property Or Object	Action Type
CLASS	icm0	CLASS	smlb0	PROPERTY	ICSAdminClass	TRAVERSE_AND_PROCESS
CLASS	icm0	CLASS	bldb0	PROPERTY	ICSAdminClass	TRAVERSE_AND_PROCESS

4. Add the following new clauses.

Primary Obj...	Primary Object	Secondary Obj...	Secondary Object	Relation Type	Related Property Or Object	Action Type
CLASS	smlb0	CLASS	smlb0	PROPERTY	ICSParent	TRAVERSE_AND_PROCESS
CLASS	smlb0	CLASS	bldb0	PROPERTY	ICSView	TRAVERSE_AND_PROCESS
CLASS	smlb0	CLASS	unct_dict	PROPERTY	ICSAttributes	TRAVERSE_AND_PROCESS
CLASS	unct_dict	Class	stxt	PROPERTY	ICSKeyLOV	TRAVERSE_AND_PROCESS

5. Modify this transfer mode to include any additional closure rules for your company-specific data.

PLM XML restrictions

- The format of input/output files must adhere to the PLM XML standard. This standard defines five special characters for structuring content. You must ensure that these characters are written using the format shown in the following table.

Character	PLM XML acceptable format
&	**&**
<	**<**
>	**>**
'	**'**
"	**&quote;**

- When you import a classification object (**dictionary** attribute, **key-LOV**, **class**, **view**, or **ICO**) using PLM XML and the object already exists in the database, the import action updates existing objects in the database.

  The Classification import functionality ignores the incremental change flag on transfer modes and updates existing classification objects.

Migrating SML subclasses to storage classes

Legacy SML subclass migration

If you no longer want to use legacy SML subclasses, you can migrate these subclasses to abstract classes and storage classes. This process does not remove the SML class or subclass. It performs the following:

- Creates a new abstract class for each SML class.

- Creates a storage class directly below the new abstract class for each subclass belonging to the SML class that is being migrated.

The new abstract class has the same parent class as the SML class and contains the attributes defined locally in the SML class and used in all SML subclasses below the SML class.

The new storage class for an SML subclass contains attributes that are not common to all classes. A default view is created for the new storage class when there are settings in the SML subclass that cannot be copied to the storage class. This is the case when one of the following occurs:

- In the SML subclass, the attribute order of inherited attributes differs from the one defined by the parent classes.
- For at least one inherited attribute, at least one of the required, protected, unique, array size properties differ from the property set in the class where the attribute is defined.
- For at least one attribute, there are layout settings in the SML subclass.

Migrate classes and subclasses

1. Right-click the SML class in the classification hierarchy that you want to migrate and choose **Migrate SubClasses**.

   The **Migrate Subclasses** dialog box displays the new target class hierarchy.

2. Enter a new ID into the **New Class ID** box. If you leave this box empty, the system automatically generates a new ID based on the value you enter in the **ICS_migrate_classid_pattern** preference.

3. (Optional) Enter a new storage class ID in the **New Storage Class ID** box. If you do not, the system assigns one for you.

4. Click **OK**.

   The new class and storage class appear in the hierarchy under the same parent as the original SML class.

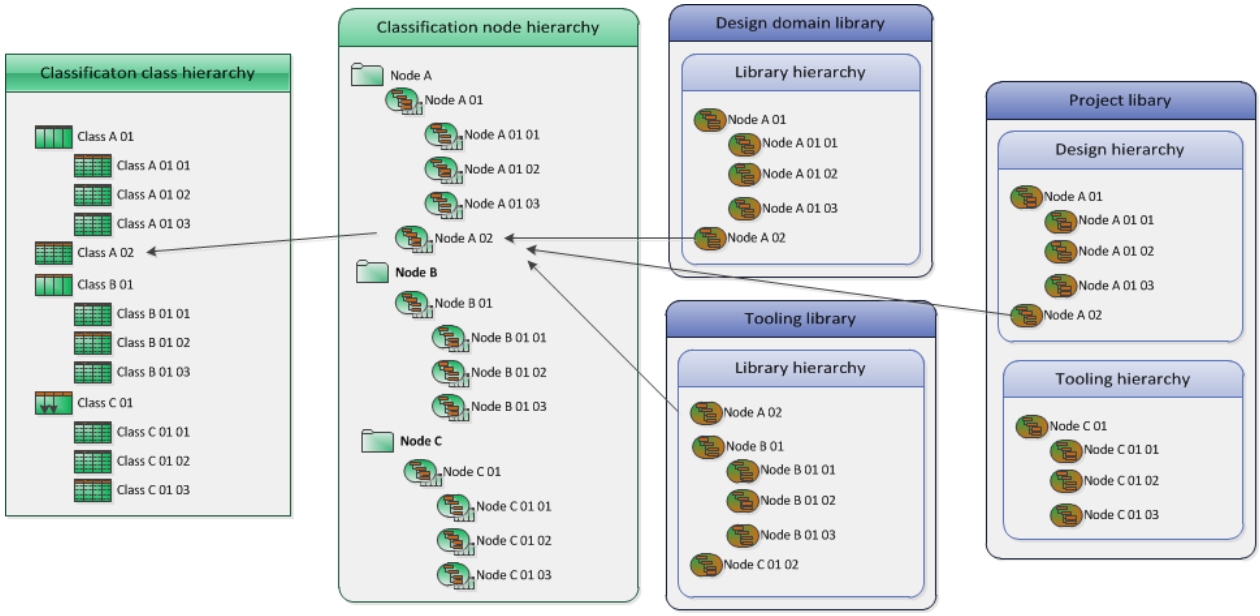
Administering classification libraries

Library management overview

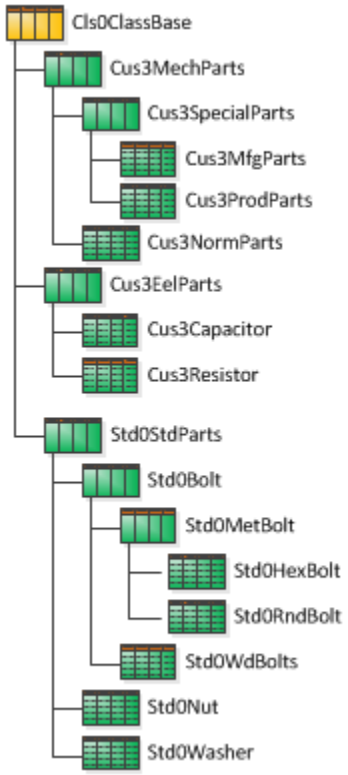
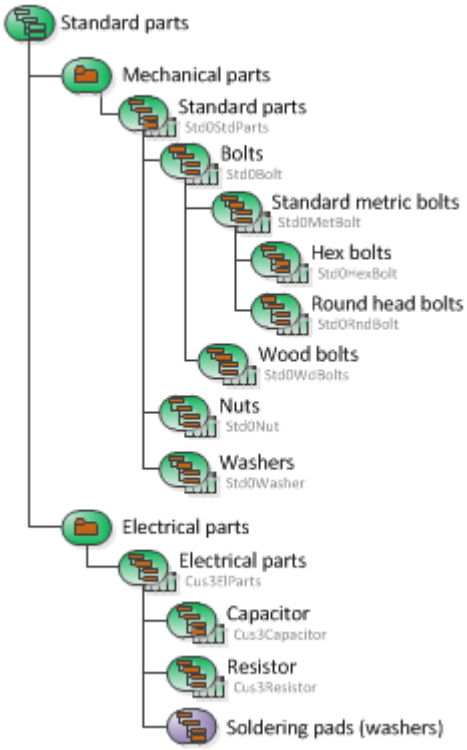
Libraries are used to organize classified objects with similar characteristics or application in a hierarchy. They reflect the business use of the objects they contain. For example, if you were looking for a bolt, you would start by searching for the bolt library. Once you find the bolt library, you can perform additional searches, use filters, or browse through the hierarchy to find the object that meets your criteria.

The class hierarchy is built based upon attribute inheritance. This does not always reflect the business use of the object for which you search. Libraries are based on a *presentation hierarchy* and provide you with a different view on the data. The presentation hierarchy is built on top of the traditional classification hierarchy and mirrors it. However, because it is built from workspace objects, it supports the flexibility of libraries. A library contains library elements that are used to manage library objects and, ultimately, reference classification classes. A classifying node links to a storage class. Using an optional

synchronization mechanism, you can ensure that any actions performed on the traditional classification hierarchy are also performed on the presentation hierarchy.



A presentation hierarchy can have its own symbols and images but its structure is a one-to-one reflection of the classification storage hierarchy. Classification nodes in the presentation hierarchy refer back to classification classes in the storage hierarchy.

Data that is stored in a storage hierarchy...	... is displayed in a presentation hierarchy
	

There is no user interface for classification libraries. You can display presentation layer nodes in My Teamcenter and search for data filtered by libraries in Active Workspace. Libraries are administered using the **clsutility** and **lbrmanger** utilities.

Membership rules are used to populate a library on demand. A membership rule tells Teamcenter to search through a part of a presentation hierarchy and find all elements that have a particular characteristic, and then create a library element for the node element. For example, select all classified bolts from the **Bolts** class that are made of titanium and whose diameter is greater than 20. You can create multiple rules for a particular library node, and you can search either presentation hierarchies or other libraries to populate a new library.

About specifications

Specifications are design or component selection guidelines set up by expert users for a particular domain or discipline. Using specifications provides a rule-based configuration that guides you in finding only those components suitable to a particular design purpose, for example, *Find a pump suitable for a high pressure oil pipe*. The specifications are typically applied while searching the library using specification information as additional input. Specifications are created in the context of a library and can be associated to one or more libraries. They are revisable.

In the following scenario, a worker uses a specification based search to insert a valve and all required supporting parts in a length of pipe. The known characteristics are as follows:

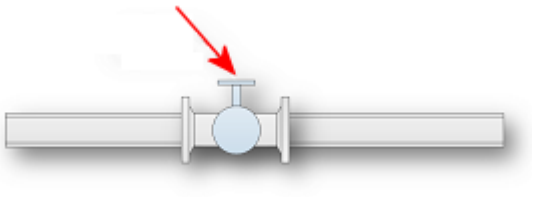
- Nominal pipe size (NPS) = **4**
- Component type = **valve**

The worker proceeds as follows:

1. Select the target (pipe), component type to insert (valve), and the predefined specification.

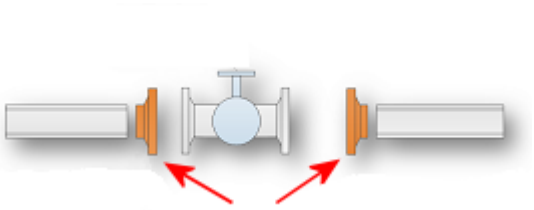
2. Find the main component to be inserted.

   Based on the specification rules, Teamcenter finds a valve.



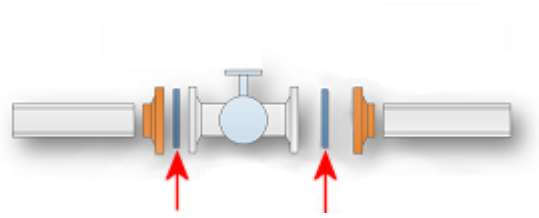
3. Find additional parts required to connect the component.

   Teamcenter adds flanges based on postplacement rules.



4. Find spare parts to complete the connections.

   Based on generic postplacement rules for flange/flange connections, Teamcenter finds and adds two gaskets.



Types of objects you work with

When creating libraries, you work with the following objects:

- Library: container
  - Hierarchies
    - Hierarchy nodes
      - ◊ Library elements
      - ◊ Membership rules
- Specifications
  - Specification rules

Library	Classification data that is filtered to suit a particular business need.
Hierarchy	Structures within a library that organize their constituent library elements. A library can contain multiple hierarchies. A hierarchy (also referred to as a presentation hierarchy) refers back to classes in the classification storage hierarchy.
Node	A representation of a classification storage class within a library hierarchy.
Membership rule	Rule that provides an automated way to populate a library with library elements. Rules identify the objects that should become members of a library and are specific to a particular library node. They can be evaluated at any time to update a library. For example: Rule A: Select all classified bolts from where material is titanium. Rule B: Select all bolts from the vendor catalog where material is titanium and diameter greater than 20.
Specification	Design or component selection guidelines set up by expert users for a particular domain or discipline to assist you in finding components suitable to a particular design purpose.

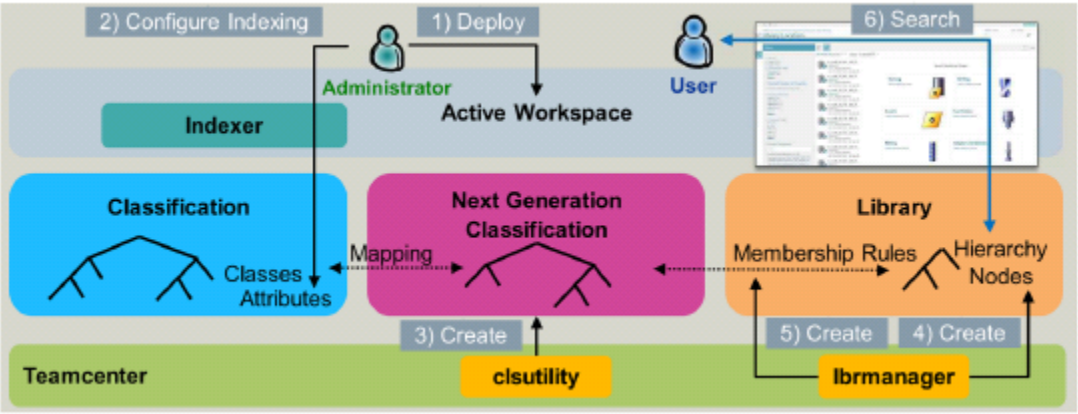
Install classification libraries

Prerequisites:

- The Classification Teamcenter Classification Collector (earlier called Library Management) feature must be installed and configured on Teamcenter.
- The presentation layer must be created with the **clsutility**.
- The library data must be created with the **lbrmanager** utility.

For more information, see how to use Library Management to selectively display the classification hierarchy in *Advanced Classification — Deployment and Administration*.

The following displays the interaction of the various components necessary to use Classification in Active Workspace:



- Server
  - To install the server components for classifying objects and searching for classified objects:
  - To install the server components necessary for enabling visual navigation cards:
  - To install the server components necessary for searching within classification libraries:

Functions	Server features		
	Classification	Next Generation Classification (Presentation layer)	Teamcenter Classification Collector
Index classification attributes	√	√	√
Index classification classes	√	√	√

Functions	Server features		
	Classification	Next Generation Classification (Presentation layer)	Teamcenter Classification Collector
Index classifying objects (ICOs)	√	√	
Hierarchical filtering of classes	√	√	√
Index catalog data			√
Index library elements			√
Visual navigation cards (VNC) for classes		√	√
Visual navigation cards (VNC) for library nodes			√
Compatible client feature	Classification client	Teamcenter Classification Collector client	

- Client
  - To install the user interface components necessary for using classification libraries:

Functions	Client features	
	Client features	Teamcenter Classification Collector
Classification authoring	√	
Browse classification hierarchy using visual navigation cards		√
Browse library hierarchy using visual navigation cards		√
Dedicated location for searching and browsing		√

Using the clsutility and the lbrmanager utility

Libraries are administered using two command line utilities:

- **clsutility**

  This utility is used to create and update the presentation layer based on the classification hierarchy.

- **lbrmanager**

  This utility is used to create, display, publish, and retract library objects.

These utilities contain many arguments and subarguments. The instructions about how to use the utilities is embedded in the utilities themselves. To obtain help for the utilities, enter:

utility-name **-h**

For example:

lbrmanager -h

To obtain help about a subargument, enter:

utility-name -subargument **-h**

For example:

clsutility -include_instances -h

These utilities are installed when you **install the library feature**.

Install and configure library management

Install library management using Teamcenter Environment Manager.

- Select the **Teamcenter Classification Collector** option from the **Reuse and Standardization** section of the installation menu.

  This step installs the following features:

  - The **clsutility** utility required to create the presentation layer.
  - The **lbrmanager** utility required to administer libraries.
  - **The specifications feature**

  Additionally, it installs the following prerequisites:

  - Advanced PLM Services for Applications
  - Advanced PLM Services for Partitioning
  - Advanced PLM Services for Realization
  - Next Generation Classification foundation (the presentation layer)

> Note:
>
> Installing library management does not affect the data dictionaries feature in Classification.

Extend classes to the presentation layer using clsutility

1. Extend classification data to the presentation layer by running the following command line utility:

   clsutility **-import -hierarchy -cid=***group-or-class-ID*

   This command extends the classification subhierarchy under the specified group or class.

   - For classification groups, running this utility creates presentation layer group nodes with the same ID as the associated groups.
   - For classification classes, running this utility creates presentation layer master nodes with the **storage_class_type** property corresponding to their associated classes.

   Additional optional arguments:

   - **-include_instances**

     Includes ICOs in the subhierarchy.

   - **-exclude_children**

     Excludes the subhierarchy. It extends the specified group or class only.

2. (Optional) **Synchronize the presentation layer with the classification hierarchy.**

Extend classes to the presentation layer when using only a subset of the classification hierarchy

When using only a subset of the classification hierarchy, you must run the **clsutility** on the entire hierarchy. In this case, the argument **-include_instances** is not required since you only need to extend the classes to the nodes that are required.

> Example:
>
> **clsutility -import -hierarchy -cid=ICM**

Synchronize the presentation layer with the classification hierarchy

Use one of the following methods to synchronize the presentation layer with the underlying classification hierarchy:

- **Manually run clsutility.**

- Enable automatic synchronization that shadows specific operations by setting the **CLS_auto_sync_node_hierarchy** preference to **true**.

  When at least the top-level node of the storage and presentation hierarchies are **linked**, the synchronization mechanism is triggered by a change to groups, classes, or ICOs in the storage hierarchy and results in the following changes to the presentation hierarchy.

Operation on class hierarchy	Operation required on node hierarchy
Add/delete group	Add/delete group node
Add/delete class	Add/delete master node
Add/delete class ICO	Add/delete classification element
Copy/paste or cut/paste of a group	Copy/paste or cut/paste of a group node
Copy/paste or cut/paste of a class	Copy/paste or cut/paste of a master node

  The synchronization mechanism functions as follows:

  - If a target is not found in the presentation hierarchy, it is created (applies to node or element).
    - To create a node in the presentation hierarchy, Teamcenter matches the parent node with the parent class.
    - To create an element in the presentation node, Teamcenter matches the node with the storage class.
  - To find targets, Teamcenter searches for the following:
    - A group node with the same ID as the classification group
    - A master node with the storage class type property pointing to the associated classification class
    - A classification element referencing the associated ICO

    The objects in the hierarchies are mapped as follows.

Object types from Classification (class hierarchy)	Corresponding object types from the presentation layer (node hierarchy)
Group	Group node
Abstract class	Master node
Storage class	Master node
Classification object (ICO)	Classification element (ICO)

Create library data using the lbrmanager utility

The following examples demonstrate how to create, display, publish, and retract library objects.

> Note:
>
> To obtain more information about any of these commands, type:
>
> *command-name -sub-command* **-h**
>
> For example:
>
> **lbrmanager -create -library -h**

Create a standalone library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -library -id=***MechanicalPartsLibrary* **-name=***MechanicalPartsLibrary* **-descr="***MechanicalParts Library***" -type=***Domain* **-disciplines="***Mechanical,Electrical***"**

To display a library:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -libraries -id=***MechanicalPartsLibrary*

Create a hierarchy

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -hierarchy -id=***FixturesHierarchy* **-name=***FixturesHierarchy* **-descr="***Fixtures Hierarchy***" -libraryId=***MechanicalPartsLibrary*

To display a hierarchy:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -hierarchies -libraryId=***MechanicalPartsLibrary* **-id=***FixturesHierarchy*

Create a library node

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -node -id=***Assemblies* **-name=***Assemblies* **-descr="***All Assemblies***" -libraryId=***MechanicalPartsLibrary* **-hierarchyId=***FixturesHierarchy*

To display a node:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* *-id=Assemblies*

Create a child library node

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -node -id=***BasePlate* **-name=***BasePlate* **-descr="***Base Plate***" -libraryId=***MechanicalPartsLibrary* **-hierarchyId=***FixturesHierarchy* **-parentNodeId=***Assemblies*

To display a node:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* **-id=***BasePlate*

Display the previously created library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -libraries -id=***MechanicalPartsLibrary*

Output	Description
`[MechanicalPartsLibrary] MechanicalPartsLibrary`	Library created in *Create a standalone library*.
`|-[FixturesHierarchy] FixturesHierarchy`	Hierarchy created in *Create a hierarchy*
`|-[Assemblies] Assemblies`	Node created in *Create a library node*
`|-[BasePlate] BasePlate`	Child node created in *Create a child library node*

Publish an item to a library

Prerequisite data:

- Two items: **BasePlateItem1**, **BasePlateItem2**
- A text file, **BasePlate.txt**, with the following content:

  -type=Item -mfkPropNames=item_id -item_id=BasePlateItem1

  -elemPropNames=lbr0ElementId -lbr0ElementId=BasePlateLE01

-type=Item -mfkPropNames=item_id -item_id=BasePlateItem2

-elemPropNames=lbr0ElementId -lbr0ElementId=BasePlateLE02

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-publish -objectsFromFile -libraryId=***MechanicalPartsLibrary* **-nodeId=***BasePlate* **-inputFile=***BasePlate.txt*

To display a library with a library element:

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-show -nodes -libraryId=***MechanicalPartsLibrary* **-id=***BasePlate*

Output	Description
`[MechanicalPartsLibrary] MechanicalPartsLibrary`	Library created in *Create a standalone library*.
`|-[FixturesHierarchy] FixturesHierarchy`	Hierarchy created in *Create a hierarchy*
`|-[Assemblies] Assemblies`	Node created in *Create a library node*
`|-[BasePlate] BasePlate`	Child node created in *Create a child library node*
`->[BasePlateLE01] BasePlateItem1`	Element created in *Publish an item to a library*
`->[BasePlateLE02] BasePlateItem2`	Element created in *Publish an item to a library*

Retract an item from a library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-retract -byElementkey -libraryId=***MechanicalPartsLibrary* **-nodeId=***BasePlate* **-elementIds=***BasePlateLE01*

Setup membership rules

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-create -memberRule -name=***TestRule* **-nodeId=***BasePlate* **-libraryId=***MechanicalPartsLibrary* **-propName=***sml01001* **-sml01001=”***steel***”**

Evaluate membership rules to populate a library

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-evaluate -memberRules -libraryId=***MechanicalPartsLibrary*

Instantiate a library element into a collaborative design

Prerequisite data:

- A collaborative design with ID **DesignID01**

lbrmanager -u=*user-ID* **-p=***password* **-g=***group* **-instantiate -test -designId=***DesignID01*-
libraryId=*MechanicalPartsLibrary* **-elementId=***BasePlateLE02*