

Teamcenter[®] Application and Data Model Administration

**Student Guide
October 2007
MT25460 – Version 2007**

Proprietary and restricted rights notice

This software and related documentation are proprietary to Siemens Product Lifecycle Management Software Inc.

© 2007 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Contents

| | |
|---|------------|
| Introduction | 13 |
| Course overview | 15 |
| Course objectives | 16 |
| Key benefits | 17 |
| Prerequisites | 17 |
| Audience | 18 |
| Learning tracks | 18 |
| Training materials provided | 19 |
| Introduction to Help | 20 |
| Introduction to administration | 1-1 |
| Administrative interface | 1-2 |
| Rich client interface | 1-3 |
| Rich client administrative applications | 1-4 |
| Activity | 1-5 |
| Activity: Log on to your computer and start the virtual image | 1-6 |
| Activity: Open the workbook | 1-7 |
| Activity: Stop and restart the virtual image, if applicable | 1-8 |
| Activity: Verify TCFS is started | 1-9 |
| Activity | 1-10 |
| Business Modeler IDE administration tasks | 1-11 |
| What is the data model | 1-12 |
| Basic item structure | 1-13 |
| Business objects and classes | 1-14 |
| Teamcenter POM schema | 1-15 |
| Lists of values | 1-16 |
| Options | 1-17 |
| Constants | 1-18 |
| Rules | 1-19 |
| Templates | 1-21 |
| Troubleshooting | 1-22 |
| Rich client administration tasks | 1-23 |
| Organization | 1-24 |
| Preferences | 1-25 |
| Query Builder | 1-26 |
| Report Builder | 1-27 |
| Access Manager | 1-28 |

| | |
|---|------------|
| Project | 1-29 |
| Workflow Designer | 1-30 |
| Command Suppression | 1-31 |
| Subscription Monitor | 1-32 |
| Activity | 1-33 |
| Teamcenter architecture overview | 1-34 |
| Two-tier architecture logical view | 1-35 |
| Four-tier architecture logical view | 1-36 |
| Teamcenter environment | 1-38 |
| Command line scripts | 1-39 |
| Summary | 1-40 |
| Introduction to organization | 2-1 |
| Defining the organization | 2-2 |
| What is a volume | 2-3 |
| Create an account | 2-4 |
| Organization interface | 2-5 |
| Create persons | 2-6 |
| Create users | 2-7 |
| Activity | 2-8 |
| Summary | 2-9 |
| Introduction to the Business Modeler IDE | 3-1 |
| What is the Business Modeler IDE | 3-2 |
| What are the Business Modeler IDE and Eclipse | 3-3 |
| Business Modeler IDE user interface | 3-4 |
| Business Modeler IDE perspective | 3-5 |
| Extensions perspective | 3-6 |
| Using the Eclipse framework | 3-7 |
| Enabling the Business Modeler IDE | 3-8 |
| Installing the Business Modeler IDE as a stand-alone application | 3-9 |
| Start the IMR | 3-10 |
| Start the Business Modeler IDE | 3-11 |
| Activity | 3-13 |
| Configuring the Business Modeler IDE | 3-14 |
| Creating a project | 3-15 |
| Project files | 3-16 |
| Create a Business Modeler IDE template project | 3-18 |
| Server profile | 3-20 |
| Add a server profile | 3-21 |
| Two-tier server connection profile | 3-24 |
| Activity | 3-25 |
| Extending the data model | 3-26 |
| Introduction to templates | 3-27 |

| | |
|--|------------|
| Basic Business Modeler IDE process | 3-28 |
| What is the basic Business Modeler IDE process | 3-29 |
| Set the active extension file | 3-30 |
| Save your changes | 3-32 |
| Deploy extensions | 3-33 |
| Deploy extensions to a test server | 3-34 |
| Activity | 3-37 |
| Summary | 3-38 |
| Data Model | 4-1 |
| Defining the data model | 4-2 |
| Extending items | 4-3 |
| New data model example | 4-4 |
| Defining business objects | 4-5 |
| What are business objects | 4-6 |
| Find business objects | 4-7 |
| Creating item business objects | 4-8 |
| Create an item business object | 4-10 |
| What are business object properties | 4-13 |
| Business objects properties table | 4-14 |
| Activity | 4-18 |
| Defining classes | 4-20 |
| What are classes | 4-21 |
| Find classes | 4-22 |
| Adding new classes | 4-23 |
| Add a new class | 4-24 |
| What are class attributes | 4-27 |
| Class attributes table | 4-28 |
| Add or change attributes on custom classes | 4-31 |
| Activity | 4-33 |
| Defining forms | 4-34 |
| Create a form business object | 4-35 |
| Storage class form | 4-37 |
| Add, change, or remove properties on a form business objects ... | 4-38 |
| Hide properties on a form business object | 4-39 |
| Activity | 4-40 |
| Summary | 4-41 |
| UML editor | 5-1 |
| Defining the UML editor | 5-2 |
| What is a UML diagram | 5-3 |
| Data model inheritance | 5-4 |
| Set preferences for the UML editor | 5-5 |
| Managing the data model using the UML editor | 5-6 |
| Opening a business object in the UML editor | 5-7 |

| | |
|--|------------|
| Open a class or business object in the UML editor | 5-8 |
| Create a new UML diagram | 5-10 |
| Create a new class or business object | 5-11 |
| Activity | 5-12 |
| Summary | 5-14 |
| Lists of values | 6-1 |
| Defining list of values (LOV) | 6-2 |
| List of values (LOV) interface | 6-3 |
| Add a list of values (LOV) | 6-5 |
| Attach an LOV to a property | 6-9 |
| Add, remove, or clear a value to an LOV | 6-11 |
| LOV variations | 6-12 |
| Create a filter LOV | 6-14 |
| Creating a cascading LOV | 6-15 |
| Create a Cascading LOV with an interdependent attachment . . . | 6-17 |
| Create a cascading LOV | 6-18 |
| Activity | 6-19 |
| Summary | 6-21 |
| Datasets | 7-1 |
| What are dataset business objects | 7-2 |
| What is a tool | 7-3 |
| Add a tool | 7-4 |
| Create a dataset business object | 7-6 |
| Dataset and named references | 7-9 |
| Using named references | 7-10 |
| import_file utility | 7-11 |
| Activity | 7-13 |
| Summary | 7-15 |
| Options | 8-1 |
| What are options | 8-2 |
| What is a note | 8-3 |
| Add a note | 8-4 |
| What is a status | 8-6 |
| Add a status | 8-7 |
| What is a unit of measure | 8-9 |
| Add a unit of measure | 8-11 |
| Activity | 8-13 |
| What is a view | 8-15 |
| Add a view | 8-16 |
| What is a change | 8-18 |
| Add a change | 8-20 |
| Activity | 8-24 |

| | |
|--|-------------|
| Summary | 8-25 |
| Constants | 9-1 |
| Defining constants | 9-2 |
| Constants framework | 9-3 |
| Constants precedence | 9-4 |
| Global constants | 9-5 |
| Create a global constant | 9-6 |
| Business object constants | 9-9 |
| Create a business object constant | 9-10 |
| Property constants | 9-13 |
| Property constants reference | 9-14 |
| Changing a property constant value | 9-16 |
| Create a property constant | 9-17 |
| Activity | 9-21 |
| Summary | 9-22 |
| Rules | 10-1 |
| Defining rules | 10-2 |
| What is a business object display rule | 10-4 |
| Create a business object display rule | 10-5 |
| What is a GRM rule | 10-7 |
| Add a GRM rule | 10-9 |
| Activity | 10-11 |
| What is a naming rule | 10-12 |
| Create and attach a naming rule | 10-13 |
| Add a naming rule | 10-14 |
| Attach a naming rule to a business object property | 10-16 |
| Naming rule examples | 10-18 |
| What is an ID context rule | 10-19 |
| What is a compound property rule | 10-20 |
| Defining a compound property rule path | 10-21 |
| Add a compound property rule | 10-22 |
| Activity | 10-25 |
| What is a deep copy rule | 10-26 |
| What is the deep copy hierarchy | 10-27 |
| Add a deep copy rule | 10-28 |
| What are extension rules | 10-31 |
| Predefined extension definitions | 10-32 |
| Use a predefined extension definition | 10-34 |
| Activity | 10-36 |
| Summary | 10-37 |
| Data model files | 11-1 |
| Defining data model files | 11-2 |

| | |
|--|-------------|
| Package extensions into a solution template | 11-3 |
| Adding extensions to the template | 11-5 |
| Installing a custom solution template | 11-6 |
| Import a Business Modeler IDE template project | 11-7 |
| Import a model file | 11-9 |
| Activity | 11-10 |
| Summary | 11-11 |
| Organization hierarchy | 12-1 |
| Defining the organization | 12-2 |
| What is a person | 12-3 |
| What is a user | 12-4 |
| What is a group and group member | 12-5 |
| What is a subgroup | 12-6 |
| Group hierarchies | 12-7 |
| What is a role | 12-8 |
| What is a discipline | 12-9 |
| Passwords | 12-10 |
| What is a volume | 12-11 |
| Activity | 12-12 |
| Define administrative privileges | 12-14 |
| System administration accounts | 12-15 |
| Create an account | 12-17 |
| Organization interface | 12-18 |
| Create the organization structure | 12-19 |
| Create groups | 12-20 |
| Create persons | 12-21 |
| Create users | 12-22 |
| Apply a new role to a group | 12-23 |
| Apply new users to a group/role | 12-24 |
| Group terms and concepts | 12-25 |
| Activity | 12-27 |
| Manual account generation | 12-28 |
| make_user examples | 12-29 |
| Activity | 12-30 |
| Import and export organization | 12-31 |
| Export organization | 12-32 |
| Import organization | 12-33 |
| Search the organization | 12-34 |
| Search the Organization tree | 12-35 |
| Activity | 12-36 |
| Summary | 12-37 |
| Query Builder definitions | 13-1 |
| Define query definitions | 13-2 |

| | |
|--|-------------|
| Query Builder interface | 13-3 |
| Properties for workspace objects | 13-8 |
| Query Builder creation features | 13-9 |
| Create a query based on an existing query definition | 13-10 |
| Activity | 13-11 |
| Create new query definitions | 13-13 |
| Using search criteria clauses | 13-15 |
| Activity | 13-19 |
| Limit access to query definitions | 13-21 |
| Access control list | 13-22 |
| Optional activity | 13-23 |
| Custom item type query definitions | 13-24 |
| Activity | 13-26 |
| Import and export query definitions | 13-27 |
| Export query definitions | 13-28 |
| Import query definitions | 13-29 |
| Activity | 13-30 |
| Create a referenced-by query | 13-31 |
| Activity | 13-35 |
| Summary | 13-36 |
| Preference management | 14-1 |
| Introduction to preferences | 14-2 |
| Creating and editing preferences | 14-4 |
| Preferences pane | 14-5 |
| Organization pane | 14-6 |
| Preference categorization | 14-7 |
| Key Teamcenter preferences | 14-8 |
| Preference scope | 14-10 |
| Preference precedence | 14-11 |
| Activity | 14-12 |
| Generating preference reports | 14-14 |
| Create preference reports | 14-15 |
| Activity | 14-16 |
| Import and export preferences | 14-18 |
| Export preferences from the database to an XML file | 14-19 |
| Import preferences into the database | 14-20 |
| preferences_manager utility | 14-21 |
| Optional activity | 14-23 |
| Summary | 14-24 |
| Report Builder definitions | 15-1 |
| Report Builder interface | 15-2 |
| Report Builder definition types | 15-3 |
| Report definition structure | 15-4 |

| | |
|---|-------------|
| Standard Teamcenter report definitions | 15-5 |
| Activity | 15-6 |
| PLM XML report data | 15-7 |
| Teamcenter data model quick review | 15-8 |
| Creating transfer mode objects | 15-9 |
| Creating closure rules | 15-10 |
| Defining property sets | 15-11 |
| Activity | 15-15 |
| import_export_reports utility | 15-17 |
| Activity | 15-18 |
| Summary | 15-19 |
| Access Manager | 16-1 |
| Protecting Teamcenter data | 16-2 |
| Rules-based protection | 16-3 |
| Object based protection | 16-4 |
| Object ACLs | 16-5 |
| Access Manager interface | 16-6 |
| Access Manager rule tree | 16-7 |
| How rules work | 16-8 |
| Add an Access Manager rule | 16-9 |
| Create and manage named ACLs | 16-10 |
| Build an effective ACL | 16-11 |
| ACL precedence | 16-12 |
| ACL guidelines | 16-13 |
| Activity | 16-14 |
| Group security | 16-15 |
| Configuring security to prevent suppliers from viewing internal data | 16-16 |
| Configuring security for data owned by a supplier (external data) | 16-17 |
| Import and export the access manager rule tree | 16-18 |
| Import and export guidelines | 16-19 |
| Activity | 16-20 |
| Summary | 16-21 |
| Projects to control access | 17-1 |
| Define projects | 17-2 |
| Who can create projects | 17-3 |
| Project naming rules | 17-4 |
| Define team members for projects | 17-5 |
| Apply project security | 17-6 |
| Creating projects | 17-7 |
| Project interface | 17-8 |
| Project terms and concepts | 17-11 |

| | |
|---|-------------|
| Create a project | 17-12 |
| Project security rule tree | 17-13 |
| Manage projects | 17-14 |
| Assign data to projects | 17-15 |
| Automatically assign new objects to a project | 17-16 |
| autoAssignToProject availability | 17-17 |
| Configure automatic project assignment | 17-18 |
| Propagation rules | 17-19 |
| Activity | 17-20 |
| Summary | 17-21 |
| Workflow process modeling | 18-1 |
| Basic concepts about Workflow Designer | 18-2 |
| Workflow in Teamcenter | 18-3 |
| Enterprise Process Modeling | 18-5 |
| Workflow Designer interface | 18-7 |
| Releasing data with a process | 18-9 |
| Batch release utility | 18-10 |
| Activity | 18-12 |
| Defining a process model | 18-14 |
| Workflow process terms | 18-15 |
| Creating Workflow process templates | 18-16 |
| Create a generic process template | 18-17 |
| Workflow template tasks | 18-18 |
| Task and Do task templates | 18-20 |
| Review task template | 18-21 |
| Acknowledge task template | 18-22 |
| Resource Pool Subscription | 18-23 |
| Add Status task | 18-24 |
| Modifying task behavior | 18-25 |
| Controlling access in a process template | 18-26 |
| Adding task handlers | 18-28 |
| Example of useful handlers | 18-29 |
| Activity | 18-31 |
| Change Management process model | 18-34 |
| Activity | 18-36 |
| Creating baseline process templates | 18-37 |
| Optional activity | 18-38 |
| Sharing process templates | 18-39 |
| Export a process template | 18-40 |
| Import a process template | 18-41 |
| Activity | 18-42 |
| Adding Secure tasks | 18-43 |
| Activity | 18-44 |
| Adding Condition tasks | 18-45 |
| Optional activity | 18-46 |

| | |
|---|----------------|
| Deploying process templates | 18-48 |
| Activity | 18-49 |
| Summary | 18-50 |
| Course summary | 19-1 |
| Classroom system information | A-1 |
| Course agenda | B-1 |
| Student profile | C-1 |
| Course evaluation | D-1 |
| Index | Index-1 |

Introduction

Introductions

Facility overview

Course overview

Teamcenter® Application and Data Model Administration addresses configuration of the Teamcenter data model and setup of the Teamcenter environment to meet your company's needs through the Business Modeler IDE and through data and process implementation scenarios. You will learn how to configure the Business Modeler IDE to extend the data model. Data model extensions covered in this course include creating business objects, classes, options, list of values, constants, and rules. You will learn how to input business data and process models into a Teamcenter environment.

Course objectives

The overall objective for this course is to extend the data model by creating business objects, classes, options, list of values, constants, and rules and to configure the application for use by creating business data and processes.

- To understand what *administrative tasks* are done using the Business Modeler IDE and rich client interfaces.
- To configure the Business Modeler IDE.
- To understand the basic Business Modeler IDE process.
- To understand how to work with *business objects* and *classes*.
- To edit the data model graphically.
- To understand and attach a *list of values (LOV)* to a property.
- To understand how configuration *options* are used.
- To understand and configure *constants*.
- To understand what *rules* do.
- To understand how data model files are used.
- To import a data model file.
- To understand how file repository projects are used.
- To import a file repository project by using the Import File Repository interface.
- To create an *organization*.
- To configure the working environment and managing *preferences*.
- To create *saved queries*.
- To create *report definitions*.
- To configure *access permissions*.
- To configure *projects*.
- To create *workflow process templates*.

Key benefits

Key benefits for completing the course objectives include:

- Administer the data model using the Business Modeler IDE.
- Planning for the administration of data and processes.
- Configure the data model according to your company requirements.
- Import new data model configurations.
- Defining your organization and building the framework for data security.
- Increasing productivity by configuring preferences.
- Making queries available for end users.
- Configuring proper permissions for end users.
- Defining projects to give external users access to product data.
- Defining processes to manage product data development.

Prerequisites

- *Using Teamcenter* course, MT25150
- Familiarity with basic Windows operating system commands

Audience

The audience for this course includes:

| User profile | Job goal |
|---------------------------|---|
| Application administrator | Administer users, administer security, define workflows, and configure the data model |
| System administrator | Maintain servers and users |
| Database administrator | Maintain databases |

Learning tracks

Learning tracks for the Teamcenter application are found on the Siemens PLM Software training website: <http://training.ugs.com/tracks/index.shtml>

Recommended Courses

- MT25150 – *Using Teamcenter*
- MT25460 – *Teamcenter Application and Data Model Administration*

Training materials provided

| Material | Description |
|-------------------------|---|
| <i>Student Guide</i> | <p>Presentation slides.</p> <p>Yours to keep and make notes.</p> <p>Evaluation is provided both online and in the appendix.</p> <p>Student profile is provided in the appendix.</p> |
| <i>Student Workbook</i> | <p>Activities are provided online in electronic format and designed to appear on the left of the monitor.</p> <p>A CD of electronic activities is provided in the back of the <i>Student Guide</i>.</p> |

Introduction to Help

Throughout this class, you access Help topics to learn more about the product. The Help Library covers functionality from end-user tasks to customization instructions.

There are several ways to access help from Teamcenter.

- To access the Help Library, click the **Help** button or choose **Help** from the main menu.
- Press F1.
 - Current application help for the rich client

Note

You cannot access application-specific help in the Thin Client.

- Context-sensitive help for the Business Modeler IDE
- Press F2.
 - Help library for the rich client

Lesson

1 *Introduction to administration*

Purpose

The purpose of this lesson is to introduce the students to the required administration tasks for Teamcenter.

Objectives

After you complete this lesson, you should be able to:

- Define the administrative interfaces.
- List the administrative tasks.
- Understand Teamcenter architecture.

Help topics

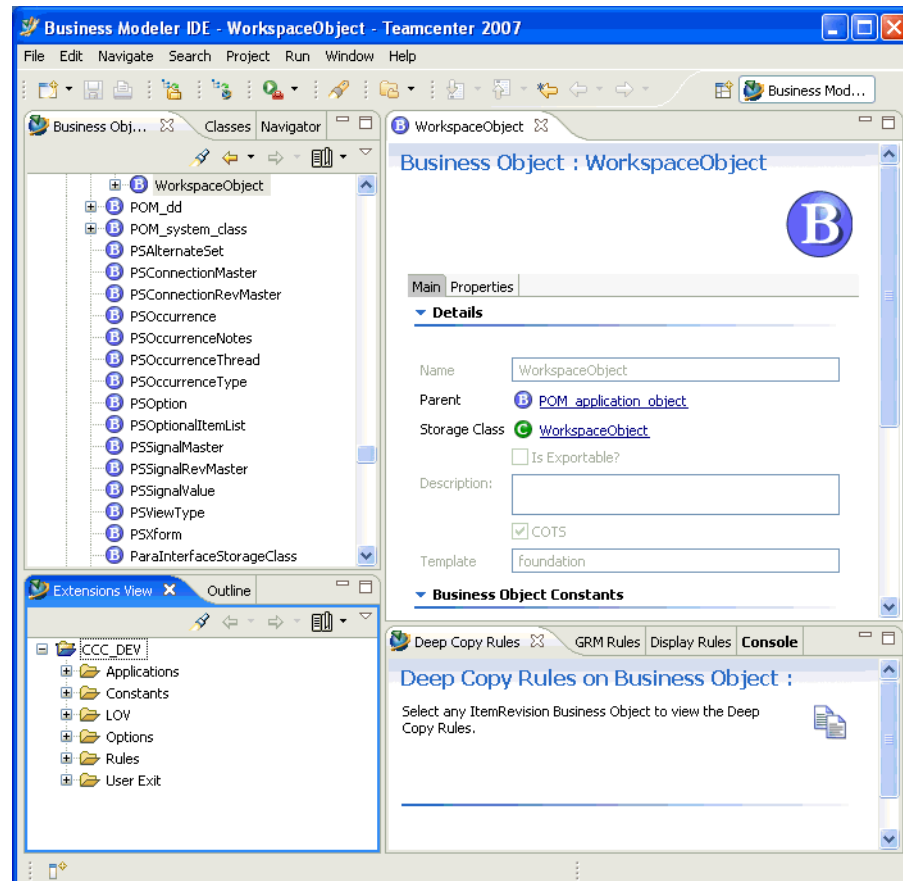
Additional information for this lesson can be found in:

- [*Getting Started with Teamcenter 2007*](#)

Administrative interface

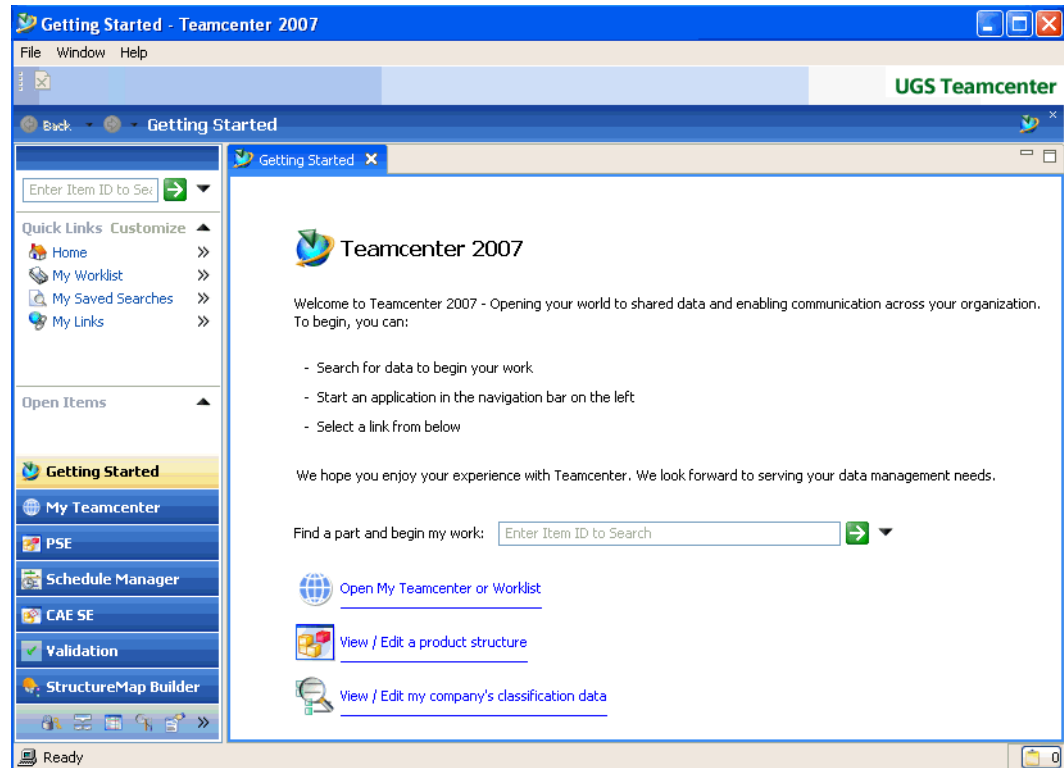
Two interfaces are used to manage administrative data:

The *Business Modeler IDE* is a tool for adding your custom data model on top of the default Teamcenter data model. The tool accomplishes this by separating the custom data model into its own set of files that are kept apart from the default data model.

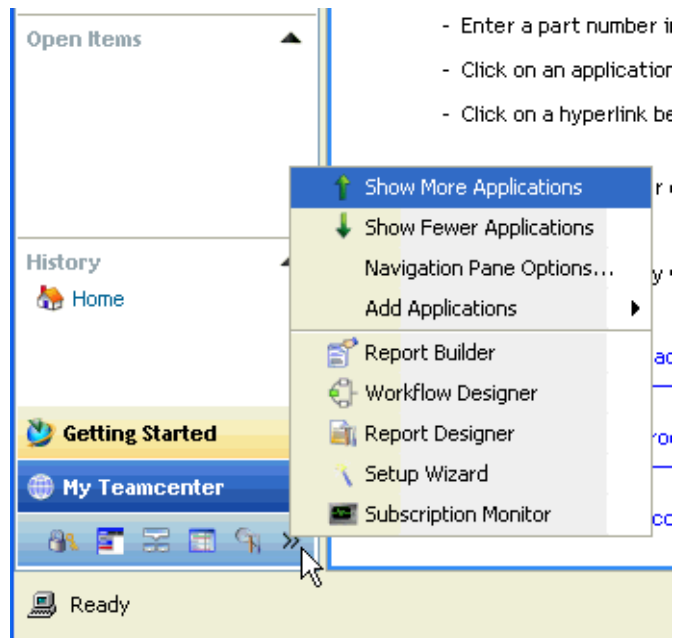


Rich client interface









The *rich client* is an interface for adding your organization, security and processes into Teamcenter. This interface is also used by the end user to manage product data.



Rich client administrative applications



Primary administrative applications:

-  Access Manager
-  Command Suppression
-  Organization
-  Project
-  Query Builder
-  Report Builder
-  Workflow Designer
-  Subscription Monitor

Activity

Perform the activities in this lesson to log on to your computer and the Teamcenter software.

Note

All activities for this lesson are included on the following pages of the student guide, except the last activity. The last activity is located in the online workbook.

- Log on to your computer.
- Start the virtual image, if applicable.
- Open the workbook.
- Stop and restart the virtual image, if applicable.
- Verify Teamcenter File Services (TCFS) is started.
- Log on to Teamcenter and view the administrative applications.

Note

This is the first activity in the online workbook.

Key points

- Not all classrooms are set up with a virtual image. Therefore, the virtual image activities in this lesson may not be applicable to your classroom.
- If your classroom is set up with a virtual image, you must start and stop the virtual image correctly to avoid corruption of the image.

Activity: Log on to your computer and start the virtual image

To complete the activities, you must log on to your classroom computer. Some classrooms are configured to run activities on a virtual image that contain the Teamcenter applications.

1. The instructor will give you a class data sheet with the user ID and password to log on to your computer.
2. Start the virtual image, if necessary.
 - Double-click the **VMware Player** shortcut on the desktop.
 - Select **Teamcenter 2007.1** under **Recent Virtual Machines**.
 - Click **Yes** if any warning messages appear like the following:
 - **Could not connect to floppy “A:”.**
 - **Virtual device floppy0 will start disconnected.**
 - Windows boots up in the virtual image. Press Ctrl-G to activate the virtual image.

Note

To switch back to the local computer, press Ctrl-Alt.

- Log on to the virtual image using the user ID and password for the virtual image on the class data sheet.

Activity: Open the workbook

Activity instructions are online and must be opened so you can follow the instructions while running Teamcenter.

1. Open the workbook.
 - Double-click the **book.html** shortcut on your desktop.
 - The instructor will show you how to position and navigate the workbook.

Activity: Stop and restart the virtual image, if applicable

In this activity, you practice the correct procedure for stopping the virtual image.

Warning

NEVER stop the virtual image by choosing **VMware Player→Exit**.

1. Stop the virtual image.
 - Choose **Start→Shut Down**.

Note

You may need to scroll down the virtual image window to see the **Start** menu.

- With **Shut down** selected, click **OK**.
2. Start the virtual image.

Follow the instructions from the previous activity to start the virtual image.
 3. Open the workbook.

Follow the instructions from the previous activity to open the workbook.

Activity: Verify TCFS is started

Before starting the Teamcenter software, verify TCFS is started.

1. Double-click the **Services** icon on the desktop.
2. Verify that TCFS is started.
 - Scroll down to find **Teamcenter Secure File Management Service**. The **Status** column should display **Started**.
 - If it is not started, right-click **Teamcenter Secure File Management Service** and select **Start**.
 - Close the **Services** window.

Activity

- Log on to Teamcenter.

Note

This is the first activity in the online workbook.

Log on to the rich client and view the administrative applications.

Business Modeler IDE administration tasks

Primary administration tasks to be performed by the Teamcenter administrator using the Business Modeler IDE:

- Configure the data model with new business objects, classes, and properties.
- Define lists of values to allow end users to pick from a list.
- Define options to configure business objects.
- Override constants to govern global system behavior.
- Define rules to govern the behaviors of business objects.
- Deploy data model updates to the client.
- Troubleshoot.

What is the data model

The *data model* is the structure in Teamcenter into which items are placed, for example, the business objects, classes, properties, and so on. Items are the fundamental object used to model data in Teamcenter. They are commonly used to identify an element of product (component, assembly, end item) or other data such as procurement specifications, test procedures, standard parts, shop tooling, engineering changes, and so forth.

Key points

Data modeled using the item object is generally revision-controlled data. All revisions of the information must be maintained, tracked, and recoverable. Data must also be modeled using the Teamcenter concept of item if it is desired to build structure of the items as in building bill of materials (BOM) for items that represent product.

In the initial setup for Teamcenter, two types of items are provided:

- **Item**

Generally used for data that represents an element of product that is CAD defined and for which product structure (BOM) data is maintained in the system.

- **Document**

Generally used for all other data that is considered revision-controlled but not necessarily considered product or is not defined using CAD applications.

Basic item structure

An *item* in Teamcenter is a structure of related objects. The basic structure of any item consists of the following minimum objects:

 **Item**

 **Item Master (Form)**

 **ItemRevision**

 **ItemRevision Master (Form)**

Key points

- **Item**

Collects data that is globally applicable to all revisions of the item.

- **Item Master (Form)**

A form object that is often used to extend the stored property data for an item to include data unique to the customer.

- **ItemRevision**

Collects data that is applicable to a single revision of the item.

- **ItemRevision Master (Form)**

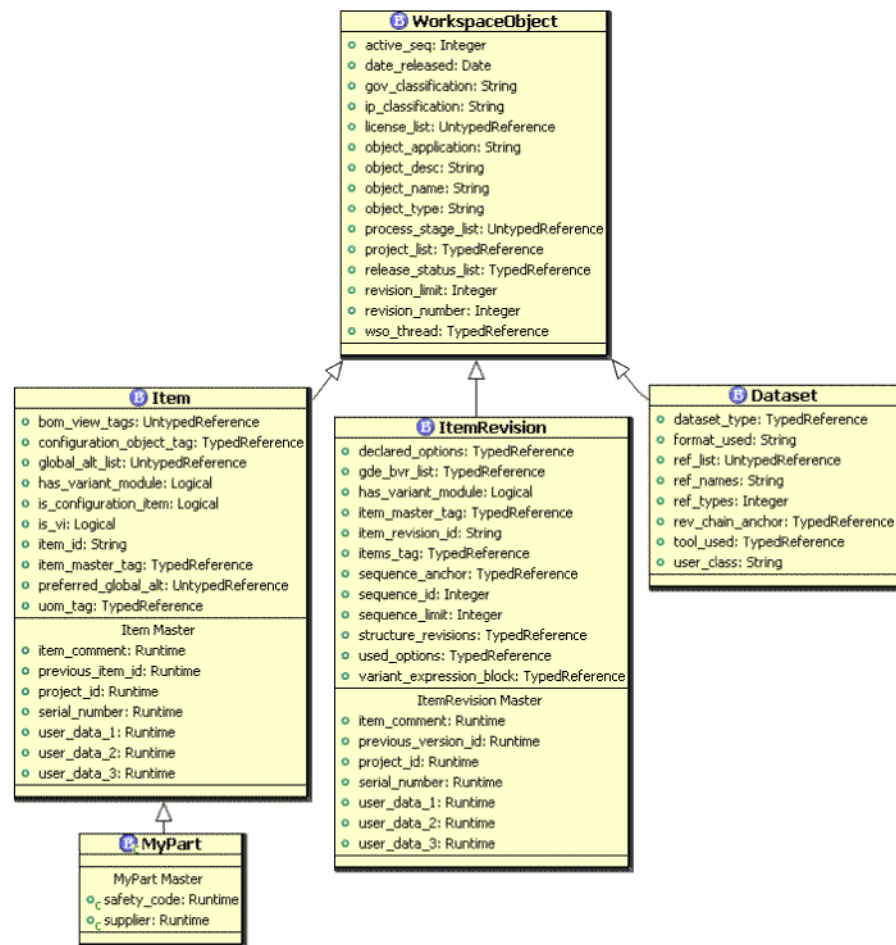
A form object that is often used to extend the stored property data for an item revision to include data unique to the customer.

Business objects and classes

Business objects are the fundamental objects used to model business data.

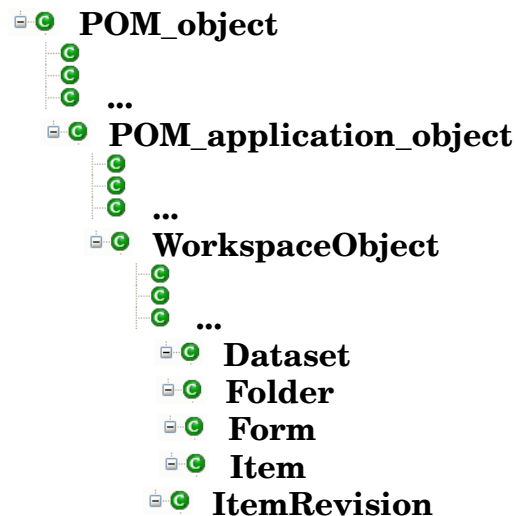
Classes are the persistent representations of the data model schema and provide properties to business objects. The class hierarchy is known as the Persistent Object Model (POM).

UML (Unified Modeling Language) is a commonly used method to graphically represent data models.



Teamcenter POM schema

The following is an abbreviated view of the Teamcenter Persistent Object Model (POM) schema.



The POM is the interface between Teamcenter objects and the Relational Database Management System (RDBMS). The persistent object manager provides definition of classes by inheritance from existing classes and definition of attributes, manipulation of in-memory objects and support for their saving and retrieval to and from the underlying RDBMS, support for applications accessing the same data concurrently, protection against the deletion of data used by more than one application, and support for the access control lists attributed to objects.

The POM_object abstract class is the superclass of all POM classes.


The POM_application_object abstract class is the superclass of all classes that are defined as part of a POM application (in this case, Teamcenter).

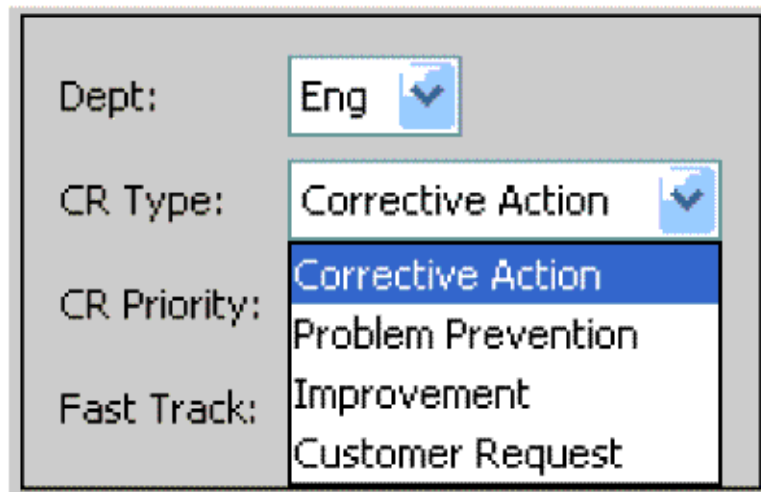
The most commonly used business objects under which you create new child business objects include:

- **Item**
Represents parts and documents.
- **Form**
Displays properties.
- **Dataset**
Represents file types.

Lists of values

Lists of values (LOV) are pick lists of data entry items. They are commonly accessed by Teamcenter users from a menu at the end of a data entry box. After you create a list of values, you must attach it to a property on a business object.

When you implement an LOV for a property, an LOV button  is displayed next to that property.

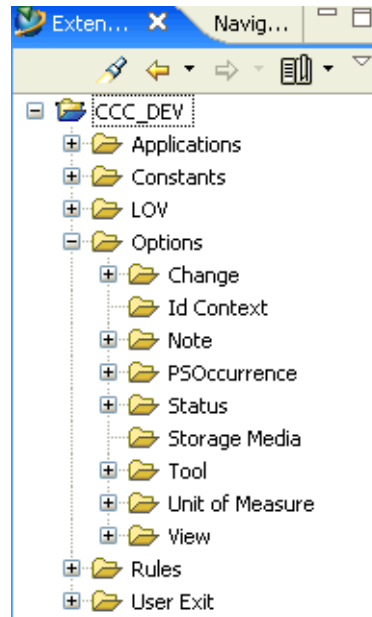


Key points

- The user simply clicks the LOV button and an object selection list displays allowable entries for that property.
- When you associate an LOV, you link this LOV to any property where that data can be entered and display an LOV to the user.

Options

Whereas business objects represent parts, documents, and other design objects, *options* represent configurations you can do to business objects. For example, a change item tracks a change to a business object, a status item designates the status of a business object, a view item holds structure information for a business object, and so on.



Constants

Constants are configuration points within the data model. They provide consistent definitions that can be used throughout the system. They can be used to set everything from user interface appearance to action behavior.

You can override an existing constant by setting a different value. Keep in mind that another template could override your setting. The last template loaded takes precedence.

There are three different kinds of constants:

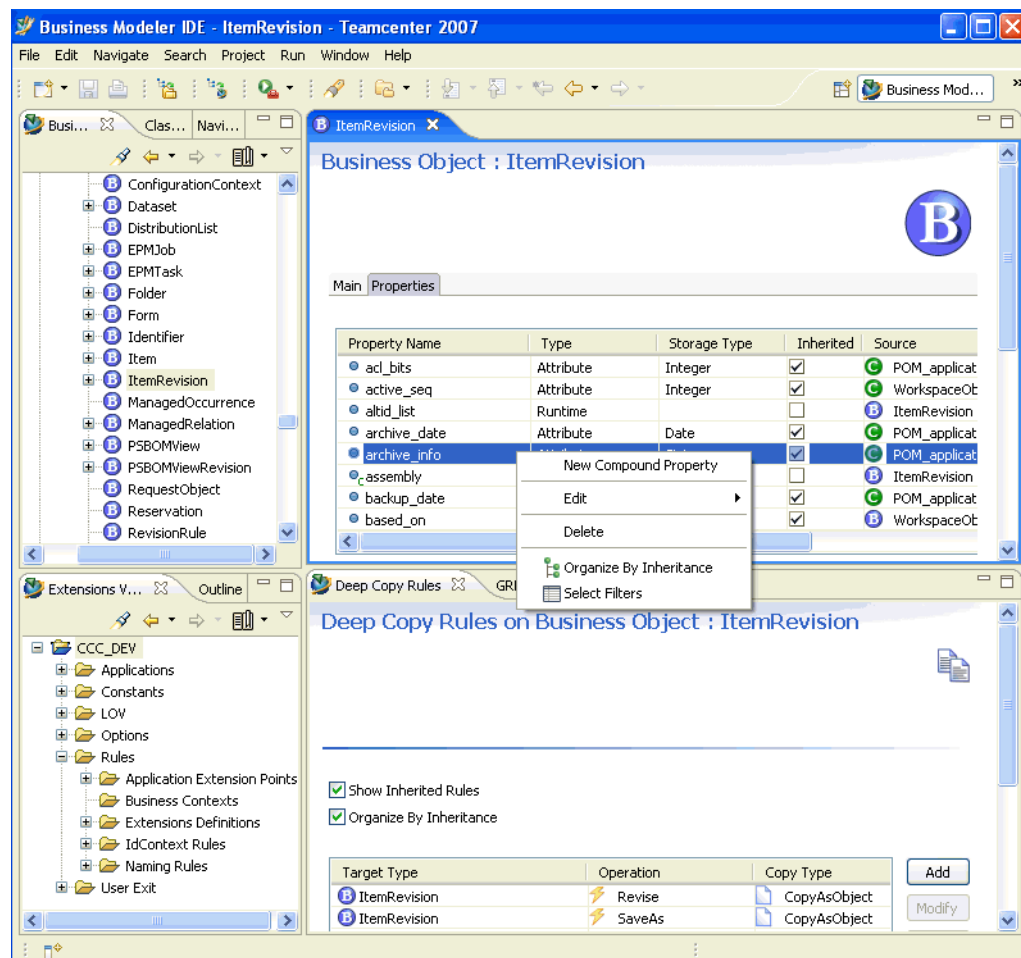
- Global constants
Provide consistent definitions that can be used throughout the system.
- Business object constants
Provide default values to business objects.
- Property constants
Provide default values to business object properties.

Rules

Rules govern the behaviors of business objects, including how they are named, what actions can be undertaken on them, and so on. Creating rules is also known as business behavior modeling.

An example of some rules are:

- Business object display rules
- Naming rules
- Compound property rules



Key points

- When a business rule is set on the parent business object and sub-business object, then the business rule set on the sub-business object takes precedence.
- When a business rule is not set on a business object, the system searches up the hierarchy for a business rule set on any parent business objects.

Example

If a naming rule, deep copy rule, or compound property rule exists for the business object, it is used; otherwise the system checks each of the business object's parents until the rule is found or the top parent is reached.

Templates

Templates are the mechanism for managing and deploying business extensions to any Teamcenter environment. You can package extensions to the data model as a solution template and distribute the template for installation to a production environment. Templates are installed using Teamcenter Environment Manager.

Templates are:

- a consistent method for defining extensions for customers and Siemens PLM Software.
- a mechanism for building and packaging solutions.
- a mechanism for protecting extensions that you share with others.
- a mechanism for concurrent development support.

Key points

- You can import a data model from projects or model files into the Business Modeler IDE.
- XML files contain business extensions like business objects, classes, LOVs, and rules.
- Dependencies can be declared so that a template cannot be installed unless the dependent templates are present in the system.










Troubleshooting

The *Digital Dashboard* is an application that system administrators can use to examine log files so they can troubleshoot problems. The Digital Dashboard works in conjunction with the Log Manager on the Teamcenter server. The Log Manager provides a log volume to hold log files, and any log file in the log volume can be imported into the Digital Dashboard. The Digital Dashboard is installed into the Business Modeler IDE and is accessed when you open the **Profiling and Logging** perspective.

The Digital Dashboard is a set of extensions to the Eclipse Test and Performance Tools Platform (TPTP) plug-ins. For more information, see *Business Modeler IDE Guide*.

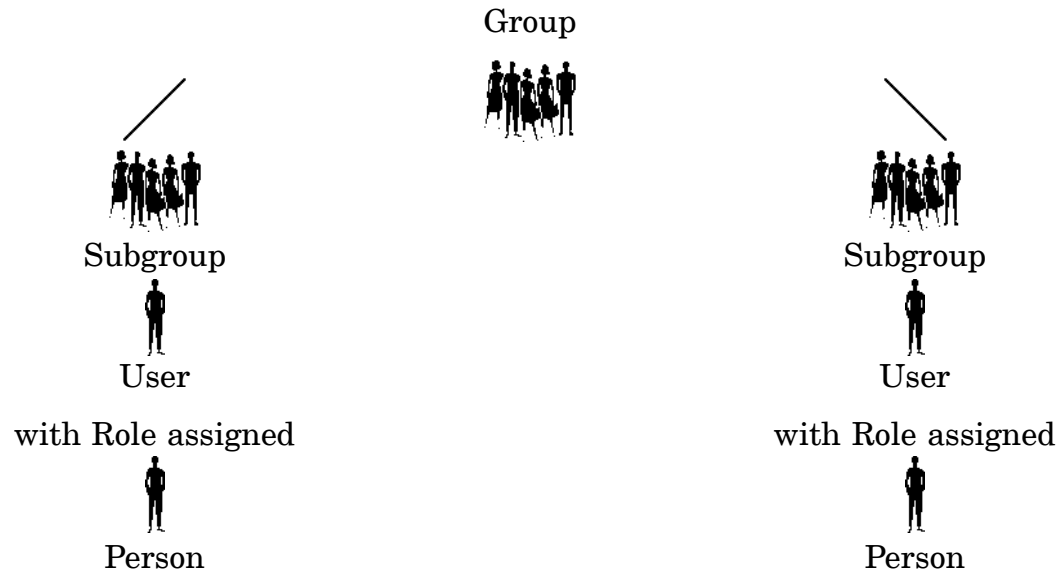
Rich client administration tasks

Primary administration tasks to be performed by the Teamcenter administrator using the rich client:

| Task | Application |
|--|--|
| Configure persons, user, groups, and roles. |  Organization |
| Define preferences to configure the environment. |  My Teamcenter |
| Define queries for end users to find information. |  Query Builder |
| Define report definitions for end users to execute reports. |  Report Builder |
| Define permissions for data. |  Access Manager |
| Define access permissions to data for internal and external users. |  Project |
| Define process tasks to simulate company workflows. |  Workflow Designer |
| Configure end user menus. |  Command Suppression |
| Track subscriptions running in Teamcenter. |  Subscription Monitor |

Organization

An *organization* is made up of groups. Groups contain subgroups, users, and persons.



Key points

- Groups represent collective bodies of users (group members) who share data.
- Users take on a role in the group.
- The system grants data access based on group and role.
- Persons can have multiple user IDs, but typically they have only one.

Preferences

Preferences are configuration variables stored in a Teamcenter database that are read when a Teamcenter session is initiated. Preferences allow administrators and users to configure many aspects of a session, such as:

- User log on names
- Columns displayed by default in a properties table
- Business rules
- Security access
- Passwords

Query Builder

Query definitions, also called *saved queries*, identify a search criteria used to find information in Teamcenter. Administrators define query definitions for end users.

Example

Find all folders named **Home**.

Find all items that have been shipped.

Report Builder

Use *Report Builder* to create PLM XML-based report definitions that allow users to generate reports in multiple output formats using My Teamcenter or the rich or thin client. These reports can be formatted using stylesheets and can reflect data in the context of one or more selected items.

Three types of reports you can generate are:

- Summary
- Item
- Custom

Access Manager

Object protection and ownership are extremely important in a distributed computing environment. Objects represent actual product information in the database and must be protected from unauthorized or accidental access, modification, and deletion. Teamcenter implements two different tiers of data protection:

- Rules-based protection
- Object-based protection

Project

The *Project* application allows Teamcenter sites to control access to a Teamcenter database by multiple organizations, such as:

- Project teams
- Development teams
- Suppliers
- Customers

Key points

- A project is the basis for identifying a group of objects made accessible to users at a supplier's site for a particular piece of work (the project).

Workflow Designer

Workflow stems from the concept that all work goes through one or more processes to accomplish an objective. Workflow is the automation of these business processes.

Key points

- Two applications are used to accomplish workflow objectives:
 - **Workflow Designer**
A system administrator uses this application to design workflow process templates that incorporate your company's business practices and procedures into process templates.
 - **Workflow Viewer**
An end user uses this application to initiate workflow processes.
- Example tasks include **do**, **perform signoff**, **route**, and **checklist**.
- A *task* is a fundamental building block used to construct a process. Each task defines a set of actions, rules, and resources used to accomplish that task.
- Additional process requirements, such as quorums and duration times are defined in the template using workflow handlers.

Command Suppression

Command Suppression controls the display of menu and toolbar commands within Teamcenter applications. You can:

- Suppress the display of entire menus.
- Suppress the display of commands for an entire group hierarchy.
- Suppress the display of commands for an entire group.
- Suppress the display of commands for users who are assigned a specific role within a group or across groups.
- Suppress the display of specific commands on a designated menu.

For more information, see the *Command Suppression Guide*.

Key points

- Sites use Command Suppression to hide (suppress) the display of one or more commands from specific Teamcenter application windows for designated groups and/or roles.
- Suppressing the display of commands is useful if you determine that certain functions at your site should only be performed by a particular Teamcenter role or group.
- Considerations for large and small sites:
 - Large sites remove such menu commands, for example, **Tools→Import**, from application menus so that full-time system administrators can perform these tasks.
 - Small sites often allow all users to perform these actions because there is no full-time system administrator.
 - Command Suppression provides the flexibility to implement Teamcenter functions according to the needs of your organization.

Note

Command Suppression cannot be used to move options from one menu to another or to add menu options. Menu entries can only be suppressed.

Subscription Monitor

Subscriptions are requests from users who want to be notified when a specific event occurs to a specified object.

As Teamcenter administrator, use Subscription Monitor to manage and troubleshoot subscriptions. Specifically, you can:

- Generate subscription reports.
- Monitor and delete action objects in the action table.
- Monitor and delete event objects in the event table.

Activity

- Getting started with Teamcenter administration.

In this activity, you navigate the *Getting Started with Teamcenter 2007* online help.

Teamcenter architecture overview

Two architectures are available with Teamcenter:

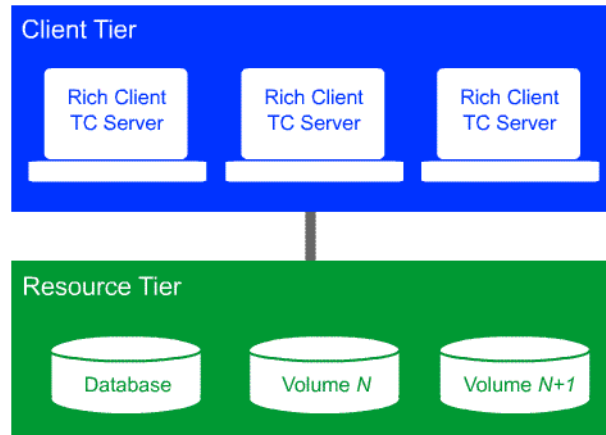
- **Two tier**
- **Four tier**

Key directories contain data you need to configure a Teamcenter site.

- **TC_ROOT** – base installation
- **TC_DATA** – database configuration settings directory

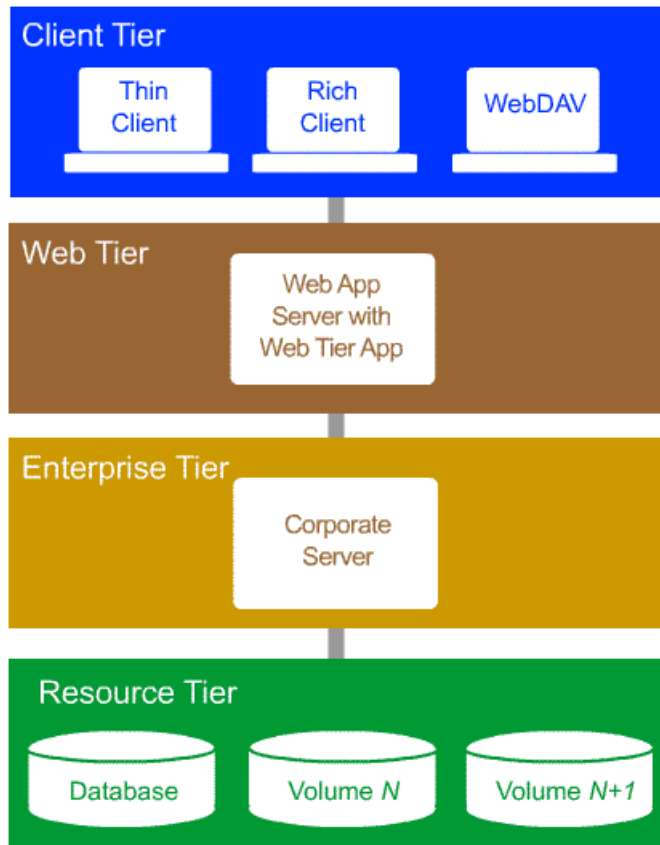
Prepare the Teamcenter system environment for command line interface tools.

Two-tier architecture logical view



- The *client tier* contains:
 - Rich client
 - Teamcenter server and executables
 - Optional applications that integrate with the rich client such as NX[®]
- The *resource tier* stores persistent metadata and files managed by the environment.
It contains:
 - Database server and database
 - Volumes
 - File servers

Four-tier architecture logical view



- The *client tier* hosts client applications, processes user interface input and output, and hosts secure file caches.

Available clients include:

- Thin client
 - Rich client
 - WebDAV client
 - Additional applications such as Teamcenter for lifecycle visualization
- The *Web tier* handles client installs, processes logon requests, routes client requests to business logic, serves static content to clients, and handles communication between the client and enterprise tiers.

The Web tier application can be:

- Java®-based and served on a J2EE Web application server such as WebLogic.
- .NET-based and served on Microsoft IIS.

- The *enterprise tier* hosts business logic, applies security rules, retrieves data from and stores data in the database, and serves dynamic content to clients.

The enterprise tier sits on the Teamcenter corporate server. It is composed of:

- Shared binary executables.
 - Shared data directories and files.
 - License server.
 - A pool of server processes managed by a server manager (four-tier environments only).
- The *resource tier* stores persistent metadata and files managed by the environment.

The resource tier contains:

- Database server and database
- Volumes
- File servers

Teamcenter environment

A working knowledge of Teamcenter environment variables is essential for administrators.

Teamcenter stores environment variable settings in a script file: **tc_profilevars** (UNIX) or **tc_profilevars.bat** (Windows).

TC_ROOT and **TC_DATA** must be set before using **tc_profilevars**.

You can automatically set these variables by your Teamcenter command prompt.

Example

start→All Programs→Siemens PLM Software Teamcenter
2007→Teamcenter 2007→prod_prod Command Prompt

Key points

The following are some of the environment variables set with **tc_profilevars**.

- **TC_BIN**
Location of the Teamcenter utilities.
- **TC_INSTALL_DIR**
Location of the Teamcenter installation files.
- **POM_SCHEMA**
Location of the database server.

Command line scripts

Run your command line scripts from your Teamcenter command prompt.

Any scripts you run during class, like **Import** and **Export**, should be run from this command prompt. This ensures the proper environment variables are set so the script runs properly.

Summary

Topics learned in this lesson:

1. The Business Modeler IDE and rich client administrative interfaces.
2. Administrative tasks for configuring the data model, defining the organization, security, reports and workflow.
3. Introduction to the two-tier and four-tier architectures.

Lesson

2 *Introduction to organization*

Purpose

The purpose of this lesson is to create users in the organization that can be used in later activities.

Objectives

After you complete this lesson, you should be able to:

- Describe the organization hierarchy.
- Create an account.

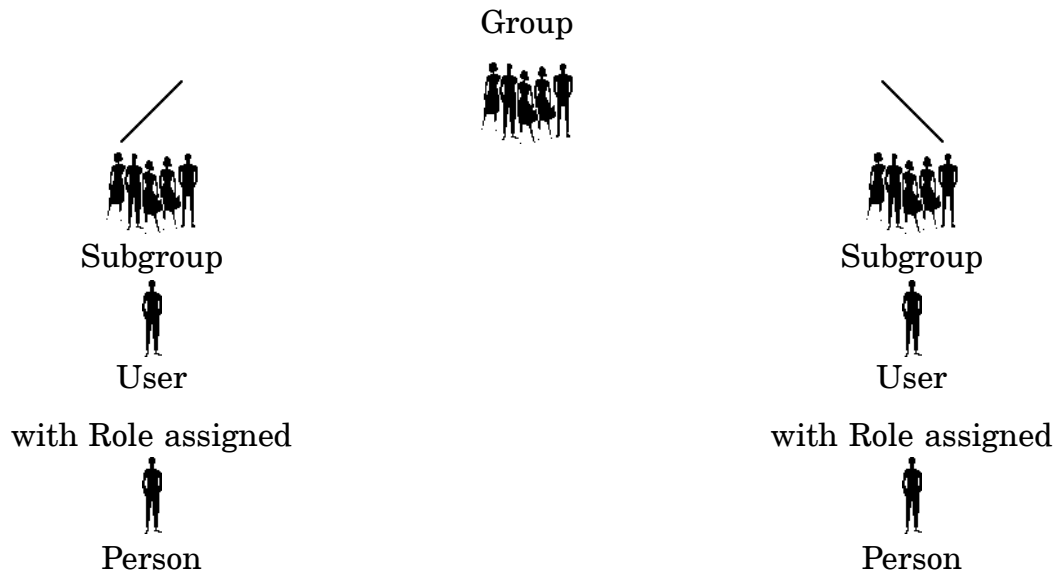
Help topics

Additional information for this lesson can be found in:

- [*Organization Guide*](#)

Defining the organization

An *organization* is made up of groups. Groups contain subgroups, users, and persons.






Key points

- Groups represent collective bodies of users (group members) who share data.
- Users take on a role in the group.
- The system grants data access based on group and role.
- Persons can have multiple user IDs, but typically they would have only one.

What is a volume

A *volume* is a location where files are stored. A volume equates to a directory on the operating system. Files stored in volumes are created by CAD applications or other third-party applications.

Example

| | |
|---|--------------------------------|
|  | <i>/disk 1/vols/user_vol</i> |
|  | <i>/disk 1/vols/group_vol</i> |
|  | <i>/disk 2/vols/group2_vol</i> |

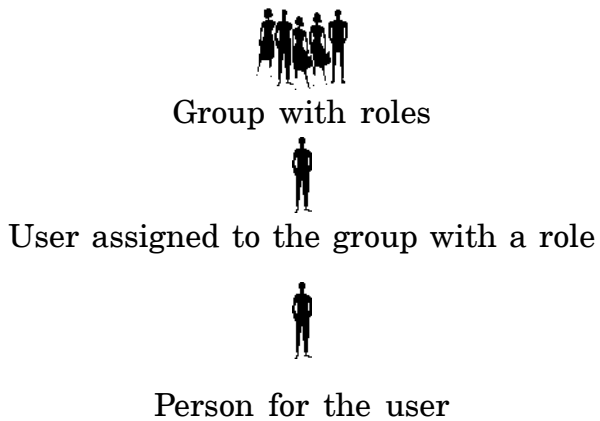
Key points

- Teamcenter retains the volume location (directory) and the file name.
- Users should always access files in volumes through Teamcenter.

Create an account

Create an account by creating a user and adding them to a group.

- Users require a person.
- Groups require a role.
- Both users and groups can optionally specify a volume.

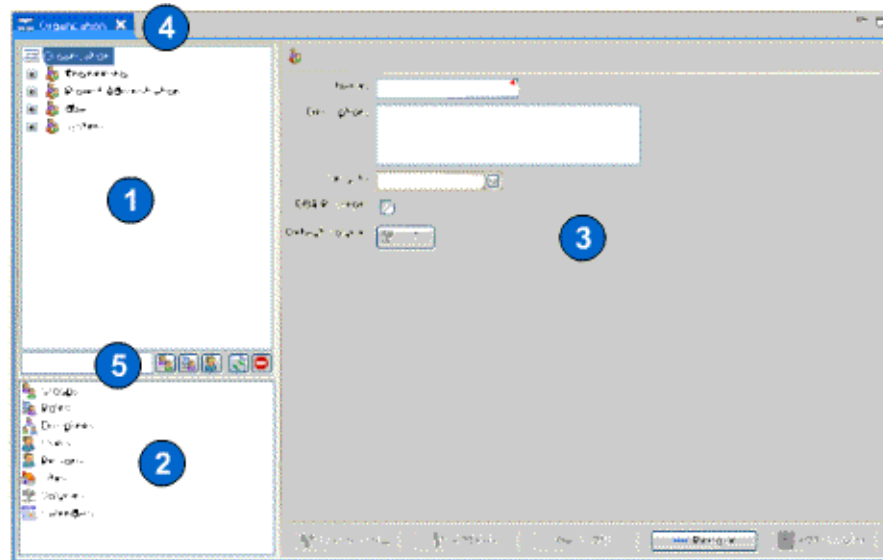


Key points

- Roles must be created before groups.
- Persons must be created before users.
- Volumes must be created before groups and users can reference a volume.
- As a best practice, volumes should be hooked to groups, not users.

Organization interface

Three distinct panes make up the **Organization** pane:



1. The **Organization** tree enables you to view the structure of your organization at a glance. By expanding and collapsing branches of the tree, you can view and manage the organizational structure.
2. The **Organization List** tree enables you to view and manage the components of your organization by listing groups, roles, disciplines, users, and persons. Sites, volumes, calendars, and ADA licenses are also maintained through the **Organization List** tree.
3. The **Organization property** pane lists the organization object's properties and values. You can create, modify, or delete items as well as clear an object's property values.
4. **Tab title** displays **Organization**.
5. **Search field** is used to filter the **Organization** tree.

To view the organizations, from **My Teamcenter** window, choose **View→Organization**. You must have the appropriate permissions to view the **Organization** application.

The organization hierarchy can be built from bottom up or using a wizard to build the hierarchy from the top down.

Create persons

The **Person** pane has two important properties:

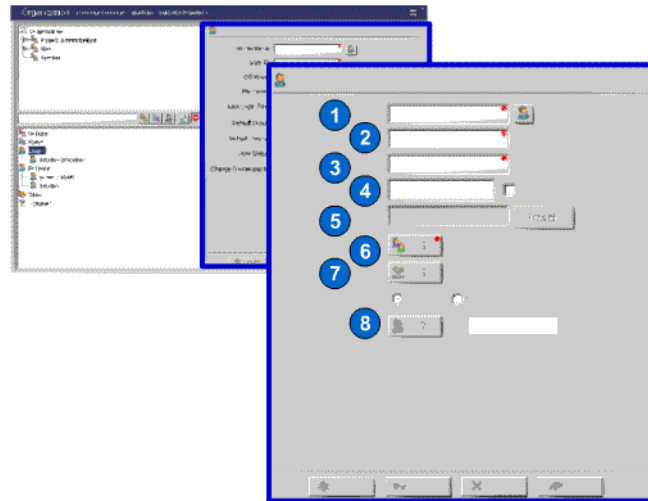
| Property | Description |
|---------------|--|
| Name | Enter the person's full name. Format the name as: <i>last name, first name, middle initial</i> Example <i>Green, Robert, M</i> |
| E_Mail | Enter the person's e-mail address. This is required for workflow notification. |

Key points

- The person name must be unique.
- The naming standard allows persons to be sorted by last name.

Create users

The **User** pane has several important properties:



| | Property | Description |
|---|----------------------------|--|
| 1 | Person Name | Specifies the person's name. Select from the list of predefined persons. |
| 2 | User ID | Specifies the unique Teamcenter user ID. |
| 3 | OS Name | Specifies the operating system user ID that is used to start Teamcenter. |
| 4 | Password | Specifies the password for the user ID. This must conform to password restrictions. |
| 5 | Last Login Time | Displays the last time this user logged onto Teamcenter. |
| 6 | Default Group | Specifies the group to which this user is assigned. |
| 7 | Default Volume | Specifies the volume to which this user is assigned. Defaults to the group's volume when left blank (recommended). |
| 8 | Change Ownership to | Changes ownership of this user's objects to another user. |

Key points

- **User ID** and **OS Name** are often the same.
- **Last Login Time** helps to determine when a user is deactivated. Reset will activate the user.
- Change ownership of the user's objects before deactivating or deleting an account. You can only change ownership if the user status is **Inactive**. A user cannot be deleted from the database if the user owns data and has approved a workflow.

Activity

- Create a new person and user.

In this activity, you create one new person and user who will be the DBA for the remainder of the course.

- Create another person and user.

In this activity, you create another person and user who will be a regular user that can be used for testing configuration.

Summary

Topics learned in this lesson:

1. The organization hierarchy breaks down into groups, roles, users, and persons.
2. Volumes store dataset files in a directory on the operating system.
3. Users require a person, and groups require a role.

Lesson

3 *Introduction to the Business Modeler IDE*

Purpose

The purpose of this lesson is to provide an introduction to the Business Modeler IDE.

Objectives

After you complete this lesson, you should be able to:

- Start the Business Modeler IDE.
- Create a project in the Business Modeler IDE.
- Describe the project files and file locations.
- Add a server profile preference to connect to the Teamcenter server.
- Define the difference between commercial off the shelf (COTS) and custom templates.
- Describe and perform the basic Business Modeler IDE process.



Help topics

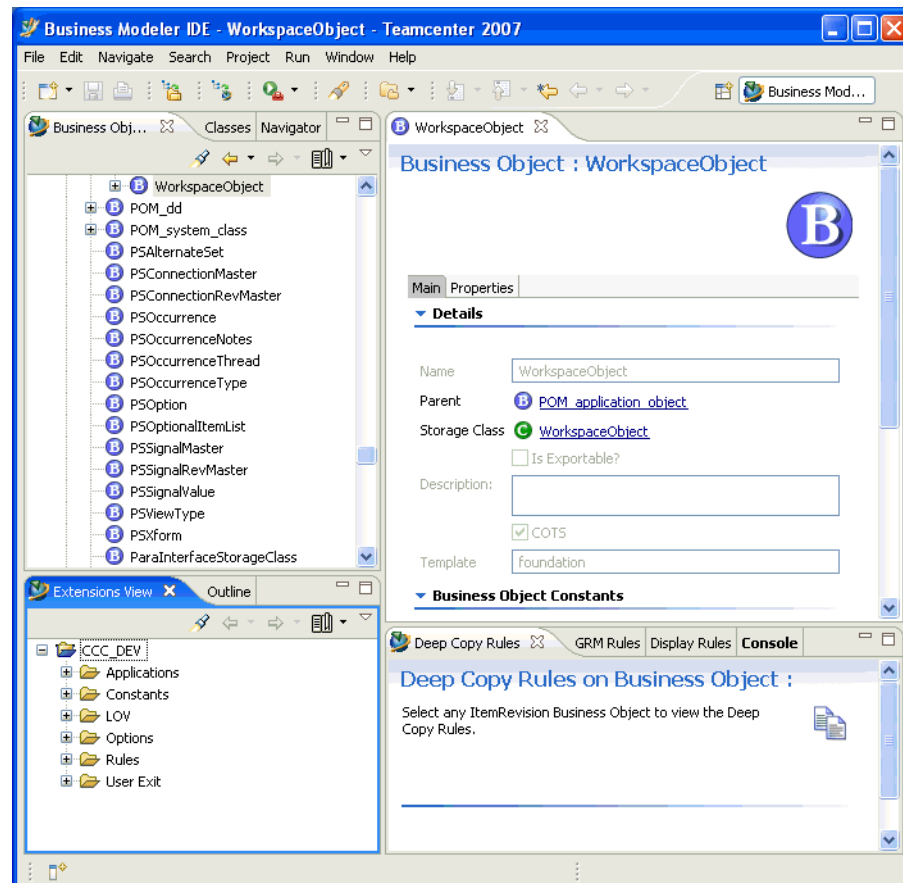
Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Getting started with the Business Modeler IDE*)

What is the Business Modeler IDE

The *Business Modeler IDE* is a tool for adding your custom data model on top of the default Teamcenter data model. The tool accomplishes this by separating the custom data model into its own set of files that are kept apart from the default data model.

-  represents business objects.
-  represents classes.



Key points

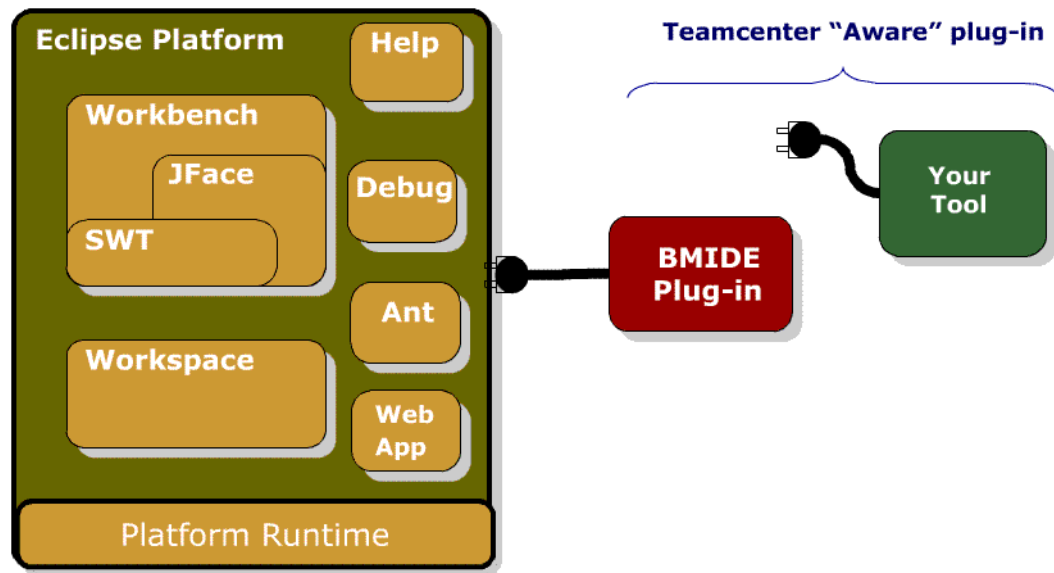
- The Business Modeler IDE (Integrated Development Environment) is a tool for customizing the data model of your Teamcenter installation. Use the IDE to create business objects, classes, attributes, lists of values (LOVs), constants, and rules.
- The Business Modeler IDE is built on top of the Eclipse platform. Eclipse is a generic platform for tool development that is extended via its plug-in and extension point technology.

What are the Business Modeler IDE and Eclipse

Whereas the *Business Modeler IDE* is the tool for adding your custom data model, *Eclipse* provides the framework for this interface.

Business Modeler IDE unification

- Best in class product tailoring
- Unification of codeless and codeful extensions
- Business analyst friendly
- Graphical programmable interface



Key points

- UML-based interface
- Focus on usability
- Emphasis on codeless extension
- Enforces best practices when programming extensions become essential

Business Modeler IDE user interface

The Business Modeler IDE utilizes the Eclipse user interface (UI), which is composed of perspectives, views, and editors. The Eclipse platform stores your projects in a directory called a workspace.

The left column below shows the logical hierarchy in the IDE. The right column shows some of the available perspectives.

| | |
|--------------------|---|
| | <i>Perspectives</i> |
| Workspace |  Business Modeler IDE (default) |
| Preferences |  CVS Repository Exploring |
| Perspective |  Debug |
| View |  Extensions |
| Editor |  Resource |
| |  Team Synchronizing |

- **View**

Tabbed window within the UI that provides a view of data.

- **Editor**

Window that allows you to edit resources.

Note

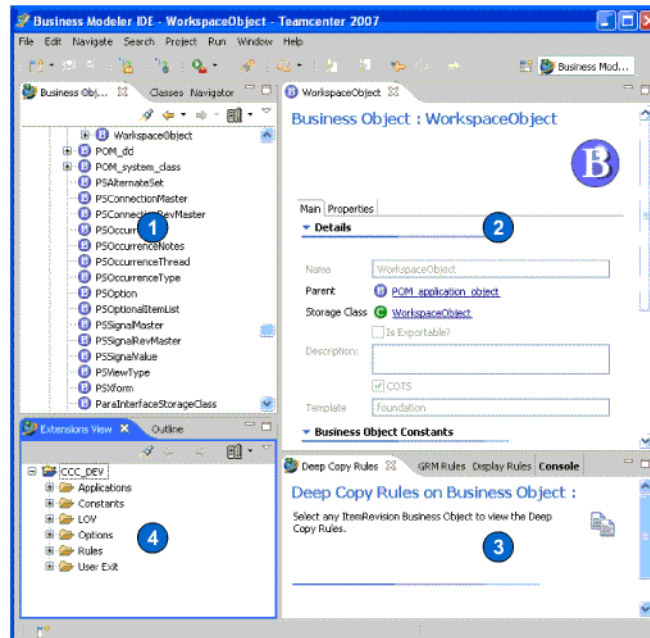
Perspectives shown in bold text are provided by Siemens PLM Software.

Key points

Each *perspective* is divided into views, and each perspective may contain multiple views and editors. If a perspective contains more than one view, the views are stacked with tabs. The **Window** menu allows you to open new perspectives and views and to move between ones already open.

The Business Modeler IDE provides its own perspectives and views, and also uses some Eclipse views. Only the perspectives and views provided by the Business Modeler IDE are documented in this course. The Eclipse user interface is fully documented in the *Workbench User Guide*, accessible by choosing **Help**→**Help Contents**.

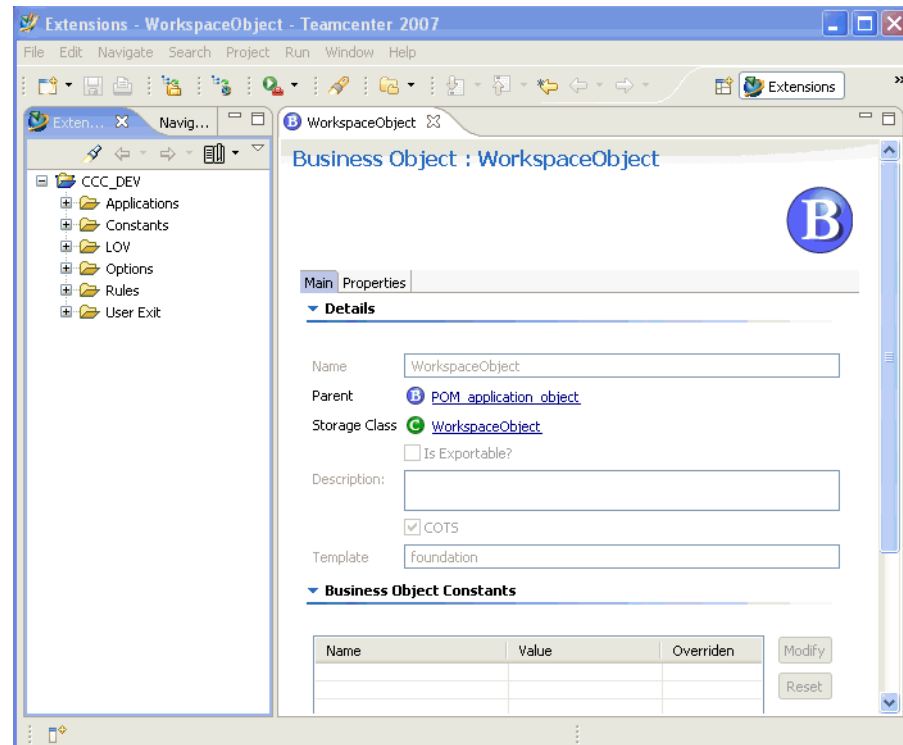
Business Modeler IDE perspective



Key points

1. Contains one view for working with business objects. Other views contain **Classes** and **Navigator**. The **Navigator** pane displays extension files, install files, output files like deployment log files, and saved UML diagrams.
2. Contains the properties for the opened item and serves as the editor for editing source files. You can have multiple items open in this pane at once. Each item is represented with a tab.
3. Contains the business rules for the selected item.
4. **Extensions view** is the same as the **Extensions** perspective that allows you to work with extensions to the data model, such as LOVs, options, and business rules. The **Outline** pane is used with the UML Editor.

Extensions perspective



Key points

1. The left side in the **Extensions** perspective contains two tabs for working with **Extensions** and **Navigator**. These views are the same as the **Extensions view** and **Navigator view** found on the **Business Modeler IDE** perspective.
2. The right side contains the properties for the selected item and serves as the editor for editing source files. You can have multiple items open in this pane at once. Each item is represented with a tab. Whatever was open in the **Business Modeler IDE** perspective will be open when you switch to the **Extensions** perspective.

Caution

Be careful not to select the **X** on the tab of the views or it will remove that view. You can use **Window**→**Reset Perspective** to restore the perspective to its original state.

Using the Eclipse framework

- Siemens PLM Software is focusing on building products on open standards.
- The Business Modeler IDE is built on the Eclipse Rich Client Platform (RCP).
- Eclipse leverages an integrated toolset for source control management.
- Eclipse has an open framework.

Key points

- Eclipse contains plug-ins that extend the framework.
- You can install your own plug-ins to extend the framework.
- Eclipse RCP provides a clear interface for building extensions.
- Plug-ins are upgradeable and interoperable.
- Source Control Management systems supported include ClearCase, Perforce, CVS, and Subversion.
- Eclipse supports C/C++ and Java editors and debuggers.
- Eclipse tools can be extended by customers to create additional tools.

Enabling the Business Modeler IDE

To enable the Business Modeler IDE, you must install it:

- As a stand-alone application.
- Or, into your Eclipse environment, if you already have Eclipse installed.

Installing the Business Modeler IDE as a stand-alone application

1. Start Teamcenter Environment Manager (TEM): **Programs→UGS Teamcenter 2007→Teamcenter 2007→Environment Manager**
2. Navigate through the **Environment Manager** until the **Select Features** window appears.

Note

The **Solutions** window does not appear.

3. Select the **Business Modeler IDE** features.

- ☒ **Client**

Installs the IDE as a client on your machine.

- ☒ **Business Modeler Templates**

Installs templates for different Teamcenter solutions. These templates are only used within the Business Modeler IDE to extend the Teamcenter data model.

Caution

Do not clear any features already selected.

4. Optionally, select a **Configure Teamcenter Servers** feature.

Select these options to automatically create a server connection profile for deployment of data model changes to a test server:

- ☒ **2 Tier Teamcenter Server Configuration**

Choose this option if you want to connect to a test server in a two-tier environment over a network using IIOP protocol.

- ☒ **4 Tier Teamcenter Server Configuration**

Choose this option if you want to connect to a test server in a four-tier environment using HTTP protocol.

5. Enter the location where you want to install the Business Modeler IDE. The Business Modeler IDE is installed to a **bmide** subdirectory.
6. After TEM finishes, you can view the installed Business Modeler IDE files in the *TC_ROOT\bmide* directory.

Start the IMR

The IMR starts the client-server communication.

By configuring the Teamcenter server from the TEM, the IMR starts automatically when starting the Business Modeler IDE.

Otherwise, before you start the Business Modeler IDE server, start the IMR using the following file:

```
TC_ROOT\portal\server_configs\start_imr.bat
```

Caution

You must do this before starting the Business Modeler IDE in the two-tier architecture in order to deploy your extensions.

Start the Business Modeler IDE

1. You can start the Business Modeler IDE in several ways, depending on how you installed it:

- Start the stand-alone application:

Windows systems:

Click the **Start** button and choose **Programs→UGS Teamcenter 2007→Teamcenter 2007→Business Modeler IDE**. This runs the **bmide.bat** file.

UNIX systems:

Run the **bmide.sh** file in the *install-location/bmide/client* directory.

- Start from an Eclipse environment

Navigate to the directory where Eclipse is installed and execute the **Eclipse** command.

Note

To ensure that you have enough memory to run Eclipse, you can run the **Eclipse.exe** command with a virtual memory argument. For example:

```
Eclipse.exe -vmargs -Xmx1024M
```

2. When you start the Business Modeler IDE for first time, the **Welcome** dialog box is displayed.

Choose one of the following links on the Welcome window to learn more about the Business Modeler IDE:

Overview

Provides links to online help topics about the Business Modeler IDE.

Tutorials

Provides links to tutorials that teach you how to use the Business Modeler IDE.

3. To work in the IDE, click the **Workbench** button on the right side of the Welcome window.

The **Workbench** is the main window in Eclipse. The **Workbench** window shows one or more perspectives. A *perspective* is an arrangement of views (such as the Navigator) and editors. At the top of the **Workbench** is a toolbar that allows you to open new perspectives and move between ones already open. The name of the active perspective is shown in the title of the window.

Note

You can access the **Welcome** window again later by choosing **Help→Welcome** from the Business Modeler IDE.

Activity

1. Start the Business Modeler IDE.

In this activity, when you start the Business Modeler IDE, the IMR is configured to automatically start. Keep the IMR window running for the remainder of the activities. You do not need to close the IMR window between activities.

2. Use the Welcome window buttons.

In this activity, you practice navigating to the **Overview**, **Tutorials**, and **Workbench** windows using the buttons on the Welcome window.

Review questions

1. The *Business Modeler IDE* is a tool for adding your custom data model on top of the default Teamcenter data model.

- True
- False

2. What is a View?

Select one answer.

- A generic platform for tool development that is extended via its plug-in.
- Tabbed window within the UI that provides a view of data.
- Unification of codeless and codeful extensions.
- Window that allows you to edit source files.

3. Why do you need the IMR?

Select all that apply.

- Client-server communication
- Deploy your extensions
- Enable the Business Modeler IDE
- Start the Business Modeler IDE

Configuring the Business Modeler IDE

You must configure the Business Modeler IDE according to the following topics before using the IDE to extend the data model.

- **Projects**

A project is a container of resources for developing, managing, packaging, and deploying a single template. It contains XML source files that hold the data model extensions. It supports concurrent development by sharing XML source files using Source Control Management (SCM) systems.

- **Business Modeler IDE preferences**

Preferences allow administrators and customizers to configure some aspects of the Business Modeler IDE.


- **Server profiles**

Server profiles define the Teamcenter servers to connect to. You must create a profile to deploy your extensions to a test server for testing or to query a server for data.

Creating a project

1. Open the perspective where the project will be created.
2. Select the **New Business Model IDE Project** as the wizard to start.
3. Follow the wizard through the creation process where you define:
 - **Project**

Add a project name. Our project example is Customer Car Company (CCC) Development.

 **CCC_DEV**

Define a location for your project files.
 - **File Repository**

Identify the location of the template files, your solution name and description, and the template dependencies.

Note

At a minimum, you will always be dependent on the **Foundation** template.

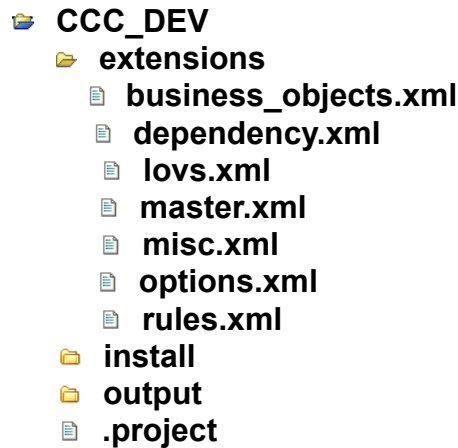
The new project now appears in your perspective.

Key points

- To create a new project, choose **File→New→Project**.
- The **Navigator** view displays your project files.
- The **Foundation** template is the base template for all customer extensions.

Project files

The **Navigator** view displays your project files.



Extensions directory

- Template XML files contain business extensions. Multiple files are available to help organize your extensions. You can create additional extension files to help organize your extensions.
- The template dependency file lists dependencies on other templates.
- The template master file contains the master include list of all XML source definition files for your template project.

Install directory

- Bundle files provide localizable strings for your TEM feature file.
- The feature file presents your template as a choice in TEM for install and upgrade.
- The install script contains a set of ordered commands which installs your template into a Teamcenter server.
- Upgrade scripts assist TEM with upgrading your template. They are only required if you have additional objects to be installed beyond what is managed inside your template.

Output directory

- The **deploy** directory is the temporary location for the template and dependency file before they are sent to the server for processing. The deploy log file is also located here.

- The **packaging** directory is the location of the template package before being used by the TEM to install or upgrade a template on a server.
- The **tcplmxml** directory is used by Global Multi-site to generate the Teamcenter PLM XML file based on the model in the Business Modeler IDE.

Note

Initially, the **output** directory is empty.

Other

- The **.project** file is used by the Business Modeler IDE and contains specific information about the type of template project the directory contains. The **.project** file is required for the Business Modeler IDE to recognize this directory as a template project directory.
- The **UML Diagrams** directory is created when the first UML diagram is saved and is the location of all subsequent UML diagram files.

Create a Business Modeler IDE template project

Before you can extend the Teamcenter data model, you must create a template project. A *project* provides an environment which manages your Teamcenter data model extensions in a custom template. The project contains folders and files that are used to organize your template XML files and to package your template for deployment.

1. Open the **Business Modeler IDE** perspective if it is not already active. Choose **Window→Open Perspective→Other→Business Modeler IDE**.
2. Choose **File→New→Project**, and in the **New Project** dialog box, choose **Business Modeler IDE→New Business Modeler IDE Template Project**.

Click **Next**.

The New Project wizard runs.

3. In the **Project** dialog box, perform the following steps:
 - a. Type a name for your project in the **Project Name** box. Do not use spaces.
 - b. Select the **Use default location** check box if you want to create the project under your default workspace.

For example, on Windows systems, the default workspace is at **C:\Documents and Settings\username\Teamcenter\BMIDE**. You can check the workspace location by choosing the **File→Switch Workspace** menu command.

If you want to create the project in another location, clear the **Use default location** check box and click **Browse** to choose another location. For example, if you are using a source control management (SCM) system to manage your XML source files, you may want to create the project in a location where the SCM can recognize it.

Caution

If you choose the location, you must follow these rules:

- The destination directory must be named the same as the project.
- Do not create the project in another project's directory.
- Do not create the project in the parent directory of your workspace.

- c. Click **Next**.

4. In the **Business Modeler IDE Template Project** dialog box, perform the following steps:

- a. Click **Browse** to the right of the **Teamcenter template folder** box to choose the directory where the data model template XML files are stored (for example, *install-location\bmide\templates*).

The template files are installed with the Business Modeler IDE.

- b. The **Solution name** defaults to your project name. This is the name of the template file created when this project's extensions are packaged for distribution.
- c. The **Solution display name** defaults to your project name. This is the solution name that appears in Teamcenter Environment Manager when this solution is installed on other servers.
- d. In the **Solution Description** box, enter a description of the project. This is the description that appears in Teamcenter Environment Manager for this solution.
- e. In the **Templates** pane, select the boxes for the templates you want to use in this project, for example, **Foundation**. (Templates contain the data model for Teamcenter solutions. The Foundation template contains the data model used for core Teamcenter functions.)
- f. Click **Finish**.

The wizard creates the project, and the IDE reads in the data model from the template XML files.

If there are any errors, they are displayed in the **Console** view.

Caution

Resolve all errors before using the Business Modeler IDE. Otherwise, you may risk extending corrupted data.

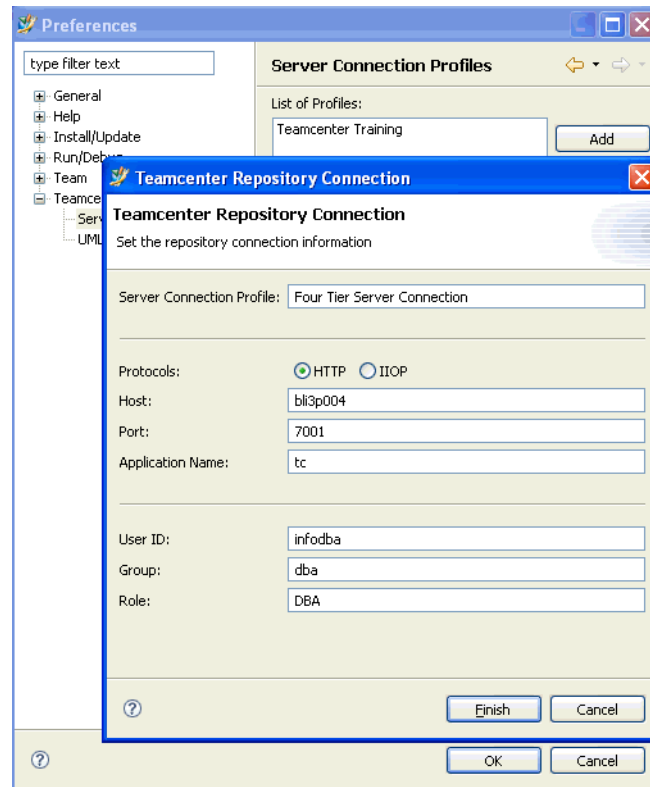
5. View the business objects in the data model.

Within the **Business Modeler IDE** perspective, click the **Business Objects** tab, the **Classes** tab, or the **Extensions View** tab. Browse through the trees in each view to see the data model.

If you want to change the project settings later, click the **Navigator** tab, right-click the new project, and choose the **Properties** menu command.

Server profile

You must deploy your extensions to a test server for verification. *Server connection profiles* define the Teamcenter test servers to connect to. A profile was created for you if the **2 Tier Teamcenter Server Configuration** or the **4 Tier Teamcenter Server Configuration** option was selected when the Business Modeler IDE was installed. If you need to connect to additional test servers, create a server connection profile for each.



Add a server profile

Server profiles define the Teamcenter servers to connect to. You must create a profile to deploy your extensions to a test server for testing or to query a server for data.

If you chose the **2 Tier Teamcenter Server Configuration** or the **4 Tier Teamcenter Server Configuration** option under **Configure Teamcenter Servers** when you installed the Business Modeler IDE, a server connection profile is already added that is named **TcData**.

Caution

You should deploy to test servers only. Do not set a server profile to deploy to a production server, because typically production servers require scheduled downtime for updating the data model. If you want to deploy your extensions to a production server, package your extensions into a solution.

1. Choose **Window→Preferences**.
2. In the **Preferences** dialog box, choose **Teamcenter→Server Connection Profiles**.
3. Click **Add**.

The Teamcenter Repository Connection wizard runs.

4. In the **Server Connection Profile** box, type the name you want to use for the profile. You can type the name of the server, for example.
5. For **Protocols**, choose either **HTTP** for a Web communication protocol (used in a four-tier Teamcenter environment) or **IIOP** for a network communication protocol (used in a two-tier Teamcenter environment).

If you chose the **2 Tier Teamcenter Server Configuration** option under **Configure Teamcenter Servers** when you installed the Business Modeler IDE, the protocol is already set up for **IIOP** as **TcData**. If you chose the **4 Tier Teamcenter Server Configuration** option, the protocol is already set up for **HTTP** as **TcData**.

6. In the **Host** box, type the host name assigned to the Teamcenter server (for the IIOP protocol) or the Web application server (for the HTTP protocol). For IIOP, this is typically set to **localhost**.
7. In the **Port** box, type the port assigned to the Teamcenter server (for the IIOP protocol) or the port of the Web application server (for the HTTP protocol).

For example, if you chose the IIOP protocol, you could enter the default port of **1572**. If you chose the HTTP protocol and you are using WebLogic as your Web application server, you could enter the default port of **7001**, or for Apache Tomcat, the default port of **8080**.

8. If you chose HTTP as the protocol, an **Application Name** box appears. Enter the name of the application to connect to.

For example, if you are using HTTP as the protocol to connect to Teamcenter, type the default application name of **tc**. This resolves the Web address to:

web_app_server:port/tc

9. If you chose IIOP as the protocol, a **Server ID** box appears. Enter the ID of the server on the network, for example, **TcServer1**.
10. In the **User ID** box, type the ID of the authorized user on the Teamcenter server (for example, type **infodba** for the default system administrator user ID).
11. In the **Group** box, type the group the user is assigned to (for example, type **dba** for the database administration group). This is optional.
12. In the **Role** box, type the role the user is assigned (for example, type **DBA** for the database administrator role). This is optional.
13. Click **Finish**.

The server profile is added to the list.

14. In the **Preferences** dialog box, click **OK** to save the preferences.
15. Perform the following steps to test the connection to the server:
 - a. Start the server by launching the Teamcenter rich client or by running the **portal/server_configs/start_imr** file.
 - b. In the **Business Objects** view, select the **Item** business object, and click the **Display Rules** tab.
 - c. Click the **Load Organization** button in the **Display Rules** view.

The Teamcenter Repository Connection wizard prompts you to log on to a server to look up the available groups and roles in the organization. (You are not asked again to log on if the Business Modeler IDE needs to query the server for data. You only have to log on once per session.)

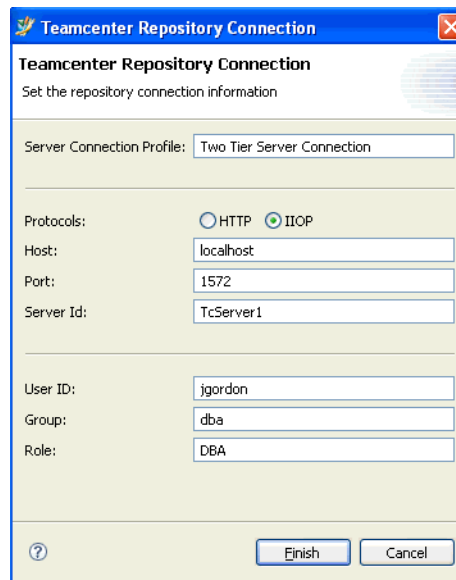
- d. If the connection is made properly, the **Display Rules** view is populated with the groups and roles from your server.

Note that the groups and roles are not stored with your template, but are temporarily cached with the Business Modeler IDE client.

Now that you have configured the Business Modeler IDE, you are ready to extend the data model.

Two-tier server connection profile

Create a two-tier server connection profile in the Business Modeler IDE and choose IIOP as the protocol.



Key points

Tip

The IIOP connection information is obtained from the following file:

`TC_ROOT\portal\plugins\configuration_RELEASE\client_specific.properties`

RELEASE represents the current product release number and equates to something like 2007.1.0.

The file contains the following definitions:

Protocol

portalCommunicationTransport=iiop

Host

IIOP_SERVER_1.HOST=localhost

Port

IIOP_SERVER_1.PORT=1572

Server

IIOP_SERVER_1.MARKER=TcServer1

Activity

1. Create a project.

In this activity, you create a project to contain the data model extensions for the Classic Car Company. The new project is named **CCC_DEV**. At this point, there are no pre-existing projects.

2. Navigate perspectives.

In this activity, you practice opening and closing perspectives. Perspectives give you a different view of your project.

3. Navigate views.

In this activity, you open, close, resize, maximize, move, and reset views so you can configure your perspectives the way you want them.

4. Create a new perspective.

In this activity, you create a new perspective with the specific configuration you need to work on a project.

Review questions

1. What is a template project?

Select all that apply.

- A container of resources for developing, managing, packaging, and deploying a single template.
 - Defines the Teamcenter servers to connect to.
 - An environment which manages your Teamcenter data model extensions in a custom template.
 - Defines access permissions to data for internal and external users.
2. Only one Server Connection Profile can be created.
 - True
 - False

Extending the data model

Once the project and server connection profile are set up, you are ready to extend the data model. The project manages your data model extensions in a custom template. Custom templates are dependent on COTS (commercial off the shelf) templates. COTS templates are protected from most changes, whereas custom templates allow updates.

| | | | |
|-------------------|--------|-------------------|--------|
| CCC_DEV | Custom | Partner | Custom |
| - | | - | |
| Foundation | COTS | CCC_DEV | COTS |
| | | - | |
| | | Foundation | COTS |

Example definition

The **Foundation** template is always a COTS template. When a customer creates the **CCC_DEV** template, it is considered Custom. If a customer provides their template to a partner to extend, the **CCC_DEV** template is a COTS template to the partner. The **Partner** template is considered custom to the partner.

COTS verses Custom templates:

- COTS templates are dependent templates that restrict most modifications and deletion of defined objects.
- Custom templates are templates that contain your extensions and allow for modification and deletion of defined objects.
- The notion of COTS or custom depends on context. Once you release your custom template to a partner, it becomes a COTS template for the partner. The extensions in your template are now COTS for the partner but still custom for you.
- The state of the COTS or custom templates is not stored in the database, but is determined at run time by the Business Modeler IDE.

Introduction to templates

A *template* is an XML file that contains the data model for an application (also known as a solution). For example, the **foundation_template.xml** file contains the data model for the Foundation solution, the base Teamcenter application. When you use the Business Modeler IDE to create custom data model, the custom data model is rolled up into a template.

You can deploy your template to a Teamcenter test server for testing purposes by right-clicking a project and choosing **Deploy Template**. This is also known as *hot deploy*.

You can also package your custom data model into a template for installation to a Teamcenter production server by choosing **File→New→Other→Business Modeler IDE→Package Template Extensions**.

Templates can exist in three locations:

- *install-location\bmide\templates*

This folder stores templates that are used for reference only within the Business Modeler IDE. The templates in this location are used when you create a project, and supply the base model for your custom data model extensions. This folder is of interest only to the Business Modeler IDE and does not affect your database status.

- *TC_DATA\model*

This folder represents the current status of your database. Templates are placed here when you use Teamcenter Environment Manager (TEM) to install Foundation or new templates or when you deploy templates from the Business Modeler IDE to a test server. All Business Modeler IDE utilities that update the database look here to find the templates to be applied to the database. This is a crucial folder for any installation or upgrade that makes database changes.

- *workspace-location\username\Teamcenter\BMIDE\project\output\packaging*

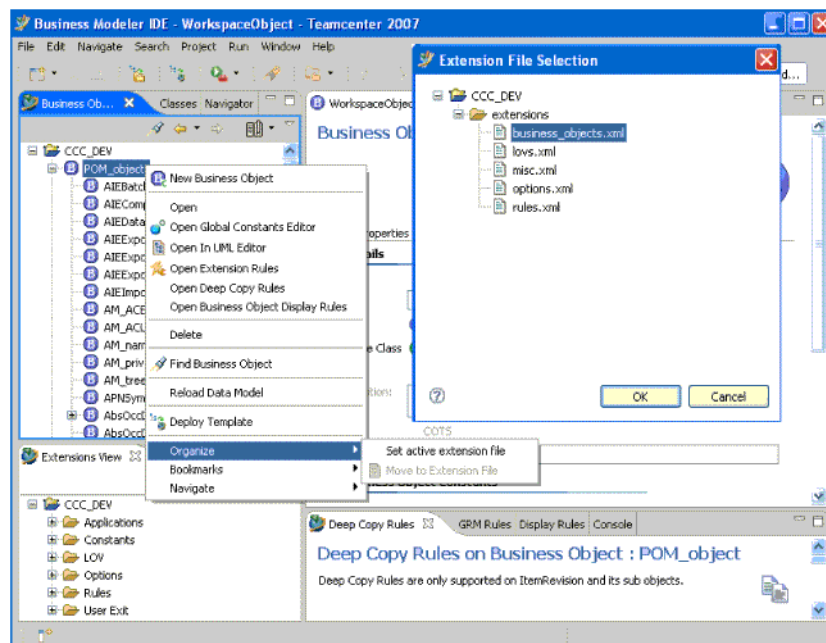
This folder is the default location where templates are packaged by the Business Modeler IDE. Templates here are a consolidation of all data model extensions you performed in your project. Once generated, you can open the template file and dependency file and verify for correctness. If you also have an installation of Teamcenter to which you want to install your template using the TEM installer, you browse to this location to obtain the template during the installation.

Basic Business Modeler IDE process

The *Business Modeler IDE* process consists of setting up extension files to store your extensions to the data model.

Additional extension files can be added, but you would typically organize your extensions in one of the following areas:

- Business objects
- LOVs
- Miscellaneous
- Options
- Rules



What is the basic Business Modeler IDE process

Whenever you extend the data model using the Business Modeler IDE, you follow this process:

1. Specify the file where you want your extensions to be saved.

Right-click an item in the **Business Objects**, **Classes**, or **Navigator** view and choose **Organize**→**Set active extension file**.

Choose an extension.

2. Perform the extension work.

3. Save your work.

Choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

4. Verify your extensions by deploying them to a test Teamcenter server.

Right-click an item in a view and choose **Deploy Template**.

Note

Before performing any of these tasks, you must have already created a project.

Set the active extension file

Before you extend the data model, choose the extension file where you want your work to be saved. By default, a new project is set up with sample XML files to store your extensions. You can choose to use these files to organize your data model extensions or replace them with your own files and folders.

When you use the Business Modeler IDE, only one XML file at a time can receive your extensions. Once you set the active extension file, all extensions are saved into this file until you set the active extension file to another file.

The extension files are provided as a convenience to organize your extension work. It really does not matter which files you place your extensions into, because all the extensions you create are rolled up into a single consolidated template file before deployment. If you forget to change the active extension file as you create your extensions, and all the changes are placed into one file such as the **business_objects.xml** file, it does not negatively affect your extension work.

1. Open one of the following views: **Business Objects**, **Classes**, or **Extensions**.
2. Right-click the project where you want to save your work and choose **Organize**→**Set active extension file**.
3. Select the file you in which you want to save your data model extensions:
 - **business_objects.xml**
Can be used to store custom business objects (types) and classes.
 - **lovs.xml**
Can be used to store custom lists of values and their attachments.
 - **misc.xml**
Can be used to store miscellaneous extensions.
 - **options.xml**
Can be used to store custom options used to support business objects, such as tools, contexts, statuses, and so on.
 - **rules.xml**
Can be used to store custom rules.

A green arrow symbol appears on the active extension file. To see which file is currently selected as the active extension file, access the **Navigator** view, open the project, and expand the **extensions** folder. This indicator symbol can be turned on or off using preferences.

4. Perform your data model extension work.
5. When you are done with the extensions, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar. This saves your work in the file you previously set.
6. To see your work in the extension file, open the project in the **Navigator** view, expand the **extensions** folder, and double-click the file. The file opens in an editor view, where you can examine the extension source code. (Do not manually edit these files.)

A **master.xml** file in the **extensions** folder points to the other files that are used to build the consolidated template. The **dependency.xml** file specifies the templates the extensions are built on (for example, the Foundation template).

Note

You can always move a custom object to another extension file (custom objects display a subscript **c** symbol indicating they are custom). Set the extension file you want as the target by right-clicking in a view and choosing **Organize→Set active extension file**, and right-click the custom object you want to move and choose **Organize→Move to extension file**. Finally, choose **File→Save Data Model**. The custom object's XML nodes are moved from their original extension file to the target extension file. To verify their move, open the original and target extension files from the **Navigator** view.

Save your changes

1. Before extending the data model, you should have chosen the file where to save your work. Right-click the project where you want to save your work and choose **Organize**→**Set active extension file**.
2. Perform your extension task. For example, create a new business object, a new LOV, a new rule, and so on.
3. After you finish performing your extension, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar. The extension work is saved in the extension file you selected earlier.

For example, if you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

4. To see your extension work in the extension file, open the project in the **Navigator** view, expand the **extensions** folder, and double-click the file. The file opens in an editor view, where you can examine the extension source code.
5. After saving changes, you can deploy the changes to the test server by running the Deploy wizard.

Note

If you are using a source control management (SCM) system with Eclipse, your project must be under source control so that the files can be saved.

If the project is under source control but is not hooked up to the SCM plug-in, when you use the **Save Data Model** option, a dialog box asks if the files are to be made writable. If you choose **Yes**, the files are made writable outside of the control of the SCM system, and if you choose **No** the save operation is aborted. The recommended option is to choose **No** and hook up the project to the SCM plug-in, or check out the files manually. If you choose **Yes**, you must synchronize the modified files with the SCM system.

For information about how to install an SCM plug-in and hook it up to a Business Modeler IDE templates project, refer to the documentation provided by the SCM plug-in.

Deploy extensions

Before deploying your extensions:

1. Close the rich client.
2. Save your project.

After deploying your extensions:

1. Check your deployment log files in one of two ways:
 - Choose the **Console** tab and select the link to open the latest log file.
 - Choose the **Navigator** pane and expand **Customer→Output→Deploy→Logs**.
2. Test the extension in the rich client.

What is deployed?

- Template definitions (**project_template.xml**)
- Dependency file (**project_dependency.xml**)

Where is it deployed?

- To the server listed in **Server Connection Profile** dialog box.

Note

The user in **Server Connection Profile** must be in the dba group and DBA role to deploy to the server.

- To the location defined for the project:
PROJECTS_DIR\CCC_DEV\output\deploy
- To the location that represents the current status of your database:
TC_ROOT\TC_DATA\model

Deploy extensions to a test server

After you make changes to the data model, you can deploy them to a test server using the Deploy wizard. This is also known as *hot deploy*. All your extensions are rolled up from your individual extension files into a single template and placed in the database. You can then run the test server to verify your extensions.

Caution

You should never use the Deploy wizard to deploy extensions to a production server. After you thoroughly test and verify your extensions on a test server, package them into a solution template using the Packaging wizard. You can then install the solution template to a production server.

A server administrator can set the **BMIDE_ALLOW_DEPLOYMENT_FROM_CLIENT** preference to **FALSE** to block a deployment from the Business Modeler IDE to a production server. By default, the preference is set to **TRUE**, which allows deployment to the server. To access preferences in the My Teamcenter application within the Teamcenter rich client, choose **Edit→Options** and click **Index** at the bottom of the **Options** dialog box.

1. To save any uncommitted changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.
2. Ensure that the Teamcenter server is running.
3. Click the **Deploy Template** button on the main toolbar, or right-click in the **Business Objects**, **Classes**, or **Extensions** view and choose **Deploy Template**.

The Deploy wizard runs.

Note

You can also run the Deploy wizard by choosing the **File→New→Other→Business Modeler IDE→Deployment** menu commands.

4. In the **Deploy** dialog box, enter the following information:
 - a. In the **Project** box, select the project whose extensions you want to deploy.
 - b. Click the arrow in the **Server Connection Profiles** box to choose the test server to deploy the extensions to.

If a server connection profile has already been created, an existing server profile is loaded. To create profiles, choose **Window→Preferences→Teamcenter→Server Connection Profiles**.

- c. For **Protocols**, choose either **HTTP** for a Web communication protocol or **IIOp** for a network communication protocol.
- d. In the **Host** box, type the host name assigned to the Teamcenter server (for the IIOp protocol), or the Web application server (for the HTTP protocol). For IIOp, this is typically set to **localhost**.
- e. In the **Port** box, type the port assigned to the Teamcenter server (for the IIOp protocol), or the port of the Web application server port (for the HTTP protocol).

For example, if you chose the IIOp protocol, type the default port of **1572**. Or if you chose the HTTP protocol and you are using WebLogic as your Web application server, type **7001**, or for Apache Tomcat, type **8080**.

- f. If you chose the HTTP protocol, an **Application Name** box appears. Enter the name of the application to connect to.

For example, if you are using HTTP as the protocol to connect to Teamcenter, type the default application name of **tc**. This resolves the Web address to:

web_app_server:port/tc

- g. If you chose the IIOp protocol, a **Server ID** box appears. Type the ID of the server on the network.
- h. In the **User ID** box, type the ID of the authorized user on the Teamcenter server (for example, type **infodba** for the default system administrator user ID).
- i. In the **Password** box, type the password for the authorized user on the Teamcenter server.
- j. In the **Group** box, type the group the user is assigned to (for example, type **dba** for the database administration group). This step is optional.
- k. In the **Role** box, type the role the user is assigned (for example, type **dba** for the database administrator role). This step is optional.
- l. Click **Finish**.

A message appears at the bottom of the dialog box stating **Deploying: Consolidating the model**.

If there are any deployment errors, take corrective action and attempt to deploy again. To see the deployment log file (**deploy.log**), click the link in the **Console** view

For every deployment, a new directory with a timestamp is generated in the **output\deploy** folder under your project. To see the files in the **deploy** folder, right-click in the **Navigator** view and choose **Refresh**. This folder contains the templates and log files generated during the deployment. View the **deploy.log** file to see the results of the deployment.

Note

You can also locate the deployment files on your system. For example, on a Windows system, they are saved by default to **C:\Documents and Settings\username\Teamcenter\BMIDE\project\output\deploy**.

5. Verify the data model changes are in the server by launching the Teamcenter rich client.

For example, if one of the changes was a new business object created as a child of the **Item** business object, in the My Teamcenter application, choose **File**→**New**→**Item**.

The new business object appears in the **New Item** dialog box.

Activity

1. Perform the basic Business Modeler IDE tasks.

In this activity, you perform the basic Business Modeler IDE tasks for creating and deploying an extension. The extension you create is a new business object.

2. Verify that **TestItem** is available for users to create.

In this activity, you log on to the rich client and create a **TestItem**.

Review questions

1. COTS templates are not protected from any changes.

- True
- False

2. The state of the COTS or custom templates is:

Select all that apply.

- Determined at run time.
- Not stored in the database.
- Predetermined.
- Stored in the database.

3. What are the four main steps in the basic Business Modeler IDE process?

Fill in the blank.

- a.
- b.
- c.
- d.

Summary

Topics learned in this lesson:

1. How to start the Business Modeler IDE.
2. Create a project and setup the server profile.
3. The difference between custom and COTS templates.
4. The basic Business Modeler IDE deployment process.

Lesson

4 *Data Model*

Purpose

The purpose of this lesson is to define the data model by creating business objects, classes and forms.

Objectives

After you complete this lesson, you should be able to:

- Define business objects and classes.
- Find and create business objects.
- Define business object properties.
- Find and create classes.
- Define class attributes.
- Create forms.
- Modify and hide the properties of a form.
- Deploy new business objects and classes.
















Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining the data model

The *data model* is the structure in Teamcenter into which items are placed. Items are the fundamental object used to model data in Teamcenter.

| Basic item structure | Business object model | Logical data model |
|---|--|--|
|  Item |  Item |  Item |
|  Item Master |  Form |  POM_object |
|  ItemRevision |  ItemRevision |  ItemRevision |
|  ItemRevision Master |  Form |  POM_object |
|  File |  Dataset |  Dataset |

- **Basic item structure**
Represents the end user view of parts and documents.
- **Business object model**
Represents the primary objects for modeling data in Teamcenter.
- **Logical data model**
Represents the persistent objects in the Teamcenter database.

Extending items

There are many reasons for extending the business objects for items:

1. Having more business objects for items may be a useful approach of categorizing data making it easier for users to find data and understand the differences between different kinds of data stored in the system.
2. Different rules for naming convention, display rules, deep copy rules, and so forth, can be configured for one business object compared to another business object.
3. Default process model association for one business object versus another is easier to implement.
4. Different designs for the **Item Master** and **ItemRevision Master** forms may be desired. Each business object can have unique and different *Master* form definitions.

New data model example

The data model can be extended to define your own basic items to create. In this example, by extending the business object model, the logical data model is automatically extended and populated.

In this example, the MyPart business object was created under Item. You can see the new MyPart forms and revision business objects and classes created as part of this process. At the end, a new dataset was also created.

A primary business object has the same name as its associated storage class. A secondary business object uses the storage class of its parent business object. Typically, most extensions to the existing business object model are secondary business objects which reuse the class of its parent.

| Business object model | Logical data model |
|---|---|
| <div><div><div><div><div></div><div>Item</div></div><div><div></div><div>MyPart</div></div></div><div><div><div></div><div>Form</div></div><div><div></div><div>MyPart Master</div></div></div></div></div> | <div><div><div><div><div></div><div>Item</div></div></div><div><div><div></div><div>POM_object</div></div><div><div></div><div>MyPartMaster</div></div></div></div></div> |
| Properties: | Attributes: |
| <div><div><div>safety_code</div></div><div><div>supplier</div></div></div> | <div><div><div>safety_code</div></div><div><div>supplier</div></div></div> |
| <div><div><div><div><div></div><div>MyPart Revision Master</div></div></div></div></div> | <div><div><div><div><div></div><div>MyPartVerMaster</div></div></div></div></div> |
| Properties: | Attributes: |
| <div><div><div>material_code</div></div><div><div>material_description</div></div><div><div>weight</div></div></div> | <div><div><div>material_code</div></div><div><div>material_description</div></div><div><div>weight</div></div></div> |
| <div><div><div><div><div></div><div>ItemRevision</div></div><div><div></div><div>MyPart Revision</div></div></div><div><div><div></div><div>POM_object</div></div><div><div></div><div>MyPartMaster</div></div><div><div></div><div>MyPartVerMaster</div></div></div><div><div><div></div><div>Dataset</div></div><div><div></div><div>CCC_MSWord</div></div></div></div></div> | <div><div><div><div><div></div><div>ItemRevision</div></div><div><div><div></div><div>POM_object</div></div><div><div></div><div>MyPartMaster</div></div><div><div></div><div>MyPartVerMaster</div></div></div><div><div><div></div><div>Dataset</div></div><div><div></div><div>CCC_MSWord</div></div></div></div></div></div> |

Defining business objects

Business objects are the fundamental objects used to model business data.

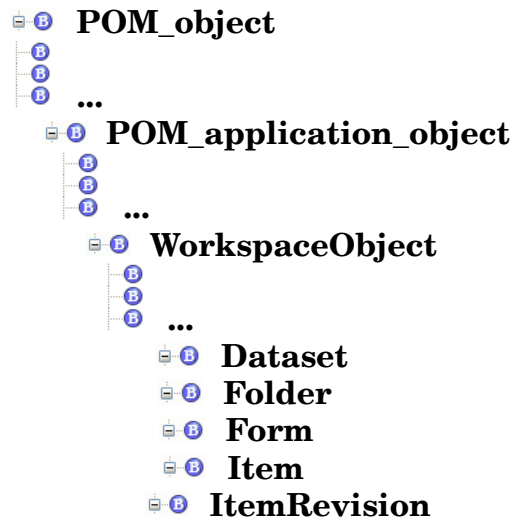
The most commonly used business objects under which you create new child business objects include:

- **Item**
Represents parts and documents.
- **ItemRevision**
Represents parts and documents revision.
- **Form**
Displays properties.
- **Dataset**
Represents file types.


What are business objects

Business objects are the abstract representations of the data model.

The following is an abbreviated view of the Teamcenter business object hierarchy.



Find business objects

1. Access the **Business Modeler IDE** perspective.
2. Select the **Business Objects** view.
3. Click **Find Business Object**  on the **Business Objects** view toolbar.
4. Type **Item** in the search box and click **OK**.

The **Item** business object is selected in the **Business Objects** view.

Note

Use bookmarks to mark commonly used business objects.

Creating item business objects

Item business objects represent product parts. You will want to extend the data model to create your own product's parts specific to your business.

New Business Object

Business Object
Create a new Business Object

Project: CCC_DEV

Name: TestItem

Parent: Item Browse...

☐ Advanced

< Back **Next >** Finish Cancel

New Business Object

Form
Create a new Form for the TestItem

Name: TestItem Master

Parent: Form Browse...

Form Storage Class

Class Name: TestItemMaster

Parent: POM_object Browse...

| Attribute Name | Type | Size | Re | |
|----------------|--------|------|----|--|
| TestPartType | String | 32 | | |

Add Modify Remove

< Back **Next >** Finish Cancel

1. After setting the **active extension file** for your project, find the **Item** business object. This is the business object you want to extend for your own item business object.
2. Right-click the **Item** business object and choose **New Business Object** to start the New Business Object wizard.
3. Give your new item a *name*.
4. Add a corresponding form with properties for the item master.
5. Create the new item revision. The revision name is automatically created based on the item name.
6. Add a corresponding form with properties for the item revision master.

7. Save you data model with the new item business object.
8. Deploy and *test* your update.

Create an item business object

Item is the most common business object under which you create a new business object. Use the **Item** business object when you want to create business objects to represent product parts. When you create a business object using **Item** or **Document** as the parent, in addition to the new business object, you also create an item master form, an item revision, and an item revision master form.

1. In the **Business Objects** view, select the project in which you want to create the new **Item** business object.
2. Right-click the project and choose **Organize**→**Set active extension file**. Select the **business_objects.xml** file as the file in which to save the data model changes.
3. Click the **Find Business Object** button on the **Business Objects** view toolbar. Type **Item** in the search box and click **OK**.

The **Item** business object is selected in the **Business Objects** view.

4. In the **Business Objects** view, right-click the **Item** business object and choose **New Business Object**.

The New Business Object wizard runs.

5. In the **Business Object** dialog box, enter the following information:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new business object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- c. In the **Parent** box, **Item** is already selected as the parent business object.
- d. Select the **Advanced** check box only if you want to choose a new class to store data for the business object.

By default, the storage class is set as the same class used by the parent business object. However, if you want to create a new class to store the data, select the **Create primary Business Object** check box. This creates a new class that has the same name as the new business object.

(A *primary* business object has the same name as its associated storage class. A *secondary* business object uses the storage class of its

parent business object. Typically, most custom business objects are secondary business objects.)

- e. Click **Next**.
6. In the **Form** dialog box, define the storage class associated with the item master form. Enter the following information in the **Form Storage Class** pane:
 - a. In the **Class Name** box, type the name you want for the new form storage class.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - b. In the **Parent** box, enter the class you want to be the parent of new form storage class.
 - c. Click the **Add** button on the right side of the dialog box to add an attribute to the form storage class.

The New Attribute wizard runs.
 - d. Click **Next**.
7. The **Business Object** dialog box displays the name of the revision to be created in the **Name** box, and displays the parent of the revision in the **Parent** box. You cannot change these values. Click **Next**.

Note

If you want to create a primary business object for the revision storage class, select **Advanced** and select **Create primary Business Object**.

8. In the **Form** dialog box, define the storage class associated with the revision master form. Enter the following information in the **Form Storage Class** pane:
 - a. In the **Class Name** box, type the name you want for the new revision master storage class.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - b. In the **Parent** box, enter the class you want to be the parent of new revision master storage class. The box is already populated if you previously chose the parent.

- c. Click the **Add** button on the right side of the dialog box if you want to add attributes to the revision master storage class.
- d. Click **Finish**.

The new business object appears in the **Business Objects** view. A **c** on the business object icon indicates that it is a custom business object.

To find the master, revision, and revision master, click the **Find Business Object** button at the top of the **Business Objects** view and search on the new business object name.

- 9. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

- 10. Deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

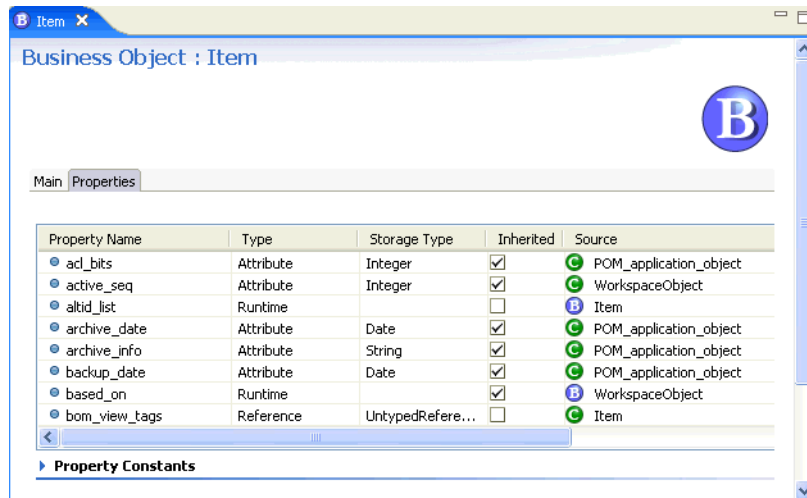
- 11. After deployment, test your new business object in the Teamcenter rich client by creating an instance of it.

For example, in the My Teamcenter application, choose **File→New→Item**.

Your new business object appears in the **New Item** dialog box. Choose your new business object and create an instance of it.

What are business object properties

A business object derives its *properties* from its persistent storage class. Properties contain information such as name, number, description, and so on. In addition to the properties that are derived from the persistent storage class, business objects can also have additional properties such as run-time properties, compound properties, and relation properties.



| Property Name | Type | Storage Type | Inherited | Source |
|---------------|-----------|------------------|-------------------------------------|------------------------|
| ad_bits | Attribute | Integer | <input checked="" type="checkbox"/> | POM_application_object |
| active_seq | Attribute | Integer | <input checked="" type="checkbox"/> | WorkspaceObject |
| altid_list | Runtime | | <input type="checkbox"/> | Item |
| archive_date | Attribute | Date | <input checked="" type="checkbox"/> | POM_application_object |
| archive_info | Attribute | String | <input checked="" type="checkbox"/> | POM_application_object |
| backup_date | Attribute | Date | <input checked="" type="checkbox"/> | POM_application_object |
| based_on | Runtime | | <input checked="" type="checkbox"/> | WorkspaceObject |
| bom_view_tags | Reference | UntypedRefere... | <input type="checkbox"/> | Item |

Property Constants

Business objects properties table

When you right-click a business object, choose **Open**, and click the **Properties** tab in the resulting view, the properties of the business object appear in a table.

Table 4-1. Columns on the business object properties table

| Column | Description |
|----------------------|---|
| Property Name | Displays the name for the property (attribute). |
| Type | <p>Indicates the type of property:</p> <ul style="list-style-type: none">• Attribute A simple value (for example, integer, string, or date). The value is stored in the database as an attribute and mapped to the property.• Reference A reference to another object. The reference is stored in the database as a reference attribute and mapped to the property.• Relation A reference to secondary objects of an ImanRelation business object. The reference is stored in the database as a relation type and is derived from that ImanRelation business object.• Runtime A property that is defined at run time and attached to types. Run-time properties do not map directly to persistent attributes, references, or relations. Instead, run-time properties derive data from one or more pieces of system information (for example, date and time) that are not stored in the Teamcenter database. Run-time properties can also be used to display a property of one type as if it were a property of another type. |

Table 4-1. Columns on the business object properties table

| Column | Description |
|---------------------|---|
| Storage Type | <ul style="list-style-type: none"> Compound A property displayed from one type as if it were the property of that type, though it actually resides on another type. Though run-time properties can be used to display such a property, they require custom coding to do so. Compound properties allow you to create such properties without custom coding. |
| | Indicates how the property is stored: |
| | <ul style="list-style-type: none"> Character A single character, such as A, B, Z. |
| | <ul style="list-style-type: none"> Date A calendar date. A form using this format displays a date selector. |
| | <ul style="list-style-type: none"> Double A double-precision floating point decimal number (an 8-byte decimal number from the range 1.7E to 308). |
| | <ul style="list-style-type: none"> ExternalReference Points to data outside of Teamcenter. |
| | <ul style="list-style-type: none"> Float A 4-byte decimal number from the range 3.4E to 38. |
| | <ul style="list-style-type: none"> Integer An integer without decimals from 1 to 999999999. |

Table 4-1. Columns on the business object properties table

| Column | Description |
|------------------|---|
| | <ul style="list-style-type: none"> • Logical A boolean value of True or False. • LongString A string of unlimited length. • Note A string of unspecified length that can be used for comments. • Short An integer number without decimals from 1 to 9999. • String A string of ASCII characters. • TypedReference Points to a Teamcenter class. • UntypedReference Points to any class of data. |
| Inherited | Indicates if the property is inherited from a parent business object. |
| Source | Lists the parent business object or class that provides the property. |
| COTS | Indicates whether the property is a standard (COTS) or custom property. COTS means <i>commercial off-the-shelf</i> . |
| Template | The template that provides the attribute. |
| LOV | Indicates whether a list of values (LOV) is attached to this property. To attach or detach an LOV to the property, right-click in the cell and choose Attach LOV or Detach LOV . |

Table 4-1. Columns on the business object properties table

| Column | Description |
|--------------------|--|
| Naming Rule | Indicates whether a naming rule is attached to this property. To attach or detach a naming rule to the property, right-click in the cell and choose Attach Naming Rule or Remove Naming Rule . |

Activity

1. Find and open a business object.

In this activity, you find and open the **Item** business object. You learn how to filter your search criteria to narrow the items to find.

2. Create an item.

In this activity, you create an item business object for the Custom Car Company. This new item is a sub-business object with **Item** as its parent.

3. Deploy the business object.

4. Verify that the new business objects are available for users to create.

In this activity, you verify that the users can create the new item, **CCC_Item**.

Review questions

1. Which of the following is the primary model for defining new objects in Teamcenter?

Select one answer.

- Basic item structure
- Business object model
- Logical data model
- Template project

2. A secondary business object

Select all that apply.

- Is the most common business objects created.
- Has the same name as its associated storage class.
- Uses the storage class of its parent business object.

3. When creating new items, how many forms are defined?

Select one answer.

- None
- One
- Two
- Four

Defining classes

Classes are the persistent objects used to store the business data in the database.

The most commonly used classes under which you create new child classes include:

- **POM_object**

Represents storage classes in the database.

- **POM_application_object**

Represents Teamcenter application classes in the database.

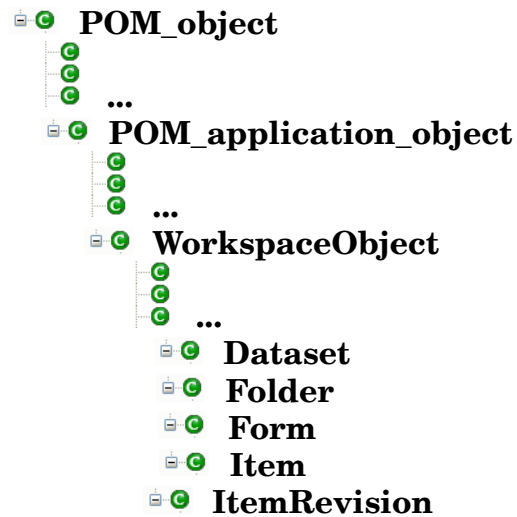
- **WorkspaceObject**

Represents commonly used classes, such as Item and Document, in the database.


What are classes

Classes are the persistent representations of the data model schema and provide attributes to business objects. The class hierarchy is known as the Persistent Object Model (POM).

The following is an abbreviated view of the Teamcenter POM schema.



Find classes

1. Access the **Business Modeler IDE** perspective.
2. Select the **Classes** view.
3. Click **Find Class**  on the **Classes** view toolbar.
4. Type **Item** in the search box and click **OK**.

The **Item** class is selected in the **Classes** view.

Note

Bookmark your favorite classes.

Adding new classes

You can add new classes to the data model or add and change class attributes. Common classes to add are new storage classes.

New Class

Class
Create a new Class

Project: CCC_DEV

Name: NewItem

Parent: POM_object Browse...

☐ Exportable ☐ Uninheritable ☐ Uninstantiable

Application Name:

| Attribute Name | Type | Size | Reference Class |
|----------------|--------|------|-----------------|
| new_attribute | String | 32 | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Add Modify Remove

? Finish Cancel

1. Right-click the class to extend and choose **New Class**.
The New Class wizard runs.
2. Fill in the class details and attributes.
3. The new class displays in the **Classes** view. A **c** on the class icon indicates that it is a custom class.
To bookmark the class, right-click the class and choose **Bookmarks**→**Add Bookmark**.
4. Save the data model and deploy the new class.

Add a new class

When you add a class, a matching primary business object is automatically generated.

A *primary* business object has the same name as its associated storage class. (A *secondary* business object uses the storage class of its parent business object. Typically, most custom business objects are secondary business objects.)

You can only create new subclasses for the following classes by creating primary business objects: **AppInterface**, **Dataset**, **Form**, **Item**, and **ItemRevision**, and **StructureContext**. If you right-click any of these classes or their children and choose **New Class**, the following message is displayed:

```
Cannot subclass directly.  
Use the Business Object Wizard to create a new Primary Business Object.
```

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Classes** tab.
3. In the **Classes** view, select the project in which you want to create a new class.
4. Right-click the project and choose **Organize→Set active extension file**. Select the **business_objects.xml** file as the file in which to save the data model changes.
5. Browse to a class under which you want to create the new class. To search for a class, you can click the **Find Class** button at the top of the view.
6. Right-click the class and choose **New Class**. The New Class wizard runs.

There are several other ways to launch the New Class wizard:

- Drag a class from the **Class** view into the UML editor.
 - Drag **Class** from the UML editor palette into the UML editor.
7. In the **Class** dialog box, enter the following information:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name of the class.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- c. The **Parent** box displays the already-selected parent class.

- d. Choose **Exportable** if the class can be exported using PLM XML.
 - e. Choose **Uninheritable** if the class cannot have children classes.
 - f. Choose **Uninstantiable** if the class cannot be instantiated.
 - g. Click the dropdown arrow in the **Application Name** box to choose the application that can be used to add or edit attributes on the class.
 Available applications appear in the **Applications** folder in the **Extensions** view. You cannot configure applications, but you may need to look at them in the **Extensions** view to understand which classes you cannot edit with ITK code.
 - h. Click the **Add** button to add an attribute to the class.
 The New Attribute wizard runs.
 - i. Click **Finish**.
 The new class is created, as well as the corresponding primary business object.
8. The new class appears in the **Classes** view. A **c** on the class icon indicates that it is a custom class. To see the corresponding new business object, open the **Business Objects** view.
 To bookmark the class, right-click the class and choose **Bookmarks→Add Bookmark**. To access the class later, click the arrow in the **Bookmarks** button at the top of the **Classes** view.
 9. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.
 If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.
 10. Deploy your changes to a test server. Right-click in the **Classes** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
 11. To verify your changes, look for the attributes on the new class that are inherited by a business object. Perform the following steps:
 - a. In the Business Modeler IDE create a new business object that uses the new class as the parent for the form storage class.
 - b. Save the changes by choosing **File→Save Data Model**.
 - c. Deploy to the test server again.

- d. In the Teamcenter rich client, create an instance of the new business object.










Open the instance and select its master or revision form. In the **Viewer** tab, you should see the new attributes you created on the new class.

What are class attributes

Attributes are class characteristics, such as name, number, description, and so on.

Attributes are attached to classes, and business objects derive their properties from the attributes on their storage class. All children of a class inherit the attributes of the parent class. All children of a business object inherit the properties of the parent business object.

▼ Attributes

| Attribute Name | Type | Size | Reference Class | Inherited | Source Class | COTS | |
|-----------------------|-----------|------|--|-------------------------------------|--|-------------------------------------|--|
| • acl_bits | Integer | | | <input checked="" type="checkbox"/> |  POM_appli... | <input checked="" type="checkbox"/> | <div>Add</div> <div>Modify</div> <div>Remove</div> |
| • archive_date | Date | | | <input checked="" type="checkbox"/> |  POM_appli... | <input checked="" type="checkbox"/> | |
| • archive_info | String | 128 | | <input checked="" type="checkbox"/> |  POM_appli... | <input checked="" type="checkbox"/> | |
| • backup_date | Date | | | <input checked="" type="checkbox"/> |  POM_appli... | <input checked="" type="checkbox"/> | |
| • bom_view_tags | Untype... | | | <input type="checkbox"/> |  Item | <input checked="" type="checkbox"/> | |
| • configuration_ob... | TypedR... | |  CFM_confi... | <input type="checkbox"/> |  Item | <input checked="" type="checkbox"/> | |
| • creation_date | Date | | | <input checked="" type="checkbox"/> |  POM_appli... | <input checked="" type="checkbox"/> | |
| • date_released | Date | | | <input checked="" type="checkbox"/> |  Workspace... | <input checked="" type="checkbox"/> | |

Class attributes table

When you right-click a class and choose **Open**, the attributes of the class appear in a table.

Table 4-2. Columns on the class properties table

| Column | Description |
|----------------|--|
| Attribute Name | Displays the property name for the attribute. |
| Type | Displays the storage type of the attribute: <ul style="list-style-type: none">• Character A single character, such as A, B, Z.• Date A calendar date. A form using this format displays a date selector.• Double An 8-byte decimal number from the range 1.7E to 308.• ExternalReference Points to data outside of Teamcenter.• Float A 4-byte decimal number from the range 3.4E to 38.• Integer An integer without decimals from 1 to 999999999. |

Table 4-2. Columns on the class properties table

| Column | Description |
|------------------------|---|
| | <ul style="list-style-type: none"> • Logical A Boolean value of True or False. • LongString A string of unlimited length. • Note A string of unspecified length that can be used for comments. • Short An integer number without decimals from 1 to 9999. • String A string of ASCII characters. • TypedReference Points to a Teamcenter class. • UntypedReference Points to any class of data. |
| Size | Displays the character length of string attributes. The maximum allowed length is 4000 characters. |
| Reference Class | Displays the reference class for the attribute (for typed reference classes). |
| Inherited | Indicates if the attribute is inherited from a parent class. |
| Source Class | Lists the parent class that provides the attribute. |
| COTS | Indicates whether the attribute is a standard (COTS) or custom attribute. COTS means <i>consumer off-the-shelf</i> , or from a dependent template. |
| Template | The template in which the attribute is defined. |
| Initial Value | Lists the initial value of the attribute on a creation window in the Teamcenter user interface. You can change this value if desired. |

Table 4-2. Columns on the class properties table

| Column | Description |
|-------------------------|--|
| Lower Bound | The lower numerical limit for a numerical or alphanumerical attribute. |
| Upper Bound | The upper numerical limit for a numerical or alphanumerical attribute. |
| Array | Specifies that the attribute is an array. |
| Array Length | Indicates the length of the array elements. |
| Public Write | Grants everyone write privileges on the attribute. |
| Public Read | Grants everyone read access to the attribute. |
| Nulls Allowed | Allows an empty value for the attribute. |
| Unique | Specifies that the attribute value cannot be duplicated. |
| Candidate Key | Specifies that when exporting an object, send this attribute and rely on the receiving site to look up this string in the local database. This is typically used for system administration defined classes such as unit of measure where a local administrator may want to control what units exist on the site. |
| Transient | Does not persist the attribute in the database. |
| Follow on Export | Exports the referenced object when the attribute is exported. |
| Export As String | Specifies that when exporting an object, export this attribute as a string. |
| No Backpointer | <p>Does not record the attribute in the POM backpointer table.</p> <p>Each time a forward pointer is created for attribute, an inverse record is stored in the POM backpointer table. Therefore, the backpointer table keeps a record of where-referenced attributes, and is used for where-referenced calls and for a check on delete that things are not referenced.</p> <p>To help system optimization, you may want to use the No Backpointer key for those attributes that are unlikely ever to be deleted (such as owning-user, owning-group, or dataset-type). Choosing the No Backpointer key saves the POM system from having to check every reference column in the table whenever a where-referenced or delete action is called.</p> |

Add or change attributes on custom classes

Attributes are item characteristics, such as name, number, description, and so on.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Classes** tab to display the **Classes** view.
3. Select the project in which you want to save the attribute extensions. Right-click the project and choose **Organize→Set active extension file**. Select the **business_objects.xml** file as the file in which to save the data model changes.
4. Browse to the custom class for which you want to manage attributes. To search for a custom class, you can click the **Find Class** button at the top of the view and select **Custom** on the **Find Class** dialog window.
5. Right-click the custom class and choose **Open**. A view displays the class details and attributes.
6. To add an attribute, click the **Add** button to the right of the **Attributes** table.

Perform the following steps in the **New Attribute** dialog box:

- a. In the **Name** box, type the attribute name.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
- b. In the **Attribute Type** box, select the storage type for the attribute.
- c. If the attribute is a string attribute, in the **String Size** box, type the character length of the attribute.
- d. If you selected **TypedReference** as the attribute type, in the **Reference Class** box select the class where the attribute is stored.
- e. In the **Initial Value** box, type the value to populate the attribute.
- f. In the **Lower Bound** box, type the lower numerical limit for a numerical or alphanumerical attribute.
- g. In the **Upper Bound** box, type the upper numerical limit for a numerical or alphanumerical attribute.
- h. In the **Array Keys** section, check the properties that apply.

- i. In the **Keys** section, check the properties that apply.
 - j. When done making changes, click **Finish**. The new attribute appears in the properties table and is marked with a **c** indicating it is a custom attribute.
7. To change values on the new attribute, click the **Modify** button. You can only change values on your custom attributes. You cannot change values on COTS (commercial off-the-shelf) attributes.

To add another attribute, click the **Add** button. To remove an attribute, click the **Remove** button.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

9. Deploy your changes to the server. Right-click in the **Classes** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
10. After deployment, test your changed attributes in the Teamcenter rich client.

For example, if you changed attributes on a child of the **Item** class, in the My Teamcenter application select a business object that inherits from that class and view its attributes in the **Viewer** tab.

Activity

1. Find and open a class.

In this activity, you search for the **CCC_ItemMaster** class that was created when you created the **CCC_Item Master** form in an earlier activity.

2. Create a class.

In this activity, you create a class. In the next activity, you make this new class the storage class for the **CCC_Item Master** form.

3. Assign a new storage class.

In this activity, you assign **CCC_ItemMasterStorage_2** as the new storage class for the **CCC_Item Master** form.

4. Deploy the new business objects.

5. Verify new class.

In this activity, you verify that the new class, **CCC_ItemMasterStorage_2**, was deployed to the server and that it is the new storage class for the **CCC_Item Master** form.

You do this by checking that the properties, **CCC_Customer**, **CCC_Shipped**, and **CCC_CarType** are in the **CCC_Item Master** form. The properties previously were **CCC_Customer** and **CCC_CarType**.

Review questions

1. Classes are the abstract objects used to store the business data in the database.

- True
- False

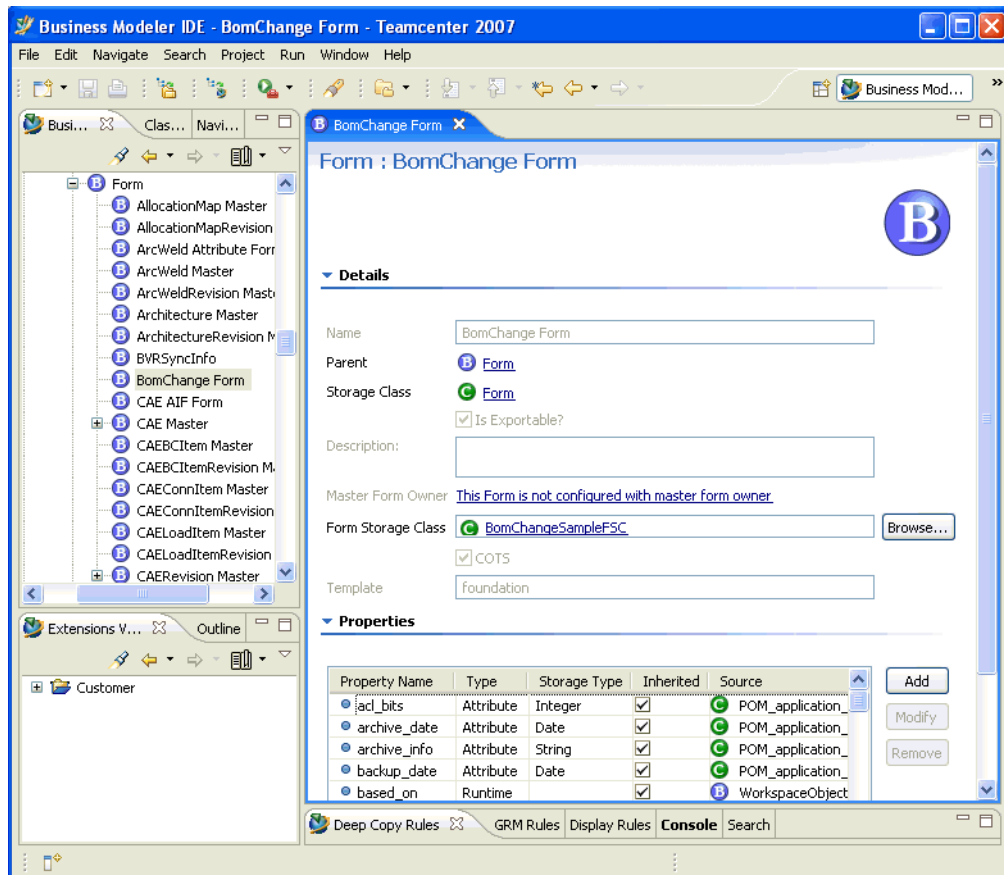
2. The most commonly used classes under which you create new child classes include:

Select all that apply.

- **Form**
- **Item**
- **POM_application_object**
- **POM_object**
- **WorkspaceObject**

Defining forms

Forms are one of the various ways of storing attribute values in Teamcenter. Forms store their data in the database as opposed to a file on the volume.



Key points

- Form properties come from two places. First, the properties of the parent business object are inherited to the child form business object. Secondly, the attributes on the form's storage class are added as run-time properties on the form business object.
- Forms can be placed on multiple items.
- An item can contain multiple forms.
- Form display can be controlled by access rules.
- Forms may apply to specific item data like CAX files.

Create a form business object

Use the **Form** business object to create a form to hold attributes. All **Item** business objects have a form associated with them, but these are typically created when you use the New Business Object wizard to create a new **Item** business object. Use the following procedure to create additional **Form** business objects to go with your **Item** and **ItemRevision** business objects.

1. In the **Business Objects** view, select the project in which you want to create the new **Form** business object.
2. Right-click the project and choose **Organize**→**Set active extension file**. Select the **business_objects.xml** file as the file in which to save the data model changes.

3. In the **Business Objects** view, right-click the **Form** business object and choose **New Business Object**.

The Create New Form Business Object wizard runs.

4. In the **Create New Form Business Object** dialog box, enter the following information:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new business object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- c. In the **Parent** box, **Form** is already selected as the parent business object.
- d. Select the **Advanced** check box only if you want to specify a different storage class from which the business object derives its properties and attributes. Choose one of the following in the **Form Storage Class** pane:
 - Click **Use new class** to create a new storage class to store the attributes.

In the **Name** box, type the name for the new form storage class. Click the **Browse** button to the right of the **Parent** box to select the class you want to be the parent of the new form storage class.

Note

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- Click **Use existing class** to use an existing class to store the attributes.

Click the **Browse** button to the right of the **Name** box to select the class you want to be the form storage class.

- e. Click the **Add** button on the right side of the dialog box to add an attribute to the form storage class.

The New Attribute wizard runs.

After the attribute is added, you can change values as desired by clicking cells in the attribute table.

- f. Click **Finish**.

The new business object appears in the **Business Objects** view. A **c** on the business object icon indicates that it is a custom business object.

5. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

6. Deploy your changes to a test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

7. After deployment, test your new business object in the Teamcenter rich client by creating an instance of it.

For example, in the My Teamcenter application, choose **File→New→Form**. In the **New Form** dialog box, click the **More...** button and choose the new form business object from the list of available forms. Create the instance of the form and click **OK**.

Storage class form

When a business object form is created:

1. A form storage class is created under the POM_object.
2. A business item is created under the POM_object with the same name as the form storage class.

Business object model

 **Form**
 **MyPart Master**

Properties:

safety_code

supplier

 **POM_object**
 **MyPartMaster**

Properties:

safety_code

supplier

Logical data model

 **Form**

 **POM_object**
 **MyPartMaster**

Attributes:

safety_code

supplier

Add, change, or remove properties on a form business objects

Properties are essentially attributes that are inherited by business objects. Children of the **Form** business object are the only business objects to which you can directly add properties. This is the same process as adding attributes to classes.

| Property Name | Type | Storage Class |
|-----------------------|-----------|---------------|
| ad_bits | Attribute | Integer |
| archive_date | Attribute | Date |
| archive_info | Attribute | String |
| backup_date | Attribute | Date |
| based_on | Runtime | |
| change | Runtime | |
| checked_out | Runtime | |
| checked_out_change_id | Runtime | |

1. Access the **Business Modeler IDE** perspective.
2. Select the **Business Objects** view.
3. Select the project in which you want to save the form properties extensions.
4. Browse to the child of the **Form** business object for which you want to add properties.
5. Open the form business object.
6. Use **Add**, **Modify**, or **Remove** to change the properties in the form.

Hide properties on a form business object

Once forms are deployed and populated in Teamcenter, you may need to hide a property you no longer use.

To hide a property on a form, override the **Visible** property constant for the attribute on the form business object.

The screenshot shows the 'MyPart Master' form business object configuration window. The 'Properties' tab is selected, displaying a table of properties. Below this, the 'Property Constants' section is expanded, showing a table of constants. The 'Visible' constant is highlighted, and its 'Value' is set to 'false'.

| Property Name | Type | Storage Type | Inherited | Source |
|-------------------|-----------|----------------|-------------------------------------|-----------------|
| reservation | Runtime | | <input checked="" type="checkbox"/> | WorkspaceObject |
| revision_limit | Attribute | Integer | <input checked="" type="checkbox"/> | WorkspaceObject |
| revision_number | Attribute | Integer | <input checked="" type="checkbox"/> | WorkspaceObject |
| safety_code | Runtime | | <input type="checkbox"/> | MyPart Master |
| supplier | Runtime | | <input type="checkbox"/> | MyPart Master |
| timestamp | Attribute | String | <input checked="" type="checkbox"/> | POM_object |
| user_can_unmanage | Runtime | | <input checked="" type="checkbox"/> | WorkspaceObject |
| wso_thread | Reference | TypedReference | <input checked="" type="checkbox"/> | WorkspaceObject |

| Name | Value | Overriden | COTS | Template |
|--------------|----------|-------------------------------------|-------------------------------------|------------|
| Enabled | false | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Exportable | Optional | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| InitialValue | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Modifiable | Read | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Required | false | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Visible | false | <input checked="" type="checkbox"/> | <input type="checkbox"/> | customer |

1. Open the form business item.
2. On the **Properties** pane, select the attribute to hide.
3. Select the **Visible** property constant.
4. Click **Modify**.
5. Click **Value** to clear the box.
6. Click **Finish**.

Clear **Public Read** for the attribute on the storage class.

Activity

1. Create a form.

In this activity, you create a custom change form called **CCC_Change**.

2. Add a property to a form.

In this activity, you add a property called **CCC_Custom** to the **CCC_Item Master** form.

3. Deploy the new form.

4. Verify that the new form is available for users to create.

In this activity, you verify that the users can create the new form, **CCC_Change**. You also verify that the new property, **CCC_Custom** was added to the **CCC_Item Master** form.

Review questions

1. What items best describe forms?

Select all that apply.

- An item can contain multiple forms.
- Forms can be placed on one item.
- Forms store their data in a file on the volume.
- Form properties are inherited from the form's parent.

2. Where is the form storage class created?

Select one answer.

- **Form**
- **Item**
- **POM_object**
- **WorkspaceObject**

Summary

Topics learned in this lesson:

1. Extended the **Item** business object with a custom item.
2. Locate and navigate business objects.
3. Create new storage classes.
4. Create forms and modify form properties.
5. Hide form properties.

Lesson

5 *UML editor*

Purpose

The purpose of this lesson is to use the Unified Modeling Language (UML) editor to manage the data model.

Objectives

After you complete this lesson, you should be able to:

- Define a UML diagram.
- Set preferences for the UML editor.
- Open a class or business object in the UML editor.
- Create new classes and business objects in the UML editor.

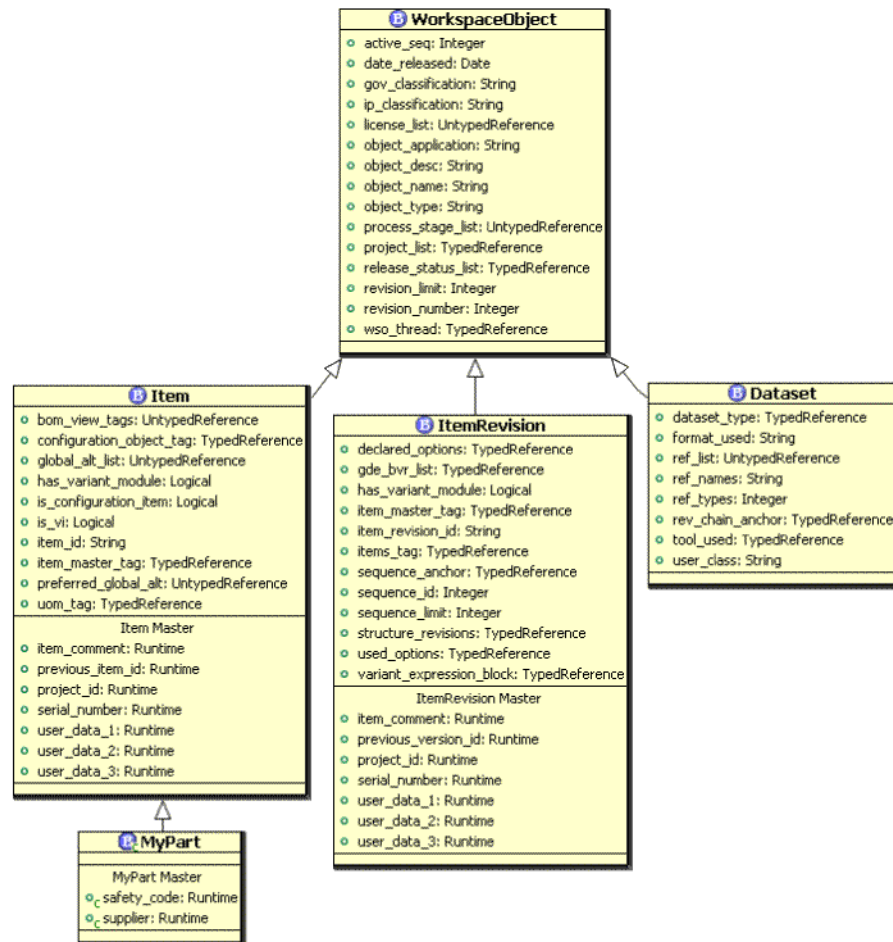
Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining the UML editor

UML (Unified Modeling Language) is a commonly used method to graphically represent data models. The UML editor allows you to graphically view and change the data model for business objects and classes.



Key points

- UML is an industry standard language.
- You can do much of your data model extension work directly from the UML editor.
- For more information about UML, see the following URL:

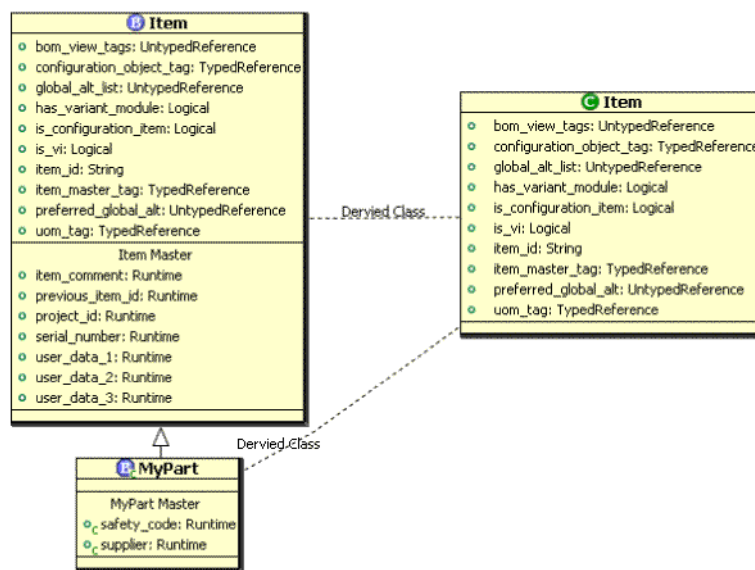
<http://www.uml.org/>

What is a UML diagram



The Business Modeler IDE UML diagram can be used to quickly understand the *business object model* and *logical data model*.

Business object model

Logical data model

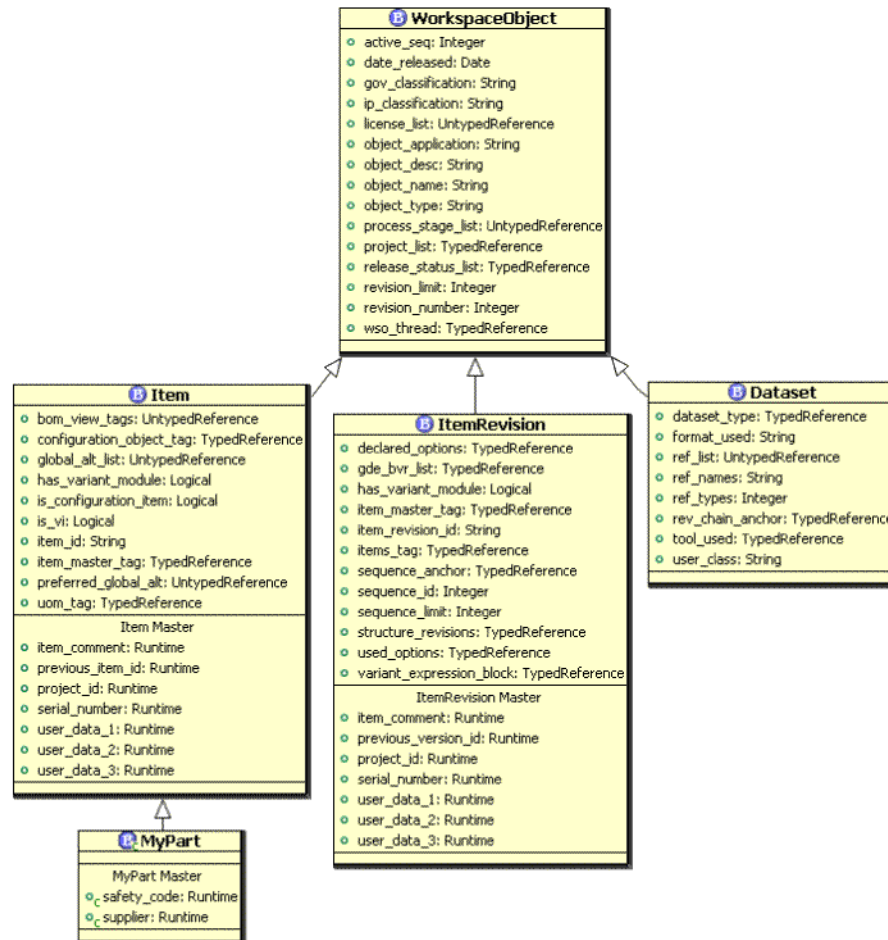


Key points

- Business objects  display their properties and optionally, their associated form's properties. The Primary Form Group filter is available to hide the associated form properties from displaying.
- The small green  is a symbol that indicates that your object is a *custom-defined* object. *Custom* indicates that you defined it; you own it.
- Each business object has a storage class in the logical model where the attribute data is persisted.
- The attributes on the storage class are displayed as properties on the business object.
- When you extend to create a new business object, your storage class is the same as your parent business object.
- UML files store only the names and relative positions of the data model. The definitions of the data model created with the UML editor are not stored in the UML files. They are stored in the project template XML files.

Data model inheritance

The **MyPart** business object inherits properties and behavior from its parents, **Item** and **WorkspaceObject**. **ItemRevision** and **Dataset** inherit from **WorkspaceObject**.



Inheritance is displayed in the UML editor by right-clicking on the title of the business object or class and selecting **Show→Parent**, **Show→Children**, or **Show→Inheritance to Root**.

Set preferences for the UML editor

You can set Business Modeler IDE UML editor preferences from the **Window→Preferences** menu.

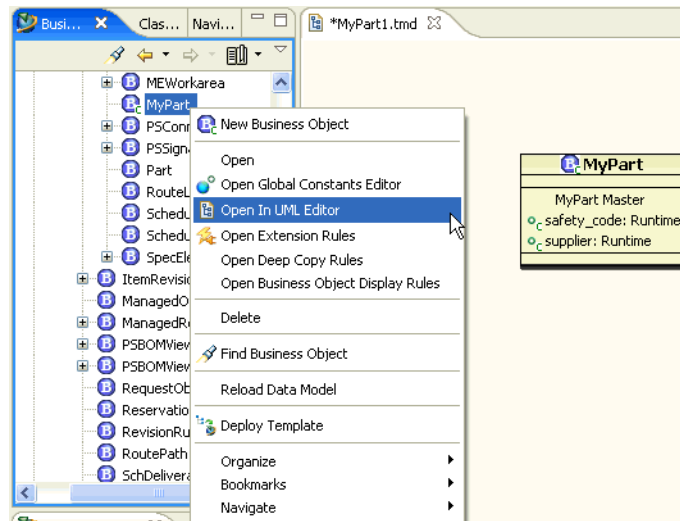
1. Choose **Window→Preferences**.
2. In the **Preferences** dialog box, select **Teamcenter**.
3. In the **UML Editor Preference** dialog box, click the **UML Editor Background Color** box to select a new color to display behind the items displayed in the UML Editor.
4. When done making changes, click **Apply** to commit the changes for that preference.
5. Click **OK** to commit the changes you have made to all preferences.

Managing the data model using the UML editor

You manage the data model in the UML editor by:

- Opening a business object in the UML editor.
- Expanding parents and children of the business object.
- Expanding the **Palette** to control selection of business objects and classes.
- Working with the **Outline** view to navigate the UML structure.
- Creating a new UML diagram for your customizations.
- Creating a new business object in the UML editor.

Opening a business object in the UML editor



To open the **MyPart** business object in the UML editor:

1. Find **MyPart** in the **Business Objects** view.
2. Right-click **MyPart** and choose **Open In UML Editor**.

Open a class or business object in the UML editor

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. If you want to work with business objects, click the **Business Objects** tab to display the **Business Objects** view. If you want to work with classes, click the **Classes** tab to display the **Classes** view.
3. Select the project in which you want to work. Right-click the project and choose **Organize→Set active extension file**. Select the file where you want to save the data model changes. For example, if you are creating new business objects or classes, choose the **business_objects.xml** file.
4. Browse to the business object or class you want to work with. To search for a business object or class, you can click the **Find** button at the top of the view.

Item is the most common business object or class with which you work.

5. Right-click the business object or class and choose **Open In UML Editor**.
The business object or class appears in the UML editor as a box.
6. To learn the basics of the UML editor, perform the following steps:
 - a. Work with the shortcut menu.

Right-click the name of the business object or class that appears at the top of the box.

A shortcut menu appears. Try the following tasks:

- To learn how to display hierarchy, choose **Show** and choose **Children**, **Parent**, or **Inheritance to Root**. If you are viewing a business object, also choose **Show→Storage Class** to see the class where the business object data is stored. To filter out what is displayed, choose **Select Filters**.
- To create a new business object or class, choose **Add Business Object** or **Add Class**.
- To undo actions, choose **Undo**.
- To access other views for a business object, choose **Open Extension Rules**, **Open Deep Copy Rules**, or **Open Display Rules**.

- b. Work with the palette.

Click the arrow at the top of the **Palette** bar docked on the right of the UML editor. The palette expands to display a series of buttons. Try the following tasks:

- To select individual business objects or classes, click the **Select** button.
 - To select a group of business objects or classes, click the **Marquee** button and drag a square around a grouping you want to select.
 - To create a new business object or class, drag **Class** or **Business Object** into the UML editor.
- c. Work with the **Outline** view.

The **Outline** view displays a thumbnail of the UML editor with a gray box indicating what is currently displayed.

- Drag the gray box across the **Outline** view to change what is displayed in the UML editor.
 - Click in the UML editor to activate the thumbnail in the **Outline** view, then click the zoom control in the toolbar at the top of the window to change the view from 100% to a different size.
- d. View the UML file.
- The data is saved in a file with a **.tmd** suffix as displayed on the tab at the top of the UML editor. To open this **.tmd** file later and view it in the UML editor, access the **Navigator** view and double-click the **.tmd** file in the **UML Diagram** folder. Saving the diagram does not save the data model; you must choose **File→Save Data Model**.

7. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

You can see the data model changes in an XML file. For example, if you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

8. After you make changes to the data model using the UML editor, you can deploy your changes to the test server. Right-click in the **Business Objects** or **Classes** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

Create a new UML diagram

1. Open the **Business Modeler IDE** perspective if it is not already active. Choose **Window→Open Perspective→Other→Business Modeler IDE**.
2. Select the project in which you want to work. Right-click the project and choose **Organize→Set active extension file**. Select the file where you want to save the data model changes. For example, if you are creating new business objects or classes, choose the **business_objects.xml** file.
3. To create a new folder to hold the new UML diagram, click the **Navigator** tab and then choose **File→New→Other→General→Folder**.
4. Click the **New** button on the toolbar, or choose **File→New→Other** menu, and in the **New** dialog box, choose **Business Modeler IDE→Create a new Teamcenter UML diagram**.

Click **Next**.

The New UML Diagram wizard runs.

5. Perform the following steps in the **New UML Diagram** dialog box:
 - a. Select the folder where you want to save the diagram, for example, **UML Diagrams**.
 - b. In the **File Name** box, type the name you want to give to the diagram. The file has a **.tmd** file suffix.
 - c. Click **Finish**.

The UML editor appears.

6. To work in the UML editor, right-click in the view and choose menu commands.

You can also create business objects or classes by dragging the **Class** or **Business Object** icons from the UML editor palette into the editor.

7. To save the UML diagram, click the **Save** button on the main toolbar. The diagram is saved in a file with a **.tmd** suffix as displayed on the tab at the top of the UML editor.

To open this **.tmd** file later and view it in the UML editor, access the **Navigator** view and double-click the **.tmd** file.

Note

Saving the diagram does not save the data model. To save the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

Create a new class or business object

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. If you want to work with business objects, click the **Business Objects** tab to display the **Business Objects** view. If you want to work with classes, click the **Classes** tab to display the **Classes** view.
3. Select the project in which you want to work. Right-click the project and choose **Organize→Set active extension file**. Select the file where you want to save the data model changes. For example, if you are creating new classes choose the **business_objects.xml** file, or if you are creating new options, choose the **options.xml** file.
4. Right-click the name of the business object or class that appears at the top of the box. A shortcut menu appears. Try the following task:
 - To create a new business object or class, choose **Add Business Object** or **Add Class**.
5. Click the arrow at the top of the **Palette** bar docked on the right of the UML editor. The palette expands to display a series of buttons. Try the following task:
 - To create a new business object or class, drag **Class** or **Business Object** into the UML editor.
6. The data is saved in a file with a **.tmd** suffix as displayed on the tab at the top of the UML editor. To open this **.tmd** file later and view it in the UML editor, access the **Navigator** view and double-click the **.tmd** file. Saving the diagram does not save the data model; you must choose **File→Save Data Model**.
7. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

You can see the data model changes in an XML file. For example, if you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

Activity

1. Open a business object in the UML editor.

2. Manage how objects are displayed in the UML diagram.

In this activity, you expand the data model hierarchy and learn to use the palette toolbar.

3. Create a business object in the UML editor.

In this activity, you create the **CCC_Car** business object in the UML editor.

4. Save a UML diagram.

In this activity, you save a UML diagram and deploy data model changes to the server.

Saving the UML diagram does not save any changes that you made to the data model.

5. Open a saved UML diagram.

In this activity, you open a saved UML diagram.

6. Verify the new business object.

In this activity, you verify that the **CCC_Car** business object you created in the UML editor is available for users to create.

Review questions

1. Which are true for the UML editor?

Select all that apply.

- A commonly used method to graphically represent data models.
 - Allows you to graphically view and change the data model for business objects and classes.
 - An industry standard language.
 - You can do much of your data model extension work using the UML editor.
2. The UML editor displays only the business object model.
 - True
 - False
 3. How do you display inheritance in the UML editor?

Select all that apply.

- **Show→Children**
- **Show→Inheritance to Root**
- **Show→Parent**
- **Show→Storage Class**

Summary

Topics learned in this lesson:

1. Define the UML editor.
2. Set preferences for the UML editor.
3. Open a class or business object in the UML editor.
4. Save a UML diagram.
5. Create new classes and business objects in the UML editor.

Lesson

6 *Lists of values*

Purpose

The purpose of this lesson is to define and manage list of values (LOV).

Objectives

After you complete this lesson, you should be able to:

- Describe list of values (LOV).
- Create, modify and delete list of values.
- Add, remove and clear values from LOV.
- Attach LOV to a property.
- Describe LOV variants.
- Create a cascading LOV.


Help topics

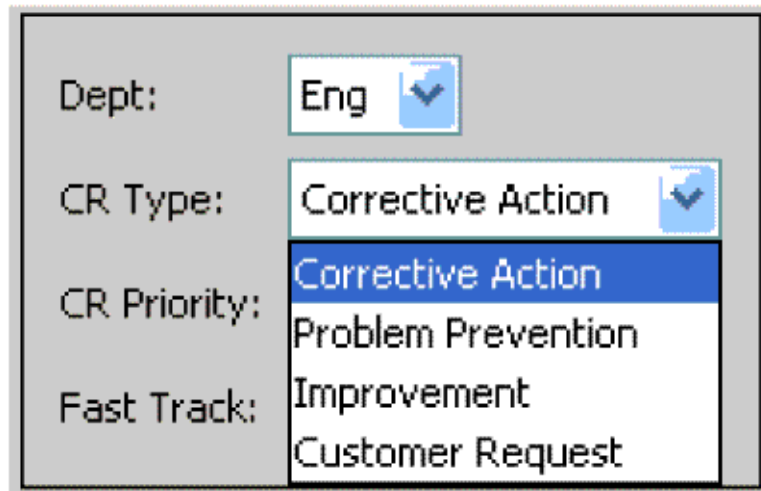
Additional information for this lesson can be found in:



- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining list of values (LOV)

List of values (LOV) are pick lists of data entry items. They are commonly accessed by Teamcenter users from a menu at the end of a data entry box. After you create a list of values, you must attach it to a property on a business object.

When you implement an LOV for a field, an LOV button  is displayed next to that property.



| | |
|--------------|---|
| Dept: | Eng  |
| CR Type: | Corrective Action  |
| CR Priority: | <div>Corrective Action</div> <div>Problem Prevention</div> <div>Improvement</div> <div>Customer Request</div> |
| Fast Track: | |

Key points

- The user simply clicks the LOV button and an object selection list displays allowable entries for that property.
- When you associate an LOV, you link this LOV to any property where that data can be entered and display an LOV to the user.

List of values (LOV) interface

The screenshot shows the 'LOV : Make' interface. It has a 'Details' section with fields for Project (Customer), Name (Make), Description, Type (String), Usage (Exhaustive, Suggestive, Range), Reference, and a 'Show Cascading View' checkbox. Below these is a table with 'Value' and 'Description' columns, containing 'Racing' and 'Show'. To the right of the table are buttons: Add, Remove, Modify, Move Up, Move Down, Clear, and Load. Below the table is a 'Template' field with 'customer' and a 'COTS' checkbox. The 'LOV Attachments' section at the bottom has a table with columns: Business Object, Property, COTS, and Template. It lists 'CCC_Car Master.CCC_Make' with COTS checked and Template 'customer'. To the right are buttons: Attach, Detach, and Edit.

LOV Type:

- You must pick the appropriate type and it must match the class attribute type.

LOV Usage:

- Exhaustive**

Indicates that the list contains all possible choices.

- Suggestive**

Specifies that the list contains suggested choices. The user can enter their own value if they want.

- Range**

Indicates that the list falls within a range of numeric values.

LOV Reference:

- If you choose string, tag, or tag extent as the type, a **Reference** box appears. You can use this to obtain values for the LOV from a class attribute.
- If a class attribute is specified for a string type, click the **Load** button to obtain values from the database and populate the table with items from the attribute.
- If a class attribute is specified for a tag extent, the actual values of the LOV are computed at run-time when the user attaches this LOV to a property.

Add a list of values (LOV)

Perform the following steps to create a pick list that end users can access from a menu on a data entry box:

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create an LOV.

Right-click the project and choose **Organize→Set active extension file**. Select the **lovs.xml** file as the file in which to save the data model changes.

3. Expand the project folder until you see the **LOV** folder displayed.
4. Right-click the **LOV** folder and choose **New LOV**.

The New LOV wizard runs.

5. Enter the following information in the **LOV** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new LOV.
When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - b. In the **Description** box, state the purpose of the LOV.
 - c. In the **Type** box, select the type of LOV you are creating. Entries containing **Extent** must match an existing value in the database.

- **Character**

Single ASCII character.

- **Date**

Date and time in the format used at your site.

- **Double**

Double-precision floating point decimal number (sometimes called a real).

- **External Character Extent**

Single ASCII character in the database.

- **External Date Extent**

Date and time in the database.

- **External Double Extent**
Double-precision floating point decimal number in the database.
- **External Integer Extent**
Whole number in the database.
- **External String Extent**
A string in the database.
- **Filter**
Reference to another nonfiltered LOV with the capability to filter the referenced LOV's values.
- **Integer**
Whole number.
- **Integer Site Extent**
A site name in the database. Valid values are all existing instances of the selected site.
- **String**
String of ASCII characters.
- **String Extent**
A string of ASCII characters in the database. Valid values are all existing instances of strings.
- **String GR Name Extent**
A group name in the database. Valid values are all existing instances of group names.
- **String PUBR Type Extent**
Workspace object in the database. Valid values are all existing workspace object types.
- **String Status Extent**
Status in the database. Valid values are all existing release status names.
- **String User Name Extent**
A user name in the database. Valid values are all existing instances of the active **user_name** attribute in the **POM_user** class.

- **String User ID Extent**
A user ID in the database. Valid values are all existing instances of the active **user_id** attribute in the **POM_user** class.
 - **String WSO Class Extent**
Workspace object class in the database. Valid values are all existing instances of the selected class.
 - **Tag**
A unique tag (for example, an item ID).
 - **Tag Extent**
Reference to a unique tag in the database.
 - **Tag RDV Search Revision Rule**
Repeatable Digital Validation (RDV) tag.
- d. In the **Usage** section, click one of the following buttons:
- **Exhaustive**
Indicates that the list contains all possible choices.
 - **Suggestive**
Specifies that the list contains suggested choices. The user can enter their own value if they want.
 - **Range**
Indicates that the list falls within a range of numeric values.
- e. If you chose string, tag, or tag extent as the type, a **Reference** box appears. You can use this to obtain values for the LOV from a class attribute, such as **item_id**. Click the **Browse** button to the right of the **Reference** box to choose the class and attribute to use for the LOV values. Select the attribute in the pane at the bottom of the dialog box.
- If a class attribute is specified for a string usage or tag LOVs, click the **Load** button to obtain values from the database and populate the table with items from the attribute. The Teamcenter Repository Connection wizard prompts you to log on to a server to look up its available attribute values.
- If a class attribute is specified for a tag extent, the actual values of the LOV are computed at run time when the user attaches this LOV to a property. For example, if the class name is **Item** and the attribute name is **object_name**, while attaching this LOV to a property, the

values are dynamically generated by getting the object names of items in the database.

- f. Click the **Add** button to add values to the LOV, and type descriptions for the LOVs if desired. The values are added to the **LOV Values** table. Change the LOVs as desired by using the **Remove**, **Move Up**, **Move Down**, and **Clear** buttons.

If you chose the **Range** usage, fill in the high and low values for the range in the **Upper** and **Lower** boxes.

- g. To create cascading LOVs, select the **Show Cascading View** check box. Click the **Add Sub LOV** button to add existing LOVs to the list.

- h. When you are done creating the LOV, click **Finish**.

The new LOV appears in the **LOV** folder.

- 6. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **lovs.xml** file, the changes are saved in that file.

To display the LOV in the Teamcenter rich client or thin client user interface, you must attach it to a property on a business object.

Attach an LOV to a property

To display an LOV in the user interface, you must attach it to a property on a business object. For example, if you want to use an LOV to list possible descriptions for an item, attach the LOV to the **object_desc** property of the **Item** business object.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.

2. In the **Extensions** view, select the project in which you want to make the change.

Right-click the project and choose **Organize→Set active extension file**. Select the **lovs.xml** file as the file in which to save the data model changes.

3. Expand the project folder until you see the **LOV** folder displayed.
4. In the **LOV** folder, right-click the LOV you want to attach and choose **Open**. The LOV details appear in a new view.
5. Scroll to the bottom of the view and click the **Attach** button under the **LOV Attachments** heading.
6. Perform the following steps in the **Property Selection** dialog box:
 - a. Select the business object with the property to which you want to attach the LOV.
 - b. In the **Properties** table, select the property to which you want to attach the LOV.
 - c. Click **Finish**.

The LOV is attached to the property, and appears on the **Property Attachments** table for the LOV.

7. To attach the LOV to another property, click the **Attach** button.

To remove the LOV from a property, click the **Detach** button.

For interdependent LOV attachment, click the **Edit** button.

Interdependent attachment is used to attach an LOV value description and sub-LOV values to a cascading LOV.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **lovs.xml** file, the changes are saved in that file.

9. To see the LOV attachment on the business object property, open the **Business Modeler IDE** perspective, right-click the business object, choose **Open**, click the **Properties** tab, and locate the property on the properties table.

The LOV attachment appears in the **LOV** column.

Note

You can also attach an LOV to a property on an opened business object by right-clicking the property and choosing **Edit→LOV→Attach LOV**.

10. After you attach the LOV, you can Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
11. After deployment, test your newly attached LOV in the Teamcenter rich client by creating an instance of the business object and clicking the arrow in the property box where the LOV is attached.

The list of values appears.

For example, if you attached an LOV to the **object_desc** property of the **Item** business object, in the My Teamcenter application, choose **File→New→Item** to create an instance of the **Item** business object. After you have created the business object, notice that in the **Summary** or **Viewer** tab that the **Description** box has a dropdown arrow. When you click the arrow in the **Description** box, your new list of values appears.

Add, remove, or clear a value to an LOV

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project where the LOV was created.
Right-click the project and choose **Organize→Set active extension file**.
Select the **lovs.xml** file as the file in which to save the data model changes.
3. Expand the project folder until you see the **LOV** folder displayed.
4. Select the LOV to modify in the **LOV** folder. The LOV values appear.
5. Enter the following information in the **LOV** dialog box:
 - a. Click the **Add** button to add values to the LOV, and type the values in the **Value** column of the table. Type descriptions for the LOVs if desired. Change the LOVs as desired by using the **Remove**, **Move Up**, **Move Down**, and **Clear** buttons.

If you chose the **Range** usage, fill in the high and low values for the range in the **Upper** and **Lower** boxes.
 - b. When you are done modifying the LOV, click **Finish**.
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **lovs.xml** file, the changes are saved in that file.

LOV variations

There are two variations to creating LOVs.

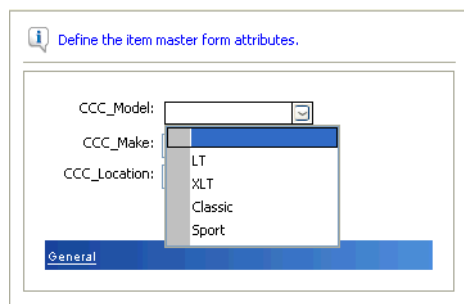
- **Filter LOVs**

Filter LOVs allow you to determine values for one LOV by a value selected in another LOV.

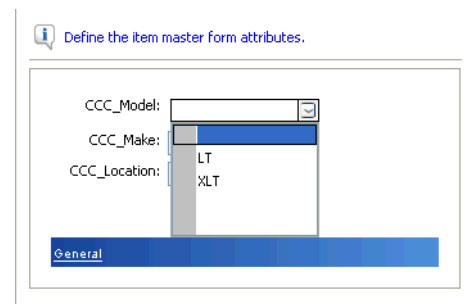
Example

The original, unfiltered list contain all car models. A new list was created using only a subset of the original list.

Unfiltered LOV



Filtered LOV

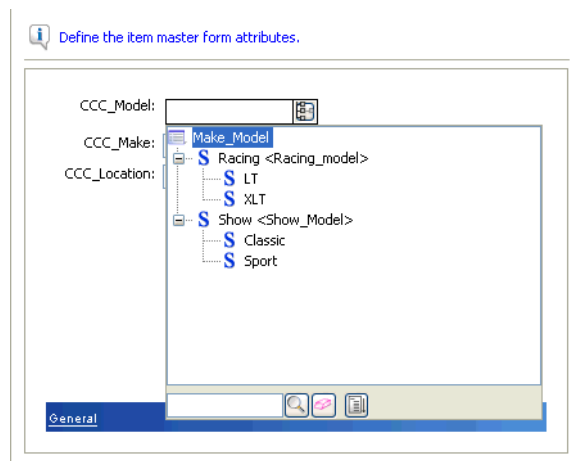


- **Hierarchical LOVs**

Cascading LOVs allow you to organize and present your lists in a hierarchy. This is especially beneficial for long lists and allows for end users to search for a value in the list.

Example

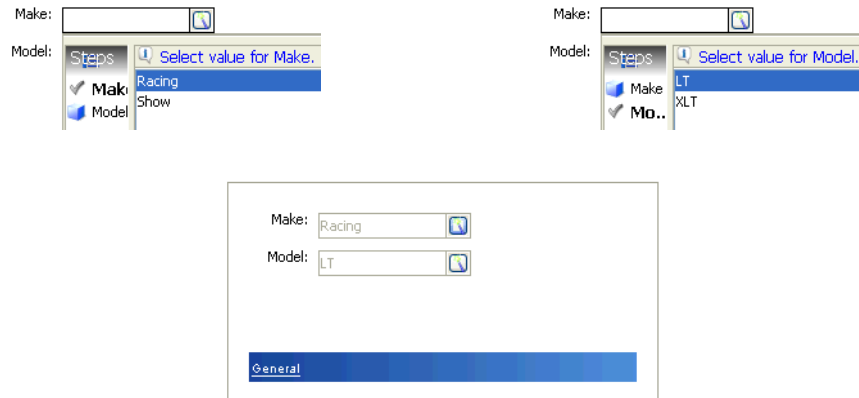
The list contains all car models organized by make, then by model.



An interdependent attachment to a Cascading LOV lists the appropriate property values when you select a value higher in the LOV tree.

Example

When the car make is selected, the next dialog box displays the corresponding models. Select the desired model and then click **Finish** to populate the fields.



Create a filter LOV

To create a filter LOV, you must create a dependency to a value in another LOV.

1. Create an LOV with a type of **Filter**.
2. Click **Browse** to find a base LOV.
3. Select **Direct Dependency** to show and select all the values from the base LOV.
4. Select **All** to show all the values of the original LOV or **Selected** to show only the selected values.
5. Select the values you want to use for the new LOV.

New LOV
Create a new LOV

Project: Customer
Name: Racing_model
Description:
Type: Filter
Usage: ☒ Exhaustive ☐ Suggestive ☐ Range

Filter LOV Options
Based On LOV: Model Browse...
☐ Direct Dependency Show: Selected

LOV Values
☐ Show Cascading View

| Value | Description |
|---|-------------|
| <input checked="" type="checkbox"/> LT | |
| <input checked="" type="checkbox"/> XLT | |
| | |
| | |
| | |
| | |
| | |

Select All Deselect All

? Finish Cancel

Creating a cascading LOV

A cascading LOV (also known as a hierarchical LOV) is an LOV whose values have their own sub-LOVs. An example would be a list of car makes that each contain a list of models.

New LOV

LOV
Create a new LOV

Project: Customer

Name: ccc_models

Description: Classic Car Company car models

Type: String

Usage: ☒ Exhaustive ☐ Suggestive ☐ Range

Reference Class and Attribute

Reference: Browse...

LOV Values

☒ Show Cascading View

| Value | Description |
|--|-------------------------|
| <input checked="" type="checkbox"/> Best of Show(ccc_best_of_show) | Best of Show models |
| Cruiser | Cruiser show model |
| Low Rider | Low Rider show model |
| <input checked="" type="checkbox"/> High Performance(ccc_high_performance) | High Performance models |
| Drag | Drag racing model |
| Racer | Track racing model |

Add Sub LOV
Remove Sub LOV
Expand All
Collapse All

Finish Cancel

To see the cascading values on an existing LOV:

1. Right-click the LOV in the **Extensions** view.
2. Choose **Open**.
3. Select the **Show Cascading View** check box.

Note

Cascading LOVs do not display unless you select the **Show Cascading View** check box.

To create a cascading LOV:

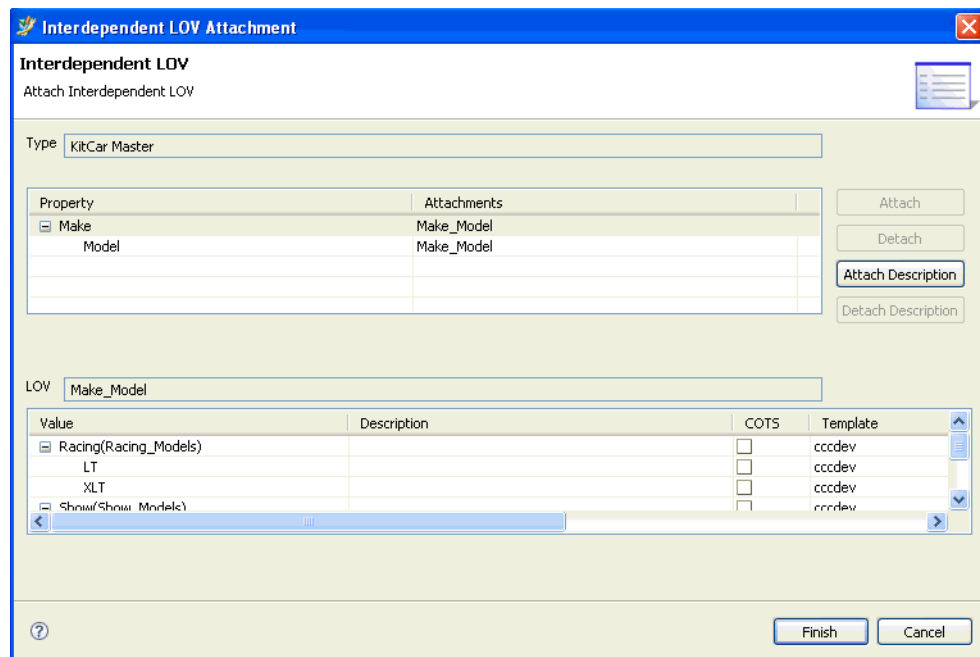
1. Right-click the **LOV** folder in the **Extensions** view.
2. Choose **New LOV**.
3. Fill in the properties.
4. Add LOV values.

5. In the LOV wizard, select the **Show Cascading View** check box.
6. Select an LOV value.
7. Click the **Add Sub LOV** button.
8. Select the **LOV** to add as the subordinate.

Create a Cascading LOV with an interdependent attachment

To create a Cascading LOV with an interdependent attachment, you must build a tree in the table that represents each level of the cascading LOV.

1. Create a cascading LOV and attach it to a property.
2. Edit the property in the **LOV Attachments** table to see the **Interdependent LOV** dialog box.
3. Attach the properties.



The dialog box titled "Interdependent LOV Attachment" is used to attach an interdependent LOV to a property. It contains the following sections:

- Type:** A text field containing "KitCar Master".
- Property Attachments Table:**

| Property | Attachments |
|----------|-------------|
| Make | Make_Model |
| Model | Make_Model |
- Buttons:** "Attach", "Detach", "Attach Description", and "Detach Description".
- LOV:** A text field containing "Make_Model".
- LOV Table:**

| Value | Description | COTS | Template |
|-----------------------|-------------|--------------------------|----------|
| Racing(Racing_Models) | | <input type="checkbox"/> | cccdev |
| LT | | <input type="checkbox"/> | cccdev |
| XLT | | <input type="checkbox"/> | cccdev |
| Show(Show_Models) | | <input type="checkbox"/> | cccdev |
- Footer:** A help icon (?) and "Finish" and "Cancel" buttons.

Create a cascading LOV

A cascading LOV (also known as a hierarchical LOV) is an LOV whose values have their own sub-LOVs, for example, a list of states that each contain a list of cities.

1. Create a main LOV to hold the sub-LOVs. Then create sub-LOVs that can be used under the main LOV.

For example, create an LOV that contains a list of states, and then create LOVs for each state that contain a list of cities.

2. Right-click the main LOV in the **Extensions** view, choose **Open**, and select the **Show Cascading View** check box on the LOV. Cascading LOVs do not appear unless you select the **Show Cascading View** check box.

3. Select a value on the main LOV and choose **Add Sub LOV**. In the **Find LOV** dialog box, choose the sub-LOV for that value.

For example, in a **States** LOV, select the **California** value and add the **California Cities** sub-LOV. Then select the **Florida** value and add the **Florida Cities** sub-LOV.

After you add the sub-LOVs, they appear in a cascade format in the main LOV.

4. Attach the main LOV to a property by scrolling down to the **LOV Attachments** table and choosing **Attach**.

For example, attach the main LOV to the **Item Master** business object on the **user_data_1** property. Then when you create an **Item** in the Teamcenter rich client, the cascading LOV appears in the **User Data 1** box in the second dialog box of the New Item wizard.

Note

If you want to create an interdependent attachment so that the user is prompted to enter values for each level in the cascading LOV, select the attachment in the **LOV Attachments** table and choose **Edit**.

Activity

1. Create an LOV.

In this activity, you create an LOV whose values correspond to categories of cars sold by the Custom Car Company.

2. Add and remove LOV values.

In this activity, you add two new values and remove one value from the **CCC_Cat** LOV so that the LOV values match the categories of cars sold by the Custom Car Company.

3. Attach an LOV to a property.

In this activity, you attach the **CCC_Cat** LOV to the **Category** property of the **CCC_Car Master** form.

4. Create another LOV.

In this activity, you create an LOV whose values correspond to torque for fasteners. This LOV will be used with a note type option created in a later activity.

5. Create a cascading LOV.

In this activity, you create a cascading LOV by creating a sub-LOV for the **Kit Cars** values of the **CCC_Cat** LOV.

6. Deploy and test LOVs.

In this activity, you deploy and test to make sure the LOVs are attached to the correct property.

Review questions

1. After you create a list of values, you must attach it to a property on a business object.
 - True
 - False
2. What kind of LOV usage indicates that the list contains all possible choices?

Select one answer.

- Exhaustive
- Range
- Suggestive

3. What type of LOV variation allows you to organize and present your lists in a hierarchy?

Select one answer.

- Filtered LOV
- Cascading LOV
- Interdependent LOV

Summary

Topics learned in this lesson:

1. List of Values (LOV) allow the display of a pick list for users.
2. An administrator defines a list of values, attaches it to a property and deploys it to the server.
3. Filter LOVs allow you to determine values for one LOV by a value selected in another LOV.
4. Cascading LOVs allow you to organize and present your lists in a hierarchy.
5. Interdependent LOVs list appropriate property values when you select a value in another LOV property.

Lesson

7 *Datasets*

Purpose

The purpose of this lesson is to describe datasets and tools.

Objectives

After you complete this lesson, you should be able to:

- Define and create datasets.
- Define and create tools.

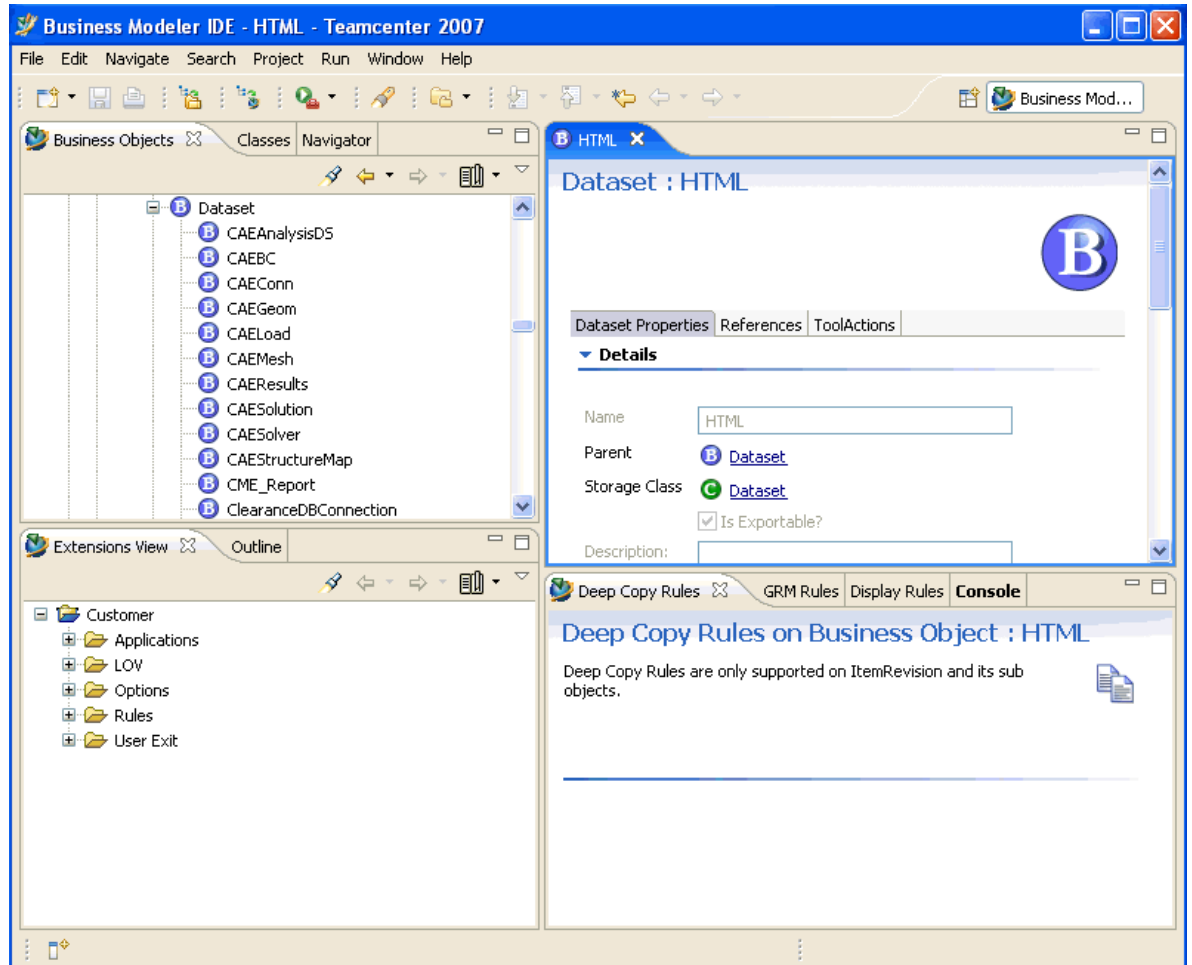
Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

What are dataset business objects

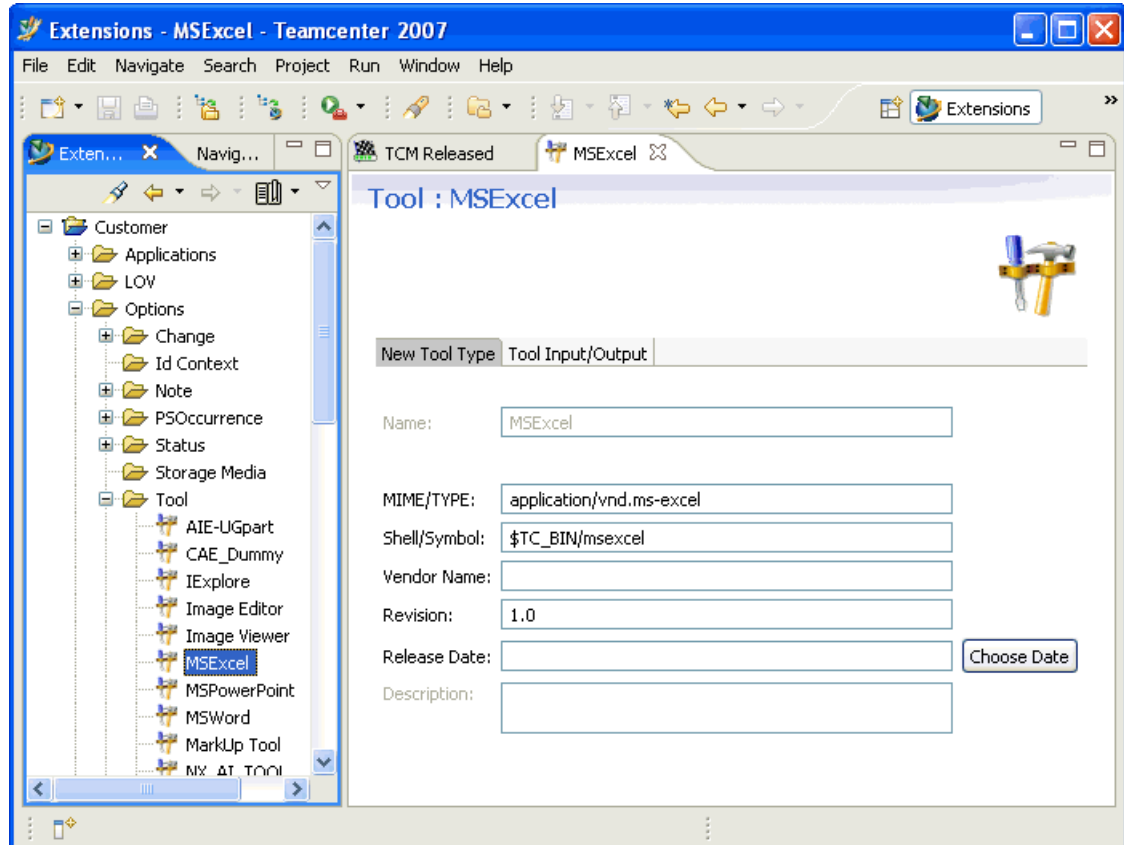
Datasets are objects used to manage file data associated with external software applications. They typically consist of a single application data file or logical groupings of application data files.



There are numerous types of datasets predefined in Teamcenter. However, your site may need to add more business objects to be able to manage your site's specific application data files and the viewing/editing software applications associated with these files.

What is a tool

Tool describes a software application behaving in a specific manner. The Teamcenter tool definition relates to the OS application by specifying the MIME/type definition for the software application on the workstation. Tools are used by dataset business objects.



Add a tool

A tool represents a software application, such as Microsoft Word or Adobe Acrobat. You associate a tool with a type of dataset so you can launch the dataset file from Teamcenter.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new tool object. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new tool object.
3. Expand the project and the **Options→Tool** folders.
4. Right-click the **Tool** folder and choose **New Tool**.
The New Tool wizard runs.

5. Perform the following steps in the **Tool** dialog box:

- a. The **Project** box defaults to the already-selected project.
- b. In the **Name** box, type the name you want to assign to the new tool object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
- c. In the **MIME/TYPE** box, enter the kind of application association. For example, for Acrobat you would type **application/acrobat**, or for Microsoft Word you would type **application/msword**.
- d. If the program is to be run in a shell, in the **Shell/Symbol** box, type the full path and program name to be run in a shell. For example, for Microsoft Word you would type **\$TC_BIN/msword**. When you leave this box empty, the tool is associated with an application but is not run in a shell.
- e. In the **Vendor Name** box, type the name of the application vendor. For example, Adobe is the vendor for Acrobat, and Microsoft is the vendor for Word.
- f. In the **Revision** box, type the version number of the application, for example, **6.0**, **2006**, or something similar.
- g. Click the **Choose Date** button to the right of the **Release Date** box to enter the release date of the application.

- h. In the **Description** box, type a description of the new tool object.
 - i. Click **Next**.
- 6. Perform the following steps on the **New Tool Input/Output** dialog box:
 - a. Click the **Add** button to the right of the **Input** pane to add the type of data the tool accepts, for example, **ASCII** or **Binary**.
 - b. Click the **Add** button to the right of the **Output** pane to add the type of data the tool outputs, for example, **ASCII** or **Binary**.
 - c. Click **Finish**.

The new tool object appears under the **Tool** folder in the **Extensions** view.
- 7. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **options.xml** file, the changes are saved in that file.

Now the new tool is available for use in a dataset. When you create a new dataset business object, you can select the new tool in the **Tools for Edit** or **Tools for View** boxes.

Create a dataset business object

Use the **Dataset** business object to represent a file from a specific software application.

For example, files created in Microsoft Word are represented by the **MSWord** dataset object, text files are represented by the **Text** dataset object, and so on. Associate a **Tool** menu command to each dataset type so that the appropriate software application launches when you open a file in Teamcenter.

1. In the **Business Objects** view, select the project in which you want to create the new **Dataset** business object.
2. Right-click the project and choose **Organize**→**Set active extension file**. Select the **business_objects.xml** file as the file in which to save the data model changes.
3. Click the **Find Business Object** button on the **Business Objects** view toolbar. Type **Dataset** in the search box and click **OK**.

The **Dataset** business object is selected in the **Business Objects** view.

4. In the **Business Objects** view, right-click the **Dataset** business object and choose **New Business Object**.

The New Dataset wizard runs.

5. In the **Dataset** dialog box, enter the following information:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new business object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - c. In the **Parent** box, **Dataset** is already selected as the parent business object.
 - d. In the **Description** box, type a description of the new business object.
 - e. Select the **Advanced** check box only if you want to choose a new class to store data for the business object.

By default, the storage class is set as the same class used by the parent business object. However, if you want to create a new class to store the data, select the **Create primary Business Object** check box. This creates a new class that has the same name as the new business object.

(A *primary* business object has the same name as its associated storage class. A *secondary* business object uses the storage class of its parent business object. Typically, most custom business objects are secondary business objects.)

- f. To the right of the **Tools for Edit** pane, click the **Add** button to select the software application to launch when the dataset is selected in Teamcenter.

Type an asterisk * in the **Find** dialog box to see all existing tools. If the tool does not exist, you must create one using the **Tool** option.

- g. To the right of the **Tools for View** pane, click the **Add** button to select the software application to be used to view the dataset files. If the desired tool does not exist, you must create one using the **Tool** option.
 - h. Click **Next** if you want to add references and parameters to the dataset. Otherwise, click **Finish** to complete the dataset creation.
6. If you clicked **Next**, the **References** table appears. Click the **Add** button to the right of the **References** table.

The Add Dataset Reference wizard runs, allowing you to add file name references to associate with the dataset. Perform the following steps in the **Dataset References** dialog box:

- a. For **Reference**, type a unique name for this file reference.
 - b. For **File Type**, enter the file name extension (for example, **txt**, **pdf**, **doc**, and so on).
 - c. For **Format**, choose whether the files are binary, object, or text.
 - d. Click **Finish**.
7. Click **Finish** on the New Dataset wizard to complete the dataset creation, or if you want to change the action taken when the dataset is launched, click **Next**.
 8. If you clicked **Next**, the **Tool Action** table appears. Click the **Add** button to the right of the **Tool Action** table.

The Add/Modify Dataset Tool Action wizard runs, allowing you to modify the action that a software application takes when a dataset is launched. Perform the following steps in the **Add/Modify Dataset Tool Action** dialog box:

- a. Click the **Browse** button to the right of the **Tools** box to select the tool you previously chose to use for viewing or editing this dataset. (The only available tool is the one you previously selected.

- b. Click the arrow in the **Operations** box to choose the operation to use with the dataset (for example, **Open**, **OpenUsing**, **Print**, **PrintUsing**, or **Send**).

The **OpenUsing** operation allows the user to select the tool to use for opening the dataset, and the **PrintUsing** operation allows the user to select the tool to use to print the dataset.

- c. To set the file name extensions to associate with the action, click the **Add** button to the right of the **References** table.

The Add Reference wizard runs. In the **Reference** dialog box, click the arrow in the **Select the Reference Name** box to choose the file name reference for this tool action. Click **Select** to use this reference, or **Export** to export it to the database.

- d. To add parameters to be passed with the action, click the **Add** button to the right of the **Parameters** box.

The Add Parameter wizard runs. Select a parameter and click **Finish**.

- e. Click **Finish** in the **Add/Modify Dataset Tool Action** dialog box.

- f. Click **Finish** in the **New Dataset** dialog box.

9. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

10. Deploy your changes to a test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

11. After deployment, test your new business object in the Teamcenter rich client by creating an instance of it.

For example, in the My Teamcenter application choose **File→New→Dataset**. Click the **More...** button and choose the new business object from the list of available ones. Create an instance of the new dataset and click **OK**.

Dataset and named references

You can import files created in the native environment and manage these files with datasets. When you import files, a dataset is created with named references to a copy of the original files located in a database volume.

You may import files for these reasons:

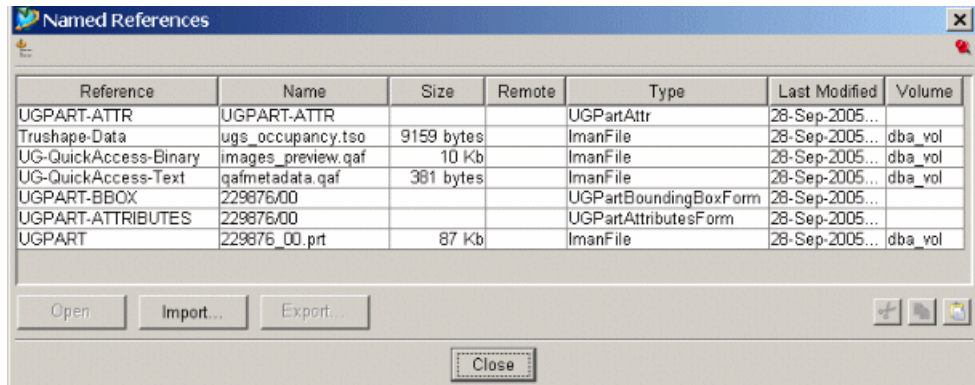
- Import document, images or other data created outside the database
- Migrating legacy data

You can use both interactive and batch mechanisms to import data. You can import data to Teamcenter using:

- Rich client interface
- Using the **import_file** utility for batch import

Using named references

Named References are Teamcenter objects that relate to a specific data file. A single dataset object may have one or more named references. To view the named references of a dataset from rich client, select the dataset and choose **View® Named References**, or right-click the dataset and select **Named References**.



| Reference | Name | Size | Remote | Type | Last Modified | Volume |
|-----------------------|--------------------|------------|--------|-----------------------|----------------|---------|
| UGPART-ATTR | UGPART-ATTR | | | UGPartAttr | 28-Sep-2005... | |
| Trushape-Data | ugs_occupancy.tso | 9159 bytes | | ImanFile | 28-Sep-2005... | dba_vol |
| UG-QuickAccess-Binary | images_preview.qaf | 10 Kb | | ImanFile | 28-Sep-2005... | dba_vol |
| UG-QuickAccess-Text | qafmetadata.qaf | 381 bytes | | ImanFile | 28-Sep-2005... | dba_vol |
| UGPART-BBOX | 229876/00 | | | UGPartBoundingBoxForm | 28-Sep-2005... | |
| UGPART-ATTRIBUTES | 229876/00 | | | UGPartAttributesForm | 28-Sep-2005... | |
| UGPART | 229876_00 prt | 87 Kb | | ImanFile | 28-Sep-2005... | dba_vol |

Open Import... Export... Close

import_file utility

The **import_file** utility allows you to import files in the database. The **import_file** utility provides arguments to associate the ownership and dataset description with the imported files. The arguments may be specified on the command line for a single file import, or in a command file for multiple files (batch) import.

Use the following arguments to import a single ASCII text file with the **import_file** utility:

```
TC_ROOT/bin/import_file -u=user-id -p=password  
-g=group -file=file_name -type=dataset type  
-d=dataset name -ref=named referenced
```

Key points

- You can import one file at a time into a dataset.
- You can subsequently add more files to a dataset.
- You can import various files using the **import_file** utility.
- You set user and group ownership on the created datasets.
- If you do not specify a directory, the dataset appears in the **Newstuff** folder.

Note

To override the default object ownership, use the **-u=user -p=password -g=group** arguments. Make sure the file will be imported to the desired volume.

Example

To import the single file *report.dat* in Teamcenter as a **Text** dataset **report1**, type the following command on a single line:

```
TC_ROOT/bin/import_file -f=report.dat -type=Text  
-d=report1 -ref=Text -vb -log=import.log
```

import_file arguments

| Command line | Results |
|--|--|
| <code>-u=username</code> | Specifies the user ID. This is generally a user with administration privileges. |
| <code>-p=password</code> | Specifies the password. |
| <code>-g=group</code> | Specifies the group <code>group</code> associated with the user. |
| <code>-f=file name</code> | Import a single file. The full path must be provided if the file does not reside in the current working directory. |
| <code>-ref=ref name</code> | Type of named referenced associated with the file. Each dataset type defines one or more named references associated with it. |
| <code>-type=dataset type</code> | Defines the dataset type in Teamcenter, for example, <code>TEXT</code> or <code>UGMASTER</code> datasets. |
| <code>-vol=volume -de={e a r n}</code> | Specifies the full path of the volume where the imported file is placed. The <code>-de</code> option indicates that a dataset exists. Used when a dataset of the same name already exists. |
| <code>-item=item name</code> <code>[-revision=version number]</code> <code>-ie={n y} -desc=item description</code> | Specifies the item revision number and revision ID. The option <code>-ie</code> specifies if the item already exists. |
| <code>-i=import list</code> | Imports multiple files using a specified import file that contains the following information: <pre>-f=bike1.dat -d=my_bike1_dataset -type=UGPART -ref=UGPART -f=bike2.dat -d=my_bike2_dataset -type=UGPART -ref=UGPART</pre> |
| <code>-vb</code> | Runs utility in verbose mode. Displays maximum amount of information. Non-verbose sessions only display error messages. |
| <code>-log=file</code> | Log created datasets. Otherwise <code>-log=-</code> can be used to print to the command window. |

Activity

1. Create a tool.

In this activity, you create a tool and add it to the list of tools for the **Text** dataset.

2. Create a dataset.

In this activity, you create a new dataset that uses Microsoft Word to view and edit associated Microsoft Word **.doc** files.

3. Deploy and test the new dataset.

4. Import files and create datasets.

In this activity, you import files and create new datasets using the **import_file** utility.

Review questions

1. What is true about datasets?

Select all that apply.

- Consists of a single application data file or logical groupings of application data files.
- Describes a software application behaving in a specific manner.
- Objects used to manage file data associated with external software applications.
- One type of dataset is predefined in Teamcenter.

2. Tools must be created after datasets.

- True
- False

3. What are named references?

Select one answer.

- Tool options.
- MIME/type definitions.
- Objects that relate to a specific data file.

4. What is the batch mechanism to import data?

Select one answer.

- The rich client interface.
- The **import_file** utility.
- The Business Modeler IDE.

Summary

Topics learned in this lesson:

1. Purpose and implementation of new datasets.
2. How tools support datasets.
3. How to import files to the database.

Lesson

8 *Options*

Purpose

The purpose of this lesson is to describe options and how they are managed.

Objectives

After you complete this lesson, you should be able to:

- Define options.
- Define and create notes.
- Define and create statuses.
- Define and create units of measure.
- Define and create views.
- Define and create changes.
- Add options to the data model.

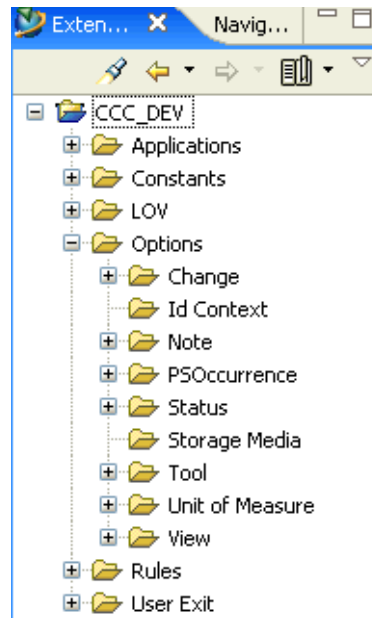
Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

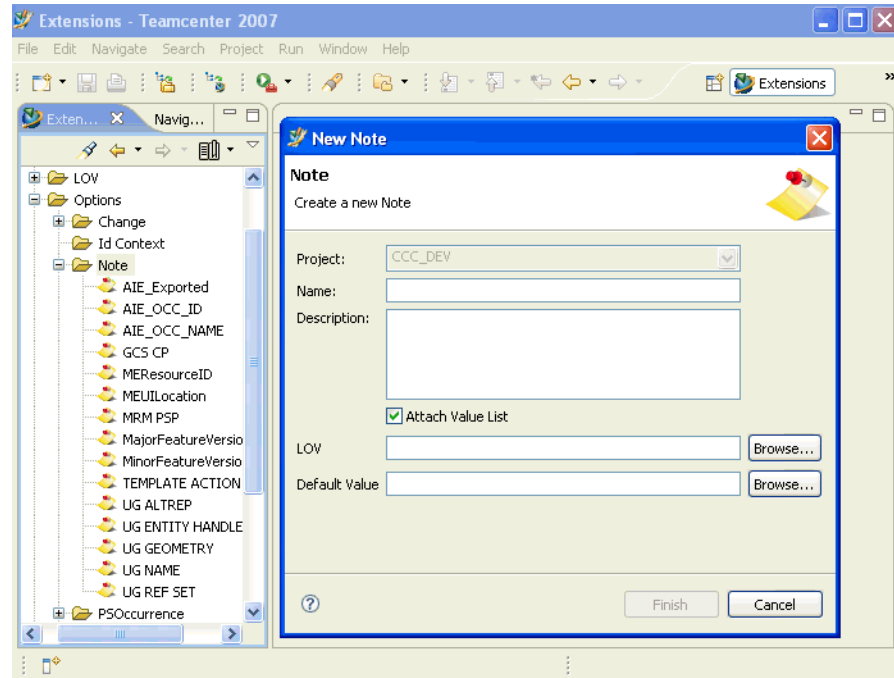
What are options

Whereas business objects represent parts, documents, and other design objects, *options* represent configurations you can do to business objects. For example, a change item tracks a change to a business object, a status item designates the status of a business, a view item holds structure information for a business object, and so on.



What is a note

Notes are associated with an occurrence in a PSE bill of materials (BOM). Users can then specify a value for any note that is defined for the site and attach a list of values for the users.



Key points

- The occurrence notes that are defined are listed in the **PSE Columns**, **BOMLine Properties**, and **Notes Editor** dialog box. The user can use any of these dialog boxes to enter a value for a particular occurrence.
- **Attach Value List** is initially set to **Yes**. If cleared, the dialog box changes to remove the **LOV** and **Default Value** boxes for the note. LOV must be of type **String** to use with the **Note**.

Warning

The initial list of notes shown are standard notes supplied with the system that are required for Teamcenter Manufacturing and for synchronizing object attributes from NX. These should not be deleted.

Add a note

A note is an object associated with a product structure occurrence in a PSE bill of material (BOM).

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new note object. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new note object.
3. Expand the project and the **Options→Note** folders.
4. Right-click the **Note** folder and choose **New Note**.

The New Note wizard runs.

5. Perform the following steps in the **Note** dialog box:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new note object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - c. In the **Description** box, type a description of the new note object.
 - d. Select the **Attach Value List** check box if you want to attach a value to the note from a list of values (LOV).
 - e. If you selected the **Attach Value List** check box, click the **Browse** button to the right of the **LOV** box to locate the list of values. Type an asterisk * in the **Find** dialog box to see all possible selections.

Click the **Browse** button to the right of the **Default Value** box to choose the value from the list of values that you want to attach to the note.
 - f. Click **Finish**.

The new note object appears under the **Note** folder in the **Extensions** view.

In addition, a new property with the same name as the new note appears on the **BOMLine** run-time business object. If you specified an LOV with the note, the new property automatically has the LOV attached to it.

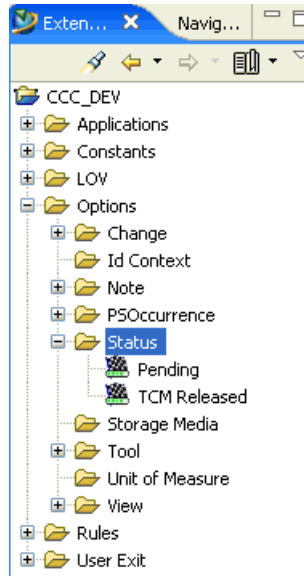
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **options.xml** file, the changes are saved in that file.
7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. After deployment, test your note in the Teamcenter rich client by viewing **BOMLine** properties in Product Structure Editor:
 - a. In the My Teamcenter application, right-click any item or item revision and choose **Send To→PSE**.
 - b. In the PSE application, right-click the item and choose **Properties**.
 - c. Scroll to the bottom of the **Properties** dialog box and click **More**.
All the **BOMLine** properties appear.
 - d. Scroll to the bottom of the **Properties** dialog box.
Your new note appears at the bottom of the dialog box.

You can add an instance of your new note to any child item in the PSE application. Select the child item, choose **View→Notes**, click the arrow in the **Create** box and choose your new note option.

What is a status

Status (or release status) can be set on almost any Teamcenter data to designate a release state. An object's properties reflect the status by name and the release status date.



Key points

- After parts are released, Product Structure Editor (PSE) can be used to display different bills of materials (BOMs) in different stages of development.
- PSE can also display Work-in-Process parts, which are those that have not had a final release status applied.
- PSE (and the NX integration) can load assemblies based on the release status list of status.

Add a status

A status is applied to an object after it goes through a workflow. Typical statuses are **Released** and **Approved**.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new status object. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new status object.
3. Expand the project and the **Options→Status** folders.
4. Right-click the **Status** folder and choose **New Status**.
The New Status wizard runs.
5. Perform the following steps in the **Status** dialog box:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new status object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - c. In the **Description** box, type a description of the new status object.
 - d. Click **Finish**.

The new status object appears under the **Status** folder in the **Extensions** view.
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **options.xml** file, the changes are saved in that file.
7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. After deployment, test your new status in the Teamcenter rich client:
 - a. In the Workflow Designer application, choose **File→New Root Template**.

- b. In the **New Root Template** dialog box, click the arrow in the **Based on Root Template** box, choose **TCM Release Process**, and click **OK**.
The new process appears.
- c. In the upper left pane, select **Add Status Task (TCM Released)**.
- d. In the lower left pane, click the **Display the Task Attributes Panel** button.
- e. In the **Attributes** dialog box, click the arrow in the **Release Status** box.
Your new status appears in the list.
Select your new status and close the **Attributes** dialog box.
- f. In the **Name** box in the lower-left pane, change **Add Status Task (TCM Released)** to something that describes your new status, for example, **Add Status Task (My Released)**.

What is a unit of measure

Unit of measure (UOM) is created so that items and item revisions can be expressed in standardized units (for example, inches, millimeters, and so forth) across an entire Teamcenter site. There are no unit of measures provided.

Choose **File**→**New**→**Item** from the rich client and open the **New Item** dialog box. The **Unit of Measure** box displays and lists the units of measure defined in the system.

Key points

- By default, items have no units of measure. This implies that item quantities are expressed in terms of each or pieces. In other words, they refer to a discrete number of component parts.
- Additional units of measure may be needed to define an accurate bill of materials (BOM).
- When a user chooses the selector in the **Unit of Measure** box of either the **New Item** or **Properties** dialog boxes, the user is restricted to entering one of the predefined values.
- The **Symbol** box is for the unit of measure symbol.

Note

In PSE, if no specific quantity value is associated with items; the default quantity is each (one component).

Note

In PSE, if unit of measure is anything other than null, the component does not open in NX.

Add a unit of measure

A *unit of measure* is a measurement category (for example, inches, millimeters, and so on). Create a unit of measure (UOM) when you need a new measurement for users.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new unit of measure. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new unit of measure.
3. Expand the project and the **Options→Unit of Measure** folders.
4. Right-click the **Unit of Measure** folder and choose **New Unit of Measure**. The New Unit of Measure wizard runs.
5. Perform the following steps in the **Unit of Measure** dialog box:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new unit of measure.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - c. In the **Symbol** box, enter the unit (for example, **in** for inches, **oz** for ounces, and so on).
 - d. In the **Description** box, type a description of the new unit of measure.
 - e. Click **Finish**.

The new unit of measure appears under the **Unit of Measure** folder in the **Extensions** view.
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **options.xml** file, the changes are saved in that file.
7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

8. After deployment, test your new unit of measure in the Teamcenter rich client. The new unit of measure is now available when you create a new item or item revision.

For example, in the My Teamcenter application, choose **File→New→Item**. In the **New Item** dialog box, click the arrow in the **Unit of Measure** box.

Your new unit of measure appears in the list.

Activity

1. Create a note.

In this activity, you create a note that gets associated with an occurrence in a PSE bill of materials (BOM). This note attaches an LOV so it displays a predefined list of values the user can select for the note on the BOM.

2. Create statuses.

In this activity, you create statuses to be used in a product development workflow. Common workflow tasks include approving and releasing parts. Statuses are used to help track the state of parts.

3. Create units of measure.

In this activity, you create units of measure to represent inches and ounces for business objects.

4. Deploy new options.

5. Test new options.

In this activity, you verify that the new note and units of measure were deployed to the server. The new statuses will be tested in a later activity.

Review questions

1. What is a note?

Select one answer.

- Associated with an occurrence in a PSE bill of materials (BOM).
- Can be set on almost any Teamcenter data to designate a release state.
- Created so that items and item revisions can be expressed in standardized units.
- Defines when you use unique item IDs.

2. What is a status?

Select one answer.

- Associated with an occurrence in a PSE bill of materials (BOM).
- Can be set on almost any Teamcenter data to designate a release state.
- Created so that items and item revisions can be expressed in standardized units.

- Defines when you use unique item IDs.

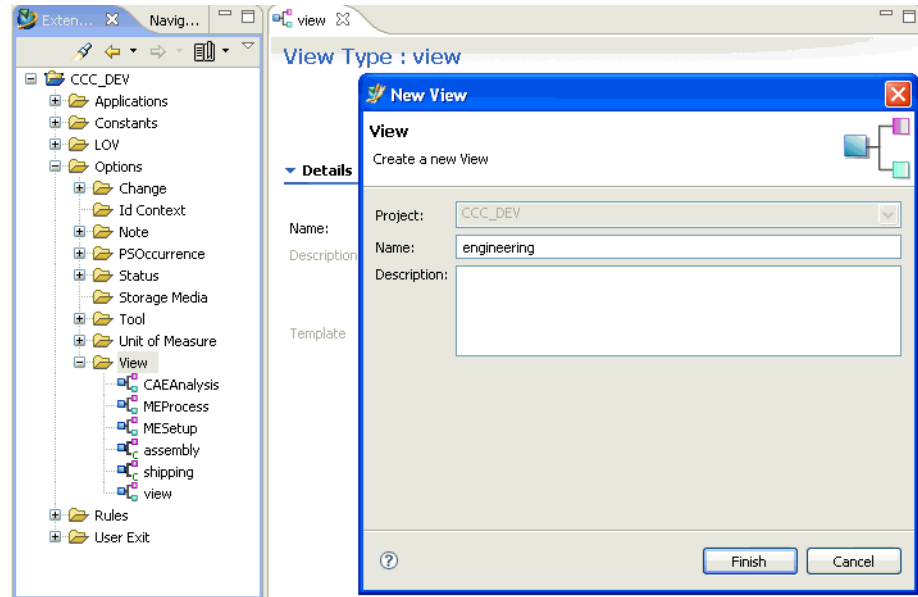
3. What is a unit of measure?

Select one answer.

- Associated with an occurrence in a PSE bill of materials (BOM).
- Can be set on almost any Teamcenter data to designate a release state.
- Created so that items and item revisions can be expressed in standardized units.
- Defines when you use unique item IDs.

What is a view

View is a definition that controls the name of a **PSView** object. The **PSView** object works with an item and item revision to maintain product structure information in Teamcenter.



Key points

- The default configuration is a single **PSView** object defined for the entire site called **view**.
- Multiple view types support different groups and users to use the most appropriate **PSView** to model their data. For example, consider a site that has used PSE primarily for modeling engineering data. If transitioning to multiple **PSView** objects, this site should consider renaming **view** to something that is more meaningful in the context of multiple **PSView** objects and that it also appropriately describes the legacy product structure data. In this case, **engineering** is a good choice. This allows another **PSView** object, such as **shipping**, to be defined.
- Use preferences to define the default view.
 1. Choose **Edit**→**Options**→**PSE**.
 2. Set **PSE_default_view_type** to **view**.

Note

If you rename and/or add views, this preference may need to be changed to another value.

Add a view

A *view* is a BOM view revision (BVR) category. View options control the name of a product structure view object. The product structure view options work with the item and item revision business objects to maintain product structure information.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new view object. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new view object.
3. Expand the project and the **Options→View** folders.
4. Right-click the **View Type** folder and choose **New View**.

The New View wizard runs.

5. Perform the following steps in the **View** dialog box:
 - a. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new view object.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - c. In the **Description** box, type a description of the new view object.
 - d. Click **Finish**.

The new view object appears under the **View** folder in the **Extensions** view.

6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

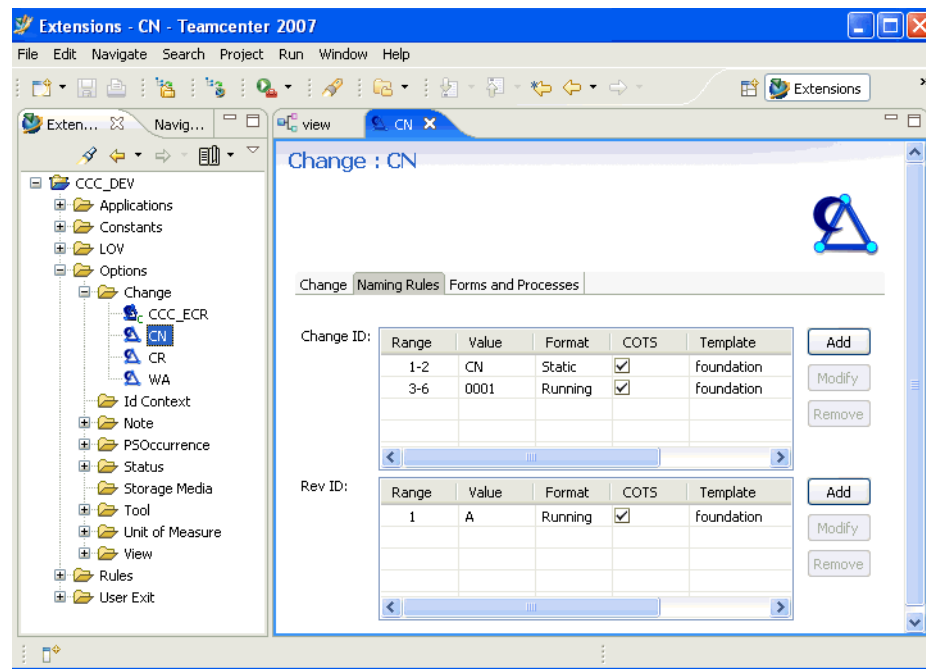
If you set the active extension file as the **options.xml** file, the changes are saved in that file.

7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. After deployment, test your new view option in the Teamcenter rich client.

For example, in the My Teamcenter application select an item or item revision and choose **File→New→BOMView Revision**. In the **New BOMView Revision** dialog box, click the **More...** button in the lower left and choose the new view object from the list of available view objects. Create an instance of the new view object and click **OK**.

What is a change

Most companies have different numbering conventions and signature approval levels for different types of changes. These are referred to as *change classifications*.



Key points

- Each change has its own unique numbering convention.
- Three change items are provided with the default Teamcenter installation:
 - **CN**
Modeled using CMII change notice (CN) process requirements.
 - **CR**
Modeled using CMII change request (CR) process requirements.
 - **WA**
Work Authorization (WA)
- Change naming rules format has two selections:
 - **Running**
You want this value to increment.
 - **Static**
You do not want this value to change.

- You can change the setup of these change or new change items can be created based on the requirements of your business.

Add a change

A *change* is an option that can accept a temporary or permanent alteration to its configuration, documentation, or design requirements.

When you create a change option you must associate it with a workflow process template on a server. When you install a solution containing this change option, ensure that the server you are installing to has the needed workflow process template.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new change. Right-click the project and choose **Organize→Set active extension file**. Select the **options.xml** file as the file in which to save the new change.
3. Expand the project and the **Options→Change** folders.
4. Right-click the **Change** folder and choose **New Change**.

The New Change wizard runs.

5. Perform the following steps in the **Change** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new change item.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - b. In the **Description** box, type a description of the new change.
 - c. Select the **Is Effectivity Shared** check box if revisions of this type of change are each allowed to have a separate effectivity.
 - d. Click **Next**.

6. Use the **Change Naming Rules** dialog box to define how change items are to be named. Define the change and revision IDs section-by-section, left-to-right, based on the number and behavior of the characters.

The change ID is a string of up to 32 characters. The string can be a combination of static and running characters. For example, the static section of the ID could be **PartECR** and the running section of the ID could be **001**. Thus, the first change ID for this particular change object would be **PartECR001**. The second ID would be **PartECR002**, and so on.

The revision ID is a string of up to 10 characters. The string can be a combination of static and running characters. For example, the static section of the ID could be **Rev** and the running section of the ID could be **A**. Thus, the first revision ID for this particular change object would be **/RevA**. The second ID would be **/RevB**. When a change is listed in the change tree, a slash (/) is placed between the change ID and the revision ID by default.

Some valid scenarios follow:

Table 8-1. Valid scenarios

| Range | Value | Format |
|--------------|--------------|---------------|
| 1 | A | Static |
| 2-6 | 00001 | Running |
| 7 | B | Static |

Here are some invalid scenarios:

Table 8-2. Invalid scenarios

| Range | Value | Format |
|--------------|--------------|---------------|
| 2 | AB | Static |
| 3-6 | 0001 | Running |

- a. Click the **Add** button to the right of the **Change ID** table.
 - 1) In the **Range** cell, type the numeric range of the ID.
For example, if you want the value of the ID to be **PartECR**, type **1-7** to match the number of characters in **PartECR**.
 - 2) In the **Value** cell, type the name portion of the ID, for example, **PartECR**.
 - 3) Click in the **Format** cell and use the arrow to select **Static** or **Running** as the type of this portion of the change ID.
For example, select **Static** if the value is **PartECR**, because you do not want this value to change.
 - 4) If desired, click the **Add** button to the right of the **Change ID** table to put in additional IDs.
For example, for the **PartECR** example, click the **Add** button again to the right of the **Change ID** table, and enter **8-10** in the **Range** box, enter **001** in the **Value** box, and choose **Running** for **Format**.

- b. Click the **Add** button to the right of the **Rev ID** table:
 - 1) In the **Range** cell, type the numeric range of the ID.

For example, if you want the value to be **Rev**, type **1-3** to match the number of characters in the value.
 - 2) In the **Value** cell, type the name portion of the ID, for example, **Rev**.
 - 3) Click in the **Format** cell and use the arrow to select **Static** or **Running** as the type of this portion of the revision ID.

For the **PartECR** example, choose **Static**.
 - 4) If desired, click the **Add** button to the right of the **Rev ID** table to put in additional IDs.

For example, for the **Part ECR** example, click the **Add** button and enter **4-6** in the **Range** box, enter **001** in the **Value** box, and enter **Running** in the **Format** box.
 - c. Click **Next**.
7. Enter the following in the **Change Forms and Processes** dialog box:
 - a. Click the **Add** button to the right of the **Form Objects** pane to choose the Form business object to hold information for the new change object. Type an asterisk * in the **Find** dialog box to see all possible selections.
 - b. Click the **Add** button to the right of the **Process Templates** pane.

The Teamcenter Repository Connection wizard prompts you to log on to a server to look up a workflow process template to use for the change.

Change objects are associated with process templates because workflows track change. Process templates are created using the Workflow Designer application.
 - c. Click **Finish**.

The new change object is created and appears in the **Change** folder in the **Extensions** view.
 8. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **options.xml** file, the changes are saved in that file.

9. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
10. After deployment, test your new change in the Teamcenter rich client:
 - a. In the My Teamcenter application, choose **File**→**New**→**Change**.
 - b. Click the **More** button in the lower left of the **Create Change** dialog box and choose your new change option from the menu.
 - c. Click the **Assign** button to the right of the **Change ID** box. The change ID value that you created earlier now appears in the box.

Also notice how the revision ID you created earlier now appears in the **Change Revision ID** box.
 - d. Finish entering data and click **OK** to create an instance of the change.

Activity

1. Create views.

In this activity, you create views to support Product Structure views.

2. Create a change.

In this activity, you create a new **CCC_ECR** change item to represent an engineering change request for the Classic Car Company.

3. Deploy new options.

4. Test new options.

In this activity, you verify that the new change item and views were deployed to the server.

Review questions

1. A view defines what?

Select one answer.

- Name of **BOMView** objects
- Name of **Change** objects
- Name of product structure objects
- Name of **PSView** objects

2. What are valid naming rules formats?

Select all that apply.

- Fixed
- Incremental
- Running
- Static

Summary

Topics learned in this lesson:

1. Use options to configure business objects.
2. Define notes for product structure occurrences to better describe the occurrence.
3. Define statuses to better designate release states on Teamcenter data.
4. Define units of measure to express standardized units for items.
5. Define views to help maintain product structure information.
6. Define changes to provide your own change classifications.
7. Deploy options to the data model.

Lesson

9 *Constants*

Purpose

The purpose of this lesson is to define and manage constants.

Objectives

After you complete this lesson, you should be able to:

- Describe constants.
- Describe and create global constants.
- Describe and create business object constants.
- Describe and create property constants.

Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining constants

Constants are configuration points within the data model. They can be used to set everything from user interface appearance to action behavior.

You can override an existing constant by setting a different value. Keep in mind that another template could override your setting. The last template loaded takes precedence.

There are three different kinds of constants:

- **Global** constants
Provide consistent definitions that can be used throughout the system.
- **Business object** constants
Provide default values to business objects.
- **Property** constants
Provide default values to business object properties.

All constants are one of three data types:

- **String**
Defines a text string.
- **List**
Contains a list of values.
- **Boolean**
Allows two choices to the user (true or false).

Constants framework

The *constants framework* supplies customers an easy way to extend and configure the system.

When you create a new constant, you must also add the code on the server to return the constant's value to the caller, so the caller can branch the business logic based on the returned value.

- It is simple, powerful, object oriented, extensible, and configurable.
- An easy way to expose a choice or configuration point.
- Unlike the other business rules which are specific to areas of the product, (for example, display rules, naming rules, deep copy rules), the constants framework can be used anywhere you need it.

Constants precedence

Constants have an order of precedence controlled by the templates.

- Each template can override the constant value of any previous template.
- The last constant to override takes precedence.
- Declaring dependencies ensures that your template is appended after the dependencies.

Each column in the diagram represents the deployed templates. Each template defines a different value for the background color constant. The bottom row shows what constant takes precedence.

| | | |
|---|---|---|
| | | Partner BackgroundColor Default Value="red" |
| | | - |
| | CCC_DEV BackgroundColor Default Value="yellow" | CCC_DEV BackgroundColor Default Value="yellow" |
| | - | - |
| Foundation BackgroundColor Default Value="green" | Foundation BackgroundColor Default Value="green" | Foundation BackgroundColor Default Value="green" |
| | | |
| BackgroundColor = "green" | BackgroundColor = "yellow" | BackgroundColor = "red" |

In column 1, only the foundation template is applied to the default background color and sets the color to green.

In column 2, the **CCC_DEV** template overrides the foundation template's default background color and sets the color to yellow.

In column 3, the **Partner** template overrides the **CCC_DEV** template's default background color and sets the color to red.

Global constants

Global constants provide consistent definitions that can be used throughout the system. These constants have only one value, either the default value or the value you set.

Global constants:

- Are global to the system.
- Can be overridden.
- Can supply a default value.

Here are some hypothetical examples:

- What folder names should be shown in the user's Home folder?
- If customers want to brand the thin client with their own company image, what image name should be used?
- Should the files be removed on checkin?
- What color should be used to represent a BOM change type?
- Should Teamcenter send event notifications to ERP?
- How many seconds before the client application should autotime out?
- What character should be used as the separator between the item number and revision?
- Should functionality *x* be enabled?

Create a global constant

Global constants provide consistent definitions that can be used throughout the system. These constants have only one value, either the default value or the value you set. When you create a new constant, you must also add the code on the server to return the constant's value to the caller, so the caller can branch the business logic based on the returned value.

You can create global constants for a number of situations: set the folder names for the Home folder, set how many seconds before the client application should time-out, enable functionality, and so on.

For example, a global constant called **ActiveWindows** has a list of three possible choices, **1**, **2**, or **3**, with a default value of **1** in the Foundation template. Once a constant is set, it can be overridden by another template. Another template can set the default value to **2**, and yet another template can set it to **3**. The rule is that the last template that gets installed determines the constant value that is used.

The server side code can use the following published ITK to retrieve a global constant value:

```
int CONSTANTS_get_global_constant_value (
    const char*      constant_name,    /* <I> */
    char **          value             /* <OF> */
);
```

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new constant. Right-click the project and choose **Organize→Set active extension file**. Select the **business_objects.xml** file as the file in which to save the new constant.
3. Expand the project and the **Constants→Global Constants** folders.
4. Right-click the **Global Constants** folder and choose **New Global Constants**.

The New Global Constants wizard runs.

5. Perform the following steps in the **Create Global Constant** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new constant.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- b. In the **Description** box, type an explanation of how the constant is to be used.
 - c. Click the arrow in the **Data Type** box to choose one of the following:
 - **String**
Indicates that the value is a text string.
 - **Boolean**
Allows two choices to the user (true or false).
 - **List**
Contains a list of values.
 - d. If you chose the **List** data type, a **Values** table appears. Click the **Add** button to add values to the list:
 - 1) In the **Value** box, type a value for the list.
 - 2) Select **Secured** to prevent the selected value from being overridden by another template.
 - 3) Click **Finish**.
 - e. In the **Default Value** box, enter the initial value of the constant. Entry differs depending on the data type you previously chose:
 - If you selected the **String** data type, type in the default value.
 - If you selected the **Boolean** data type, click the dropdown arrow to select **true** or **false** for the default value.
 - If you selected the **List** data type, click **Browse** to select a value from the available ones on the list.

Note

If the selected default value is marked as **Secured** then this value cannot be overridden by any other template.
 - f. Click **Finish**.

The new constant appears under the **Global Constants** folder in the **Extensions** view.
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.
- If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. To verify the constant on the server, run the **getglobalconstantvalue** utility. This utility tests the value of the constant in the database. The utility accepts the name of the global constant and outputs the value of the constant if present.

Business object constants

Business object constants provide default behavior to business objects. Because these constants are attached to business objects, they are inherited, just like properties.

Business object constants:

- Are applicable to business objects.
- Can be overridden.
- Support inheritance of the constant value in the business object hierarchy.
- Can supply a default value.
- Can be limited to be available on specific business object(s).

Here are some hypothetical examples:

- How many revisions should an item be allowed to have?
- When a part is revised, should the related document objects also be revised?
- What icon should this business object show in the client?
- When this business object reaches a step in the workflow, how should the workflow branch?
- Should a master form be automatically created for this Item?
- What should the display name of the business object be?

Create a business object constant

Business object constants provide default values to business objects. Because these constants are attached to business objects, they are inherited, just like properties. You can define a constant to have a specific scope so that it is available only on certain business objects. This ensures that server API can retrieve the value properly on just those business objects. When you create a new constant, you must also add the code on the server to return the constant's value to the caller, so the caller can branch the business logic based on the returned value.

You can create business object constants for a number of situations: set the display name of the business object, set the maximum number of allowed revisions for an item, define the icon to be used for the business object in the UI, and so on.

For example, create a business object constant to show an icon for Microsoft Word datasets. It could be called **IconName** and have a scope of **MSWord**, a data type of **String**, and a default value of **MSIcon.png**. Then a My Teamcenter application developer could link the **MSWord.IconName=MSIcon.png** value so that the icon is displayed for Microsoft Word datasets in My Teamcenter.

The server side code can use the following published ITK to retrieve a business object constant value:

```
int CONSTANTS_get_type_constant_value (
    const char*    constant_name,      /* <I> */
    const char*    type_name,         /* <I> */
    char**         value               /* <OF> */
);
```

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new constant. Right-click the project and choose **Organize→Set active extension file**. Select the **business_objects.xml** file as the file in which to save the new constant.
3. Expand the project and the **Constants→Business Object Constants** folders.
4. Right-click the **Business Object Constants** folder and choose **New Business Object Constants**.

The New Business Objects Constants wizard runs.

5. Perform the following steps in the **Create Business Object Constant** dialog box:

- a. In the **Name** box, type the name you want to assign to the new constant.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- b. In the **Description** box, type an explanation of how the constant is to be used.

- c. Click the **Add** button to the right of the **Scope** box.

The Define Scope wizard runs.

- 1) Click the **Browse** button to the right of the **Business Object Scope** box and choose the business object to apply the constant to. Remember that business object constants are inherited by sub-business objects.

- 2) You can keep adding business objects by clicking the **Add** button to the right of the **Scope** box.

- 3) Click **Finish**.

- d. Click the arrow in the **Data Type** box to choose one of the following:

- **String**

Indicates that the value is a text string.

- **Boolean**

Allows two choices to the user (true or false).

- **List**

Contains a list of values.

- e. If you chose the **List** data type, a **Values** table appears. Click the **Add** button to the right of the **Values** table to add values to the list:

- 1) In the **Value** box, type a value for the list.

- 2) Select **Secured** to prevent the selected value from being overridden by another template.

- 3) Click **Finish**.

- f. In the **Default Value** box, enter the initial value of the constant. Entry differs depending on the data type you previously chose:

- If you selected the **String** data type, type in the default value.

- If you selected the **Boolean** data type, click the dropdown arrow to select **true** or **false** for the default value.
- If you selected the **List** data type, click **Browse** to select a value from the available ones on the list.

Note

If the selected default value is marked as **Secured** then this value cannot be overridden by any other template.

- g. Click **Finish**.

The new constant appears under the **Business Object Constants** folder in the **Extensions** view.

You can also see the business object constant by right-clicking the business object the constant is applied to and choosing **Open**. The constant appears in the **Business Object Constants** table on the **Main** tab.

6. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.

7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. To verify the constant on the server, run the **gettypeconstantvalue** utility. This utility tests the value of the business object constant on a particular business object in the database. The utility accepts the name of a business object constant and business object, and outputs the value of the constant if present.

Property constants

Property constants provide default behavior to properties on a business object. Because these constants are attached to properties, they are inherited to the same property on a sub-business object, unless the property constant is overridden at the child level.

Property constants:

- Are applicable to properties on business objects.
- Can be overridden.
- Support inheritance of the constant value on the property in the business object hierarchy.
- Can supply a default value.
- By default available on all properties on all business objects.

Here are some examples:

- Should the property be visible in the client?
- Should the property be required in the client?
- What should be the initial value of the property?
- What property names should be used to concatenate to this property?

Property constants reference

Property constants allow you to control whether a business object property is modifiable, visible, enabled, exportable, or required. In addition, you can set an initial value for a property and define complex properties comprised of a combination of properties and strings that assign a value to the property. Once configured, property constants apply to properties as displayed in both the Teamcenter rich client and thin client interfaces.

The following constants can be applied to business object properties:

- The **Complex property** constant allows you to specify a combination of properties and constant strings to assign a value to a selected property.
- The **Enabled** constant allows you to specify whether a property is enabled in the interface.
- The **Exportable** constant defines if a business object property can be exported using PLM XML.
- The **Initial Value** constant allows you to specify the initial value to be assigned to a property when the corresponding object is created.
- The **Modifiable** constant allows you to place restrictions on the modifiability of an object property.
- The **Required** constant allows you to specify whether a value must be entered for a specific object property.
- The **Visible** constant allows you to specify whether a specific object property is visible in the interface.

Property constants defined on parent business objects are inherited by sub-business objects. For example, a property constant defined on the **Item** business object is inherited by the **Document** business object and all sub-business objects of the **Document**.

You can override property constants defined on parent business objects. For example, if a property constant is configured on the **object_desc** property of the **Item** business object to make it modifiable only if null, a separate constant can also be configured on the **object_desc** property of the **Document** sub-business object to set the initial value of the **object_desc** property. In this case, the constant defined on the **Document** sub-business object takes precedence over the constant applied to the parent **Item** business object.

Note

Access Manager (AM) rules take precedence over property constants. For example, if you define a property constant stating that an object property is modifiable, but the user has not been granted write access to the object, the property remains read-only and the user cannot modify it.

Changing a property constant value

1. Select a property in a business object.
2. Select the property constant to change and choose **Modify**.
3. Change the value.
 - For *boolean* values, select or clear the **Value** box.
 - For *lists of values*, select a value from the list in the **Value** box.
 - For *strings*, type a value in the **Value** box.
4. Choose **Finish**.
5. Save and deploy your change.

Note

Once a property constant is changed, the **Overridden** property is selected and the **Template** property is set to the current template.

The screenshot shows the 'MyPart Master' form with the 'Properties' tab selected. It displays a table of properties and a section for 'Property Constants'.

| Property Name | Type | Storage Type | Inherited | Source | |
|-------------------|-----------|----------------|-------------------------------------|-----------------|-------------------------------------|
| reservation | Runtime | | <input checked="" type="checkbox"/> | WorkspaceObject | <input checked="" type="checkbox"/> |
| revision_limit | Attribute | Integer | <input checked="" type="checkbox"/> | WorkspaceObject | <input checked="" type="checkbox"/> |
| revision_number | Attribute | Integer | <input checked="" type="checkbox"/> | WorkspaceObject | <input checked="" type="checkbox"/> |
| safety_code | Runtime | | <input type="checkbox"/> | MyPart Master | <input type="checkbox"/> |
| supplier | Runtime | | <input type="checkbox"/> | MyPart Master | <input type="checkbox"/> |
| timestamp | Attribute | String | <input checked="" type="checkbox"/> | POM_object | <input checked="" type="checkbox"/> |
| user_can_unmanage | Runtime | | <input checked="" type="checkbox"/> | WorkspaceObject | <input checked="" type="checkbox"/> |
| wso_thread | Reference | TypedReference | <input checked="" type="checkbox"/> | WorkspaceObject | <input checked="" type="checkbox"/> |

Property Constants

| Name | Value | Overridden | COTS | Template |
|--------------|----------|-------------------------------------|-------------------------------------|------------|
| Enabled | false | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Exportable | Optional | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| InitialValue | | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Modifiable | Read | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Required | false | <input type="checkbox"/> | <input checked="" type="checkbox"/> | foundation |
| Visible | false | <input checked="" type="checkbox"/> | <input type="checkbox"/> | customer |

Create a property constant

Property constants provide default values to business object properties. Because these constants are attached to properties, they are inherited, just like the properties themselves.

You can define a property constant to have a specific scope so that it is available only on certain properties on certain business objects. This ensures that server API can retrieve the value properly on just those properties. You can also define a property constant to be available on all business objects or properties by using an asterisk (*) when you set the scope. In this way, you can set the scope to be as wide or narrow as you want.

You can create property constants for a number of situations: define the display name of the property, set whether the property is required, set the initial value of the property, and so on.

When you create a new constant, you must also add the code on the server to return the constant's value to the caller, so the caller can branch the business logic based on the returned value. The server side code can use the following published ITK to retrieve a property constant value:

```
int CONSTANTS_get_property_constant_value(
    const char*    constant_name,    /* <I> */
    const char*    type_name,        /* <I> */
    const char*    property_name,    /* <I> */
    char**         value              /* <OF> */
);
```

Once a constant is set, it can be overridden by another template. The rule is that the last template that gets installed determines the constant value that is used.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Extensions** view, select the project in which you want to create the new constant. Right-click the project and choose **Organize→Set active extension file**. Select the **business_objects.xml** file as the file in which to save the new constant.
3. Expand the project and the **Constants→Property Constants** folders.
4. Right-click the **Property Constants** folder and choose **New Property Constants**.

The New Property Constants wizard runs.

5. Perform the following steps in the **Create Property Constant** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new constant.

When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.

- b. In the **Description** box, type an explanation of how the constant is to be used.
- c. Click the **Add** button to the right of the **Scope** box.

The Define Scope wizard runs. Perform the following steps in the **Property Scope** dialog box:

- 1) Click the **Browse** button to the right of the **Business Object Scope** box and choose the business object to apply the constant to. Remember that property constants are inherited by sub-business objects.

If you want the constant to apply to all business objects, type an asterisk (*).

- 2) Click the **Browse** button to the right of the **Property Scope** box and choose the property to apply the constant to.

If you want the constant to apply to all properties, type an asterisk (*).

- 3) Click **Finish**.

The business object and property appear in the **Scope** box separated by a period (.).

An asterisk indicates the constant applies to all business objects or properties. For example:

- *.*

The constant can be applied to all properties on all business objects.

- *.object_desc

The constant can be applied to the **object_desc** property on all business objects

- Item.*

The constant can be applied to all properties on the **Item** business object.

- Item.object_des

The constant can be applied only to the **object_desc** property on the **Item** business object and its children.

- d. If desired, click the **Add** button to the right of the **Scope** box to narrow the scope further.
- e. Click the arrow in the **Data Type** box to choose one of the following:
 - **String**
Indicates that the value is a text string.
 - **Boolean**
Allows two choices to the user (true or false).
 - **List**
Contains a list of values.
- f. If you chose the **List** data type, a **Values** table appears. Click the **Add** button to add values to the list:
 - 1) In the **Value** box, type a value for the list.
 - 2) Select **Secured** to prevent the selected value from being overridden by another template.
 - 3) Click **Finish**.
- g. In the **Default Value** box, enter the initial value of the constant. Entry differs depending on the data type you previously chose:
 - If you selected the **String** data type, type in the default value.
 - If you selected the **Boolean** data type, click the dropdown arrow to select **true** or **false** for the default value.
 - If you selected the **List** data type, click **Browse** to select a value from the available ones on the list.

Note

If the selected default value is marked as **Secured** then this value cannot be overridden by any other template.

- h. Click **Finish**.

The new constant appears under the **Property Constants** folder in the **Extensions** view.

You can also see the property constant by right-clicking a business object the constant is applied to and choosing **Open**. Click the **Properties** tab. The constant appears in the **Property Constants** table.

6. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **business_objects.xml** file, the changes are saved in that file.
7. Deploy your changes to the test server. Right-click in the **Extensions** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
8. To verify the constant on the server, run the **getpropertyconstantvalue** utility. This utility helps test the value of a property constant on a particular business object and property in the database. The utility accepts the name of a property constant, business object, and property, and outputs the value of the constant if present.

Activity

1. Modify property constants.

In this activity, you modify property constants to make a property required and another property hidden.

2. Deploy and test updated constants.

In this activity, you deploy and test to make sure the **CCC_Category** property is now required and the **CCC_Cost** property does not display.

Review questions

1. Which constant provides consistent definitions that can be used throughout the system.

Select one answer.

- Business object
- Class
- Global
- Property

2. Teamcenter provides a property constant for setting a property's initial value.

- True
- False

Summary

Topics learned in this lesson:

1. Constants are configuration points within the data model.
2. Global constants provide consistent definitions that can be used throughout the system.
3. Business object constants provide default behavior to business objects.
4. Property constants provide default behavior to business object properties.
5. You can override an existing constant by setting a different value in the current template.
6. You must add the code on the server to return the constant's value to the caller.

Lesson

10 Rules

Purpose

The purpose of this lesson is to define and manage rules.

Objectives

After you complete this lesson, you should be able to:

- Describe rules.
- Describe and create business object display rules.
- Describe and create Generic Relationship Management (GRM) rules.
- Describe and create naming rules.
- Describe ID context rules.
- Describe and create compound property rules.
- Describe and create deep copy rules.
- Describe extension rules.

Help topics

Additional information for this lesson can be found in:

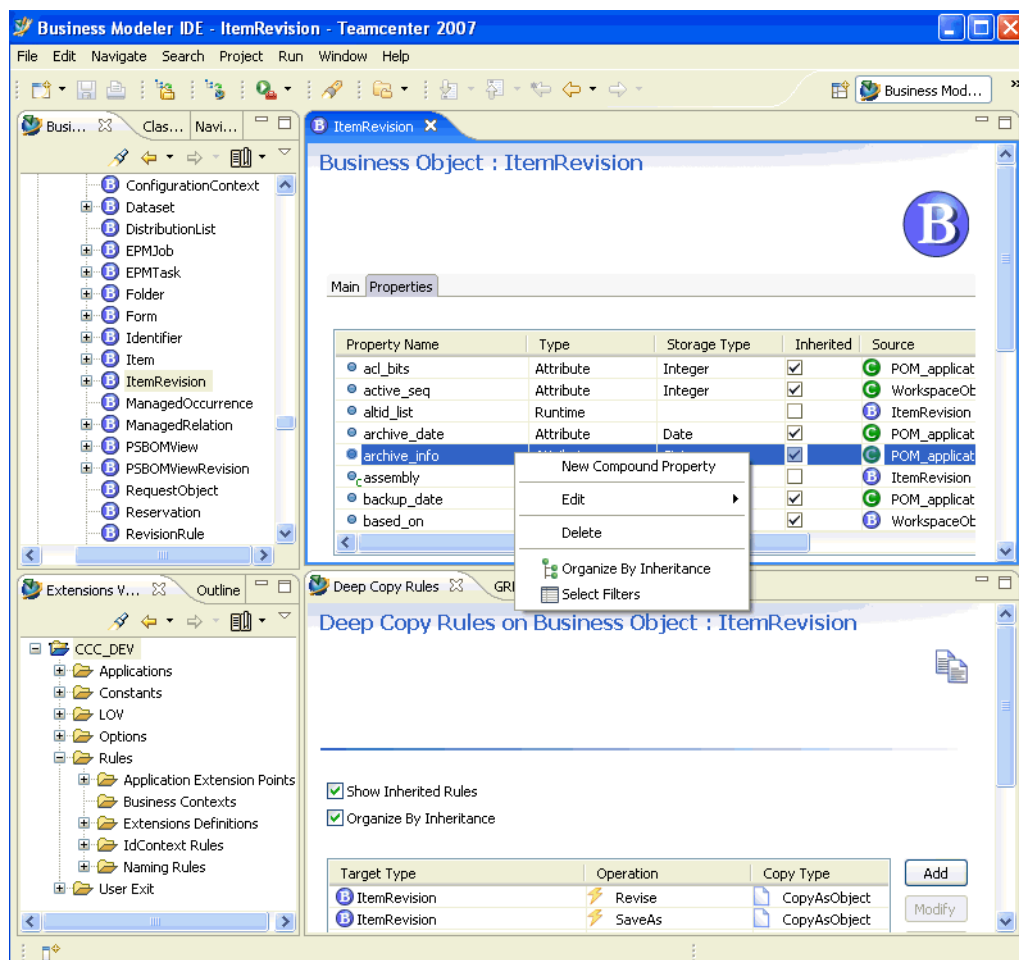
- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining rules

Rules govern the behaviors of business objects, including how they are named, what actions can be undertaken on them, and so on. Creating rules is also known as *business behavior modeling*.

An example of some rules are:

- Business object display rules
- Naming rules
- Compound property rules



Key points

- When a business rule is set on the parent business object and sub-business object, the business rule set on the sub-business object is executed.
- When a business rule is not set on a business object, the system searches up the hierarchy for a business rule set on any parent business objects.

- The first business rule found is executed.

Example

If a naming rule, deep copy rule, or property rule exists for the business object, it is used; otherwise the system checks each of the business object's parents until the rule is found or the top parent is reached.

What is a business object display rule

Business object display rules limit the objects that can be created by particular groups or roles. Simply stated, business object display rules limit the items a group of users or role sees on the **File**→**New** menu.

Business object display rules can be applied to the following business objects and their children:

- **Dataset**
- **Folder**
- **Form**
- **Item**

Create a business object display rule

Business object display rules limit the kinds of objects that can be created by particular groups or roles.

The **Display Rules** view displays the groups that can view a business object in the create menus in the Teamcenter user interface. In other words, these are the individuals that are allowed to create the selected kind of business object.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Business Objects** tab next to the **Navigator** view.
3. In the **Business Objects** view, select the project in which you want to create a new display rule.

Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the data model changes.

4. Select a business object for which you want to create a new display rule. To search for a business object, you can click the **Find Business Object** button at the top of the view.

Display rules can be applied to the following business objects and their children: **Alias**, **Dataset**, **Folder**, **Form**, **Item**, **MEActivity**, **MEOP**, **MEProgress**, and **MEWorkarea**.

5. Click the **Display Rules** tab in the lower right of the perspective.
6. Click the **Load Organization** button in the **Display Rules** view. The Teamcenter Repository Connection wizard prompts you to log on to a server to look up the available groups and roles in the organization.
7. The groups and roles from the server appear in the view. Select the groups and roles that have rights to create the business object. Unselected groups and roles have the business object hidden from them in the create menus in Teamcenter rich client applications.

Select the **Propagate to sub Business Objects** check box if all children of the business object inherit the display rule.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

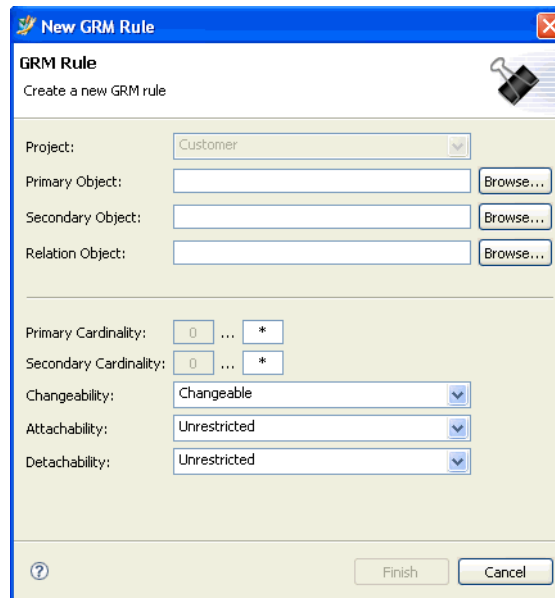
9. After you create a rule, you can deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

10. After deployment, test your new display rule in the Teamcenter rich client.

For example, if you created a display rule on the **Item** business object, log on to the My Teamcenter application as a user in a group that has rights to create items and choose **File**→**New**→**Item**. You should be able to create new **Item** business objects. Then log on as a user in a group that does not have rights to create **Item** business objects. You should not be able to create **Item** business objects.

What is a GRM rule

A *Generic Relationship Management (GRM) rule* applies constraints on the relationship between two business objects. When you create a GRM rule, you select the primary and secondary business objects for the relationship, the relationship they have to one another, and the constraints to be applied. Simply stated, the GRM rule constrains copy and paste capabilities between objects.



Key points

When you create the GRM rule, the following constraints must be defined:

- **Primary Cardinality**

Determines the number of allowed occurrences of the primary object in relation to the secondary object.

- **Secondary Cardinality**

Determines the number of allowed occurrences of the secondary object in relation to the primary object.

- **Changeability**

Specifies whether the relationship links between objects can be added, deleted, or otherwise changed.

- **Attachability**

Specifies whether new relationship links can be made between objects.

- **Detachability**

Specifies whether the relationship links that exist between objects can be removed.

Note

The GRM rule applies for all children of the primary and secondary objects.

Example

If the **ItemRevision** business object is chosen as the primary object, and the **DirectModel** dataset is chosen as the secondary object, all children of these objects that have the relationship inherit the rule. Therefore, all instances of the children of **ItemRevision** that are related to all instances of children of **DirectModel**, and use the relation specified by the rule, are subject to the constraints defined by the rule. However, rules defined for a specific sub-business object take precedence over the relation rules defined for a parent object.

Add a GRM rule

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Business Objects** tab next to the **Navigator** view.
3. In the **Business Objects** view, select the project in which you want to create the new rule.

Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the data model changes.

4. Click the **GRM Rules** tab in the lower right portion of the perspective. The **GRM Rules** view displays the active GRM rules in the project.

To search for existing GRM rules by primary object, secondary object, or relation, type strings in the **Primary**, **Secondary**, or **Relation** boxes, or click the **Browse** buttons. The GRM rules table displays the results of the search.

Use the **Add**, **Modify**, or **Remove** buttons to work with the GRM rules.

5. To create a GRM rule between two objects, click the **Add** button.

The GRM Rule wizard runs.

6. Perform the following steps in the **GRM Rule** dialog box:

- a. Click the **Browse** button to the right of the **Primary Object** box to select the primary business object in the relationship.
- b. Click the **Browse** button to the right of the **Secondary Object** box to select the secondary business object in the relationship.
- c. In the **Relation Object** box, enter the relationship between the primary and secondary business objects.

To see all the available relationships, click the **Browse** button and type an asterisk (*) in the **Find** dialog box. These relations are children of the **IMANRelation** business object.

- d. In the **Primary Cardinality** box, type the number of primary objects that can be related to a secondary object with the relationship. An asterisk (*) means an unlimited number.
- e. In the **Secondary Cardinality** box, type the number of secondary objects that can be related to a primary object with the relationship. An asterisk (*) means an unlimited number.

- f. In the **Changeability** box, select **Changeable** if the relationships using this rule can be deleted or otherwise changed, **Add Only** if only new relationships can be made between the objects, or **Frozen** if relationships cannot be changed in any way.
- g. In the **Attachability** box, select **Unrestricted** if all users can relate new objects using the rule, or select **Write Access Req.** if Access Manager rules should be used to evaluate if the relationship creation is allowed.
- h. In the **Detachability** box, select **Unrestricted** if all users can remove the relationships between objects using the rule, or select **Write Access Req.** if Access Manager rules should be used to evaluate if the relationship creation is allowed.
- i. Click **Finish**.

The rule is created and appears in the table on the **GRM Rules** view.

7. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

8. Deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
9. After deployment, verify the new relationship rule in the Teamcenter rich client.

You can use the following example:

- a. Create a GRM rule with the following characteristics:

Primary object: **Item**
Secondary object: **UGPART**
Relation object: **IMAN_manifestation**
Primary cardinality: **0 ... 1**
Secondary cardinality: **0 ... 1**
Changeability: **Changeable**
Attachability: **Unrestricted**
Detachability: **Unrestricted**

- b. In the My Teamcenter application, create an **Item** and put a **UGPART** dataset on it with the **IMAN_manifestation** relationship.
- c. Try to create another **UGPART** dataset on the same item. An error message states that you cannot do this because it violates GRM rule constraints that only allow one **UGPART** object to be related.

Activity

1. Modify display rules for a business object.

In this activity, you create a business object display rule so twhite cannot create **CCC_Car** items.

2. Create a GRM rule.

In this activity, you create a General Relationship Management (GRM) rule. This rule has **CCC_Item Revision** as the primary object and **CCC_Change** as the secondary object. It prevents users from pasting a **CCC_Change** into a **CCC_Item Revision** using the default relationship.

Review questions

1. Display rules automatically propagate to the business object's children.
 - True
 - False
2. Determines the number of allowed occurrences of the primary object in relation to the secondary object.

Select one.

- Changeability
- Attachability
- Detachability
- Primary Cardinality

What is a naming rule

A *naming rule* defines how objects are named, including how IDs are automatically assigned when objects are created. Naming rules can be used to name items, item revisions, datasets, forms, projects, and work contexts.

Example

For example, if you want all your documents to be named **Document_** followed by an incremental number, you create a naming rule to automatically assign that name when documents are created.

A naming rule consists of rule pattern and a counter. Before creating a naming rule, you should be familiar with the pattern and counter formats. Only one counter is permitted per naming rule.

Create and attach a naming rule

1. Set the **active extension file** to **rules.xml**.
2. Right-click the **Naming Rules** folder in the **Extensions View** and choose **New Naming Rules**.
3. Enter the following information in the **Naming Rule** dialog box:
 - a. In the **Name** box, type the name of the naming rule.
 - b. In the **Pattern** box, type the naming pattern for the naming rule.
 - c. Select the **Generate Counters** check box if you want to add a counter to the naming rule. **Generate Counters** applies to the first pattern only.
 - d. If **Generate Counters** is selected, type the initial value and the maximum value for the counters.
4. Save the new naming rule.
5. Find the business object and property to set the naming rule.
6. Right-click the property and choose **Edit**→**Naming Rule**→**Attach Naming Rule**.
7. Select the naming rule and case.

Naming rule patterns

| Character | Pattern match |
|-----------|---|
| & | Alphanumeric value |
| X or x | Uppercase or lowercase alphanumeric value |
| N or n | Numeric value |
| @ | Alphabetic value |
| A or a | Uppercase or lowercase alphabetic value |
| “string” | String delimiter |
| ? | Any single character value |

Note

This is only a sampling of possible naming rule patterns. See the documentation for a complete list.

Add a naming rule

A naming rule defines how objects are named, including how IDs are automatically assigned when objects are created. For example, if you want all your documents to be named **Document_** followed by an incremental number, you create a naming rule to automatically assign that name when documents are created. Naming rules can be used to name items, item revisions, datasets, forms, projects, and work contexts.

A naming rule consists of rule patterns and a counter. Before creating a naming rule, you should be familiar with the pattern and counter formats.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Extensions View** tab.
3. In the **Extensions** view, select the project in which you want to create the rule. Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the new rule.
4. Expand the project and the **Rules→Naming Rules** folders.
5. Right-click the **Naming Rules** folder and choose **New Naming Rules**.

The New Naming Rules wizard runs.

6. Enter the following information in the **Naming Rule** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new naming rule.

The name cannot contain spaces. When you name a new data model object, add a prefix to the name to designate the object as belonging to your organization, such as a three-letter acronym.
 - b. Click the **Add** button to add a naming rule pattern in the **Patterns** pane. The Add Naming Rule Pattern wizard runs:
 - 1) In the **Pattern** dialog box, type a naming pattern in the **Pattern** box.

Note

A naming rule consists of rule patterns and a counter. Before creating a naming rule, you should be familiar with the pattern and counter formats. For information about creating a pattern, click the **?** button at the bottom of the dialog box and see the help topics on naming rule patterns.

You can also click the **Insert LOV** or **Insert Rule** buttons to enter existing patterns. When the **Find** dialog box appears, you can type an asterisk * in the search box to see all existing LOVs or rules.

- 2) When you are done entering a pattern, click **Finish**.

The pattern appears in the **Pattern** pane of the **Naming Rule** dialog box.

Note

By default, the first pattern in the list is used to assign IDs for objects.

- c. The **Generate Counters** check box appears after a pattern is entered in the **Patterns** pane. Select the **Generate Counters** check box if you want to add a counter to the naming rule. Only the first pattern in a rule can have a counter assigned to it, and the counter's initial and maximum values must follow the format defined in the first pattern.

If you select the **Generate Counters** check box, type the **Initial Value** and the **Maximum Value** for the counters. For example, the counter could start at **0** and go to **999**.

- d. Change the patterns as desired by using the **Add**, **Modify**, **Remove**, **Copy**, **Move Up** and **Move Down** buttons.
- e. Click **Finish**.

The naming rule is added to the **Naming Rules** folder.

7. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

To put the naming rule into effect, you must attach it to a property on a business object, such as an item name.

Attach a naming rule to a business object property

A naming rule defines how objects are named. Naming rules can be used to name items, item revisions, datasets, forms, projects, and work contexts.

To put a naming rule into effect, you must attach it to a property of a business object. For example, if you want to use a naming rule to define how all items are named, attach the naming rule to the **item_id** property of the **Item** business object.

You can only attach naming rules to form string properties.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Business Objects** tab next to the **Navigator** view.
3. In the **Business Objects** view, select the project in which you want to make the change. Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the change.
4. Browse to the business object to which you want to attach the naming rule.

To search for a business object, you can click the **Find Business Object** button at the top of the view.

For example, to attach a naming rule to be applied to all **Item** business objects, select the **Item** business object.

5. Right-click the business object, choose **Open**, and click the **Properties** tab in the resulting view.

The business object properties appear in a table.

In the properties table, look for the property to which you want to attach the naming rule. For example, if you want to attach a naming rule to the **Item** business object, you could choose the **item_id** property.

6. Right-click the property and choose **Edit→Naming Rule→Attach Naming Rule**.

The Attach Naming Rule wizard runs.

7. Perform the following steps in the Attach Naming Rule wizard:
 - a. Select the naming rule you want to attach.
 - b. Click the arrow in the **Case** box to select the kind of user input allowed for the naming rule, **Mix** for mixed case, **Upper** for uppercase, or **Lower** for lowercase.
 - c. Click **OK**.

The naming rule is attached to the property and appears in the **Naming Rule** column of the properties table.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

9. Deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

10. After deployment, test your newly attached naming rule in the Teamcenter rich client by creating an instance of the business object.

For example, if you attached a naming rule to the **item_id** property of the **Item** business object, in the My Teamcenter application, choose **File→New→Item** and choose **Item** as the type. When you click the **Assign** button, your new naming rule pattern appears in the **ItemID / Rev - Name** boxes.

Naming rule examples

Using literal variables in patterns

To illustrate the use of literal variables, assume that the naming convention for a particular type of dataset requires a 3-character suffix, either **ENG** or **MFG**, followed by a 4-digit number ranging from **0000** to **9999**.

To establish this pattern, a list of values (LOV) named **Context** is created containing the values **ENG** and **MFG**. A naming rule is then created using the LOV and numeric characters. The pattern would appear as follows:

```
{LOV:Context}nnnn
```

The naming rule is then attached to the **name** property of the dataset business object.

Note

Literal variables cannot be used in patterns used to autogenerate counters.

Using counters with naming rules

Only one pattern in a naming rule, the first pattern, is used with counters. Other patterns in the rule are used for validation but not for automatically generating the ID.

Any number of counters can be used in a pattern, for example **nnn"."a**. With counters activated for this pattern, the **Assign** button allocates IDs as follows:

```
000.a  
000.b  
000.c  
⋮  
000.z
```

The right-side counter range completes first and then moves to the next range from the right, as follows:

```
001.a  
⋮  
001.z  
  
002.a  
⋮  
002.z  
  
003.a  
⋮  
003.z
```

What is an ID context rule

You can assign IDs to items depending on their context. You can write rules for these context ID situations.

- **Alias ID rule**

Alias identifiers store part numbers and other attribute information for similar parts. Alias IDs can be associated with many items or item revisions. You can write a rule to define the context when an alias ID can be applied to an item.

- **Alternate ID rule**

Alternate identifiers store information (such as part numbers and attributes) about the same part from different perspectives. They allow different types of users to display an item according to their own rules rather than according to the rules of the user who created the object.

Key points

- ID context rules determine behavior when creating alternate and alias identifiers by combining valid combinations of identifier business object, ID context, and identifiable business object.
- Cardinality rules for alternate IDs are also created as part of the ID context rule.

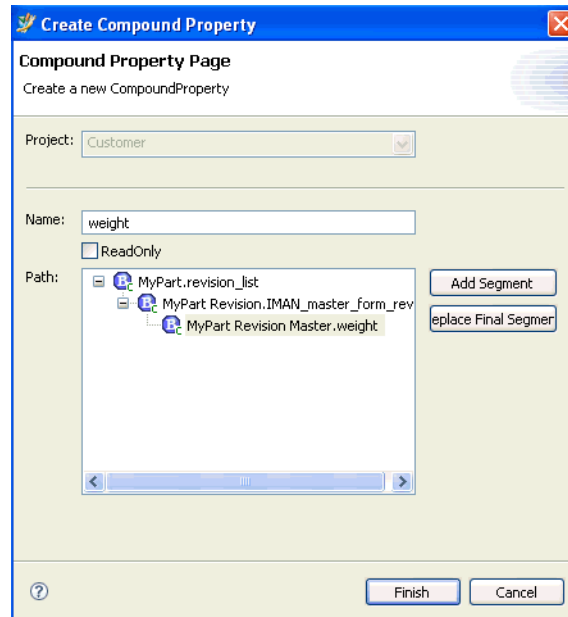
What is a compound property rule

A *compound property* is a property on a business object that can be displayed as a property of an object (the display object) although it is defined and resides on a different object (the source object).

A compound property uses **Relation** and **Reference** properties to traverse from the source to the destination object.

A compound property creates the path that the property follows to display the source object's property on the display object, if the path exists for the two objects.

Property rules allow you to control access to and the behavior of object properties. You can navigate up and down the business object tree to define the compound property.



Key points

- Once configured, property rules apply to properties as displayed in both the rich client and thin client interfaces.
- Property rules defined on base business objects are inherited by sub-business objects. For example, a property rule defined on the **Item** business objects is inherited by the **Document** business objects.

Defining a compound property rule path

This example shows two compound properties to be defined on **MyPart**. The table shows how to traverse the relation and child objects to find the target property. Knowledge of the data model is important to find the target property.

| Compound property | Object | Traverse | Child object | Target property |
|-------------------|-----------------|----------------------|------------------------|-----------------|
| safety_code | MyPart | IMAN_master_form | MyPart Master | safety_code |
| weight | MyPart | revision_list | MyPart Revision | |
| | MyPart Revision | IMAN_master_form_rev | MyPart Revision Master | weight |

Item structure with properties

MyPart

safety_code (compound property)

weight (compound property)

MyPart Master

safety_code

supplier

MyPart Revision

MyPart Revision Master

weight

material_code

material_description

Add a compound property rule

A *compound property* is a property on a business object that can be displayed as a property of an object (the display object) although it is defined and resides on a different object (the source object). A compound property uses **Relation** and **Reference** properties to traverse from the source to the destination object. A compound property creates the path that the property follows to display the source object's property on the display object, if the path exists for the two objects. For example, you can use a compound property rule to display a custom property from a form on a business object.

Perform the following steps to add a compound property rule to a business object:

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Business Objects** tab next to the **Navigator** view.

In the **Business Objects** view, select the project in which you want to make the change. Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the change.

3. Browse to the business object to which you want to add the compound property rule. To search for a business object, you can click the **Find Business Object** button at the top of the view.

For example, to add a compound property rule to the **Item** business object, select the **Item** business object.

4. Right-click the business object, choose **Open**, and click the **Properties** tab in the resulting view.

The business object properties appear in a table.

5. Right-click in the **Properties** table and choose **New Compound Property**.

The Create Compound Property wizard runs.

Note

To edit an existing compound property, right-click on the compound property in the **Properties** table and choose **Edit→Edit Compound Property**. If the compound property is not editable, then the wizard displays the compound property details in read-only mode.

6. Perform the following steps in the **Compound Property Page** dialog box:
 - a. In the **Name** box, type the name you want to assign to the new compound property.

The name should match the format of other properties (that is, all lowercase with underscores separating words).

The **Path** pane displays the target business object to which you add the new compound property rule.

- b. Select the **ReadOnly** check box if you don't want users to be able to change the compound property value on instances of the business object in Teamcenter.
 - c. Click the **Add Segment** button. The Add Compound Property Segment wizard runs, displaying the message **Choose a property**.
7. Perform the following steps in the **Compound Property Segment** dialog box:
- a. Select the property you want to use for the first segment of the compound property.
 - b. Click **Next**. The **Compound Property Segment** dialog box displays the message **Choose a business object**.
 - c. Select the business object you want to use for the next segment of the compound property.
 - d. Click **Finish**.

The **Compound Property Page** dialog box appears, showing the compound property path thus far.

8. To add more segments to the compound property, click the last segment in the **Path** pane, click **Add Segment**, and repeat step 7.

To replace a segment, select the segment and click **Replace Segment**.

9. When you are ready to add the last segment, click the **Add Final Segment** button. The Add Compound Property Segment wizard runs, displaying the message **Choose a property**.

Perform the following steps in the **Compound Property Segment** dialog box:

- a. Select the property you want to use for the final segment of the compound property.
- b. Click **Finish**.

The **Compound Property Page** dialog box appears, showing the compound property path.

10. When you are done adding segments in the **Compound Property Page** dialog box, click **Finish**.

The new compound property appears in the property table.

11. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

12. Deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.

13. After deployment, verify your new compound property rule on the business object by looking at the business object's properties page in the Teamcenter rich client. You should see the same property on both the source and the target objects.

Activity

1. Create a naming rule.

In this activity, you create naming rules for the **CCC_Item** ID and name properties.

2. Create compound property rule.

In this activity, you create a compound property rule for **CCC_Item** to display the **CCC_CarType** property from the **CCC_Item Master** form.

Review questions

1. Which characters are the pattern match for uppercase or lowercase alphanumeric value?

Select one.

- @
- &
- A or a
- N or n
- X or x

2. Compound property is the concatenation of two properties into one?

- True
- False

What is a deep copy rule

A *deep copy rule* defines whether objects belonging to an item revision can be:

- **Copy as objects**

Creates a new object of the same type as the related object and relates to the new revision.

- **Copy as reference**

Creates a new relation between the new revision and the related object.

- **No copy**

Do nothing.

This is applicable when a user performs a **save as** or **revise** operation on an item revision.

Note

Deep copy rules are only applied to item revision business objects.

Key points

- Deep copy rules should be tested to be sure it performs as expected.
- Existing deep copy rules should be tested to validate that the behavior meets company procedures.
- Deep copy rules defined for a parent business object are automatically inherited by all sub-business objects of the parent business object. Conversely, deep copy rules removed from a parent business object are automatically removed from all sub-business objects.

What is the deep copy hierarchy

Revision structure before the revise:

 **MyPart Revision (1)**

IMAN_specification (relation)

 **MyPartSpec001**

After revise with copy as objects applied:

 **MyPart Revision (2)**

IMAN_specification (relation)

 **MyPartSpec002**

After revise with copy as reference applied:

 **MyPart Revision (2)**

IMAN_specification (relation)

 **MyPartSpec001**

After revise with no copy applied:

 **MyPart Revision (2)**

Note

Objects attached to the item revision by the **IMAN_reference** relation can only be subject to **CopyAsReference** or **NoCopy** rules. Reference attachments cannot be copied forward as new objects.

Add a deep copy rule

Deep copy rules define whether objects belonging to an item revision can be copied as objects, copied as reference, or not copied when a user performs a save as or revise operation on an item revision. Deep copy rules are only applied to item revision business objects.

Perform the following steps to create a deep copy rule on an item revision business object:

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.

2. Click the **Business Objects** tab next to the **Navigator** view.

3. In the **Business Objects** view, select the project in which you want to create the new rule.

Right-click the project and choose **Organize→Set active extension file**. Select the **rules.xml** file as the file in which to save the data model changes.

4. In the **Business Objects** view, browse to the **ItemRevision** business object or a child of the **ItemRevision** business object. (Deep copy rules are only allowed for item revisions or children of item revisions.) To search for a business object, you can click the **Find Business Object** button at the top of the view.

5. Select the item revision business object and click the **Deep Copy Rules** tab. The **Deep Copy Rules** view displays the active rules on the business object.

Select the **Show Inherited Rules** check box to display all rules inherited from parent business objects.

Select the **Organize by Inheritance** check box to sort the rules by parent business object names.

Use the **Add**, **Modify**, or **Remove** buttons to work with the deep copy rules.

6. Click the **Add** button.

The Add Deep Copy Rule wizard runs.

7. Perform the following steps in the **Add Deep Copy Rules** dialog box:

- a. In the **Operation Type** box, select **Save As** or **Revise** to indicate when the copy operation is to be used.
- b. In the **Copy Type** box, choose the kind of copying to be allowed for the business object:

- **CopyAsObject**

Creates a new object of the same type and relation as the source item revision. Objects created by this method are totally independent of the source object. Therefore, modifications to the new object are not reflected in the source object.

- **CopyAsReference**

Copies forward the related object but maintains a reference to the source object. Therefore, modifications performed on the copied object are propagated to the source object.

- **NoCopy**

Specifies that objects of the selected type and relation are not copied forward to the new object during the save as or revise operation.

- c. Click the **Browse** button to the right of the **Relation Type** box to choose the relationship type the object has with the item revision to be copied. Objects with the relationship that matches with the chosen relationship are only copied from the source item revision to the destination item revision.

To see all the available relationships, click the **Browse** button and type an asterisk * in the **Find** dialog box. Available relationships are children of the **IMANRelation** business object.

- d. Click the **Browse** button to the right of the **Object Type** box if you want to specify the business object type to be copied. To see all available business objects, type an asterisk * in the **Find** dialog box.

Choose the value as **MatchAll** if you want all objects to be copied forward no matter what type of business object they are.

- e. Select **Required** if you want to prevent users of the Teamcenter rich client from overriding the rule at run time.
- f. Click **Finish**.

The rule is created and appears in the table on the **Deep Copy Rules** view.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.

If you set the active extension file as the **rules.xml** file, the changes are saved in that file.

9. Deploy your changes to the test server. Right-click in the **Business Objects** view and choose **Deploy Template**, or click the **Deploy Template** button on the main toolbar.
10. To verify the deep copy rule, open the My Teamcenter application in the Teamcenter rich client, select an item revision of the type for which you created the rule, and choose **File**→**Save As** or **File**→**Revise**. Verify that the behavior works as expected.

What are extension rules

Extension rules allow you to write a custom function or method for Teamcenter in C or C++ and attach the rules to predefined hook points in Teamcenter. Extension rule capabilities fall into two main areas.

- Create new extension rules and assign them to a business object or a business object property.
- Use predefined extension rules to perform actions.

Extension points include:

- **Pre-Condition**

Allows you to place limits before an action. For example, limit individual users by their work context to create only a certain item type.

- **Pre-Action**

Allows you to execute some code before an action. For example, add user information to the session prior to translation.

- **Base-Action**

Allows you to execute code for an action.

- **Post-Action**

Allows you to execute some code after an action. For example, automatically start an item in a workflow.

Note

Application extensions can be used to configure business logic on the server, a Teamcenter rich client application (such as My Teamcenter), a Teamcenter thin client application, or any application.

Predefined extension definitions

You can use predefined (internal) extension rules to perform actions.

For example, you can use the predefined **createObjects** extension definition to automatically create a related Microsoft Word dataset whenever an item is created.

Following are the predefined extensions definitions that can be viewed under the **Rules→Extensions Definitions** folder:

- **Check_Validation_Results**

Checks validation results with validation rules. The baseline IR is created only when the original IR passes validation rule verification. This handler is registered as pre-condition. The **baseline_precondition_validation_rule_item_revision** preference allows you to set up the validation rule set item and item revision.

- **Copy_Validation_Results**

Copies validation results from the original IR to a new IR and creates a validation master form under the new IR. This handler is registered as a post action when the IR is revised, or saved as, or baselined. This handler is a fix for those cases when an IR with validation results is copied into a new IR, and all existing validation results are lost or incorrectly created.

- **autoAssignToProject**

Automatically assigns the selected workspace object to the user's current project, as defined by the work context or user settings.

- **checkLatestReleased**

Verifies whether the latest revision of an item is in released status prior to creating a new revision.

- **convertFile**

Converts a file to the proper format after a dataset file import.

- **copyVariantExpr**

Copies forward variant information, such as options, option defaults, and rule checks from the source item revision to the target item revision. It does not copy forward the variant information, such as variant conditions at the occurrence level.

- **createCAEObjects**

Creates a CAE object after creating an item revision.

- **createObjects**

Creates objects after creating a new item or item revision.

When you assign the **createObjects** extension to an item or item revision business object, you define the arguments that specify the business object of dataset, form, or folder object that is created, and the relationship with which that object is attached to the item or item revision.

- **setIdIdentifierProperties**

Sets the selected attribute values to null when deep copying an alternate identifier using the **Revise** or **Save As** operations. This results in the selected attribute values not being carried forward.

- **validateFile**

Validates proper file format after a dataset file import.

Use a predefined extension definition

1. In the **Extensions** view, expand the project and the **Rules→Extensions Definitions** folders.

2. Double-click the predefined extension definition you want to use, for example, **createObjects**.

The details of the extension appear in a new **Extension Definition** view.

In the **Availability** table, view the business object, operation, and extension point for which the extension can be used. Decide which business object and operation for which you want to use the extension, and observe the extension point where it can be used.

3. In the **Business Objects** view, right-click the business object on which you want to use the extension, and choose **Open Extension Rules**.

The operations and properties for the business object appear in a new **Extension Rules** view.

4. In the **Extensions Rules** view for the business object, choose the operation and click the **Add** button next to the extension point (**Pre-Condition**, **Pre-Action**, **Base-Action**, or **Post-Action**).

Note

This must be an operation and an extension point that appears in the **Availability** table for the extension as described in step 2.

The Add Extension Rule wizard runs.

5. In the **Extension** dialog box, click the **Browse** button to the right of the **Extension Rule** box and choose the extension.
6. If arguments are required on the operation, the **Add** button to the right of the **Arguments** box is enabled. Click the **Add** button.

The New Argument wizard runs.

- a. Click **Browse** to the right of the **objectType** box to choose the business object.
 - b. Click **Browse** to the right of the **relationsType** box to choose the relationship.
 - c. Click **Finish** after you finish selecting arguments.
7. Click **Finish**.

The extension is added to the table under the extension point.

8. Save the data model by choosing **File**→**Save Data Model**, and deploy the data model by right-clicking a view and choosing **Deploy Template**.

Verify that the extension rules work as expected in the Teamcenter rich client.

For example, if you use the predefined **createObjects** extension definition to automatically create a dataset whenever a certain item type is created, verify that the dataset is created in the My Teamcenter application.

Activity

- Create a deep copy rule.

In this activity, you create a deep copy rule for the **CCC_Item Revision** to create new specifications and reference existing reference objects.

Review questions

1. What is the expected output for the deep copy rule to copy specification as object for this item?










MyPart Revision (1)

IMAN_specification (relation)



MyPartSpec001

Select one.

-  **MyPart Revision (2)**
IMAN_references (relation)
 **MyPartSpec002**
-  **MyPart Revision (2)**
IMAN_specification (relation)
 **MyPartSpec001**
-  **MyPart Revision (2)**
IMAN_specification (relation)
 **MyPartSpec002**
-  **MyPart Revision (2)**
IMAN_specification (relation)

2. Extension points include?

Select all that apply.

- **Post-Action**
- **Post-Condition**
- **Pre-Action**
- **Pre-Condition**

Summary

Topics learned in this lesson:

1. Rules govern the behaviors of business objects.
2. Business object display rules limit the objects that can be created by particular groups or roles.
3. The GRM rule constrains copy and paste capabilities between objects.
4. Naming rules define how objects are named and consists of rule patterns and a counter.
5. ID context rules assign IDs to alias and alternate items depending on their context.
6. Compound property rules allow a property on a business object to be displayed as a property of an object.
7. Deep copy rules define whether objects belonging to an item revision are copied or referenced when a revision is saved or revised.
8. Extension rules allow you to write custom code and attach the rules to predefined hook points in Teamcenter.

Lesson

11 Data model files

Purpose

The purpose of this lesson is to describe data model files and template projects and show how they are used to manage the data model.

Objectives

After you complete this lesson, you should be able to:

- Package extensions into a solution template.
- Add extensions to a template.
- Install custom solution templates.
- Import a template project.
- Import a model file.

Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Defining data model files

The *data model* is the structure in Teamcenter into which items are placed, for example, business objects, classes, and attributes.

You can package extensions to the data model as a solution template and distribute the template for installation to a production environment. Templates are installed using Teamcenter Environment Manager.

You can import data model from projects or model files into the Business Modeler IDE.

Note

The template project directory and all of its contents should be managed in a source control management system (SCM). This is recommended because the template project is treated as source code. An SCM is the best way to manage source code, especially when two or more users are making concurrent changes. An SCM provides robust tools that integrate with the Business Modeler IDE for handling file merges, graphical file differences, and syncing to the latest code checked in from other users.

Package extensions into a solution template

1. Choose **File**→**New**→**Other**, and in the **New** dialog box, choose **Business Modeler IDE**→**Package Template Extensions**.

Click **Next**.

The Package Template Extensions wizard runs.

2. Perform the following steps in the **Package Template Extensions** dialog box:
 - a. In the **Project** box, select the project whose extensions you want to package into a template.
 - b. Leave the **Use default location** check box selected if you want the template files to be placed in your workspace in the **output\packaging** folder under your project.

If you want to set the folder where the template files are stored, clear the **Use default location** box, and click the **Browse** button to the right of the **Target folder** box to choose the folder where the template files are to be saved.

- c. Click **Finish**.

The template files are saved in the target folder.

By default, the template files are saved to the **output\packaging** folder under your project. To see the files in the **packaging** folder, right-click in the **Navigator** view and choose **Refresh**.

Note

You can also locate the files on your system. For example, on a Windows system, they are saved by default to:

```
C:\Documents and Settings\username\TeamCenter\BMIDE\
project\output\packaging
```

The following template files are placed in the *project/output/packaging* directory:

- *template-name_install.zip*

This ZIP file contains all the support files for installing and upgrading your template, and any data files that were stored in the *project/install* folder.

- *template-name_template.zip*

This ZIP file contains the template definitions (*template-name_template.xml*), the dependency file

(*template-name_***dependency.xml**), and the optional baseline file (*template-name_***tcbaseline.xml**).

- **feature_***template-name.xml*

This file contains the information necessary for TEM to recognize the template and how to handle the template for installation and upgrade.

- *template-name***Bundle_***language-code_country-code.xml*

This file contains the localized text for the feature file so that TEM can display the feature description in the localized version.

Adding extensions to the template

After a template project is created, you can extend the data model and business rules by creating new definitions using the Business Modeler IDE. Your data model extensions are saved in the template project.

One of the benefits of using the Business Modeler IDE is the strict validation that it performs on the extensions in your template.

- Each time the Business Modeler IDE loads your template, it first loads all extensions from the dependent templates.
- After loading the dependent extensions, each of your template extension definitions is loaded and validated against the existing model. If any validation errors occur, they are displayed in the **Console** view in the Business Modeler IDE.
- You can also use the **Reload Data Model** menu command (available in most views within the Business Modeler IDE) to load and validate your model.

The **Reload Data Model** validation tool becomes very important to you whenever you add a template dependency, manually edit a source file during a SCM merge from another user, add a new version of the dependent templates, or upgrade to the next version.

Installing a custom solution template

After you package extensions, the system administrator installs the resulting solution template to a production environment using Teamcenter Environment Manager.

1. Copy the packaged extension files from the **packaging** directory on your Business Modeler IDE client to the **Templates** directory on the server.
2. From the TEM, **Perform maintenance on an existing configuration** and **Rebuild/Update the database**.
3. Use the **Database Rebuilder** to find and load the new template.
4. Verify the installation.

Import a Business Modeler IDE template project

A project stores all extensions made against the data model in XML files. You can import a project from another Business Modeler IDE installation.

Note

Two or more Business Modeler IDE installations cannot point to the same project unless they are both using a source control management (SCM) system. Import the project only after you have created a view in the SCM that contains the project to be imported.

1. Map a drive to the project directory on another computer, or copy the project directory to your own computer.
2. Choose **File→Import**.
The Import wizard runs.
3. In the **Select** dialog box, choose **Business Modeler IDE→Import a Business Modeler IDE Template Project**. Click **Next**.
4. Perform the following steps in the **Import Business Modeler IDE Template Project** dialog box:

- a. Click the **Browse** button to the right of the **Project contents** box to choose the folder that contains the project. This can be a directory on a mapped drive, or a directory you have already copied to your computer.
- b. Click the **Browse** button to the right of the **Teamcenter model folder** box to choose the model directory where the dependent data model XML files are stored. The model directory is created when you install the Business Modeler IDE.
- c. Click the arrow in the **Select active file** box to choose the extension file to make active after the project is imported, for example, **business_objects.xml**.

During normal extension work, you set the active file to hold extensions.

- d. Click **Finish**.

The wizard imports the project and displays it in views.

5. Create extensions using the new project just as you would a project you created yourself.

Note

The project files remain in the original location. A link is created from your workspace to the project folder. Any new extension files created against this project are added to the original location.

Import a model file

You can import the contents of a data model file into a Business Modeler IDE project. The data model elements in the import file are written to an extension file.

This is useful when you have an extension file from one project template that you want to share with another project template. For example, you may want to import a **business_objects.xml** from another project.

Note

If you import data extracted with the **bmide_postupgradetotc** utility, you must first create a project with the same solution name and solution display name as used in the utility.

1. Choose **File→Import**.
The Import wizard runs.
2. In the **Select** dialog box, choose **Business Modeler IDE→Import model file**. Click **Next**.
3. Perform the following steps in the **Import Model File** dialog box:
 - a. In the **Project** box, select the project into which you want to import the model file.
 - b. Click the **Browse** button to the right of the **Model file** box to choose the XML mode file to be imported, for example, **business_objects.xml**.
 - c. Click the arrow in the **Extension file** box to choose the extension file into which the model elements are to be placed, for example, **business_objects.xml**.
During normal extension work, you set the active file to hold extensions.
 - d. Click **Finish**.
The data model is imported into the extension file.
4. To verify the model is imported, browse for new data model objects in the Business Modeler IDE views.

To see the data model in the extension file, access the **Navigator** view, open the project, expand the **extensions** folder, and double-click the extension file (for example, **business_objects.xml**) to open it in an editor view.

Caution

Never manually edit the XML files; this may corrupt data.

Activity

1. Package extension into a solution template

In this activity, you package the **CCC_Dev** project extension into a solution template.

2. Import a project.

In this activity, you import the **CCC_Dev_2** project provided in class.

Review questions

1. The template project directory and all of its contents should be managed in a source control management system (SCM).

- True
- False

2. What is the purpose of **Reload Data Model**?

Select one.

- Add a new version of the dependent templates.
- Add a template dependency.
- Load and validate your model.
- Manually edit a source file during a SCM merge from another user.
- Upgrade to the next release.

Summary

Topics learned in this lesson:

1. You can package extensions to the data model as a solution template and distribute the template for installation to a production environment.
2. The template project directory and all of its contents should be managed in a source control management system (SCM).
3. The Business Modeler IDE performs strict validations on the extensions in your template.
4. You can import either an entire project template or just a model file.

Lesson

12 *Organization hierarchy*

Purpose

The purpose of this lesson is to manage the organization hierarchy.

Objectives

After you complete this lesson, you should be able to:

- Define the organization hierarchy.
- Define administrative privileges.
- Create an account.
- Manage the organization hierarchy.
- Set up accounts manually.
- Import and export the organization.
- Search the organization.

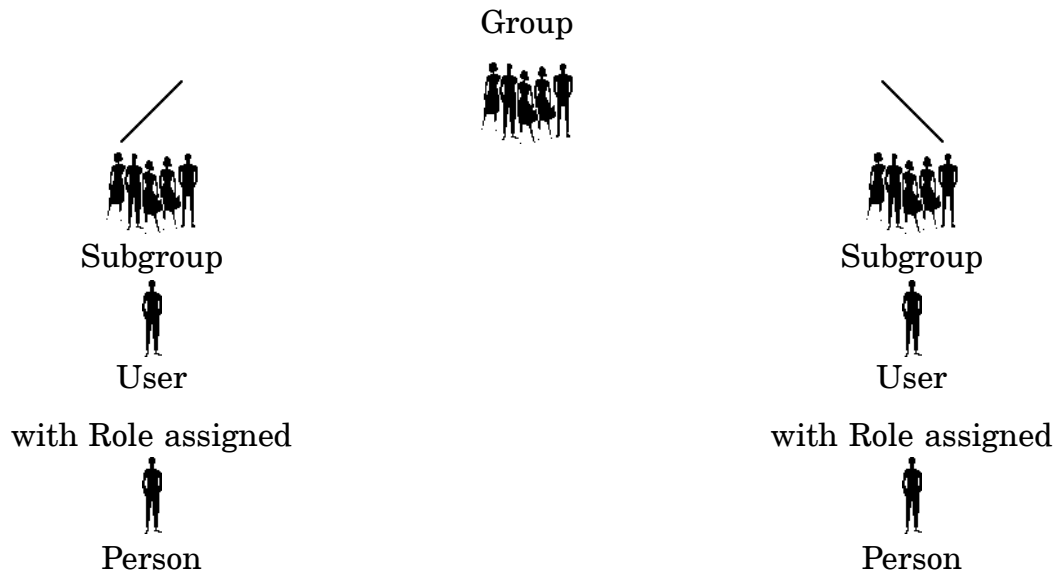
Help topics

Additional information for this lesson can be found in:

- [*Organization Guide*](#)
- [*Utilities Reference*](#) (see *Volume and database management utilities*)
- [*Utilities Reference*](#) (see *Data sharing utilities*)

Defining the organization

An *organization* is made up of groups. Groups contain subgroups, users, and persons.



Key points

- Groups are defined along project lines, not functional lines.
- Users take on a role in the group.
- The system grants data access based on group and role.
- Persons can have multiple user IDs, but typically they would have only one.

What is a person

Persons are individuals who work at your site. A person has properties such as **Person Name**, **Street Address**, and **Employee Number**.

Example



```
Green, Robert, M  
10824 Elm Street  
Wheeling, IL 60021  
Organization: Company BB  
Employee Number: 150  
Email: robertg@bblades.com
```

Key points

- Each person name must be unique.
- Consider creating all persons at your site using the following naming convention:
last name, first name, middle initial
- The Organization List displays persons in alphabetical order.
- Person definitions that are referenced by a user cannot be deleted.

What is a user

A *user* is a person with an account known to the Teamcenter system. One person can have several user accounts in Teamcenter.

The Teamcenter implementation of user is completely separate from any OS user account.

Example



Person

Green, Robert, M
10824 Elm Street
Wheeling, Il 60021



User

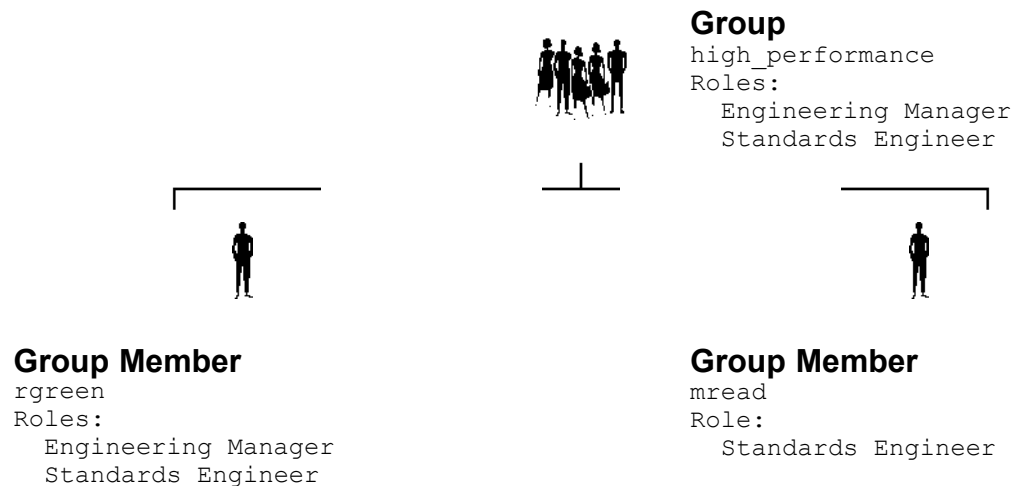
rgreen
Default group:
 high_performance
Role:
 Standards Engineer

Key points

- Each user must belong to a group.
- Each user is assigned to a role in the group.

What is a group and group member

A *group* represents a project in Teamcenter. Groups contain members (users) who take on a role or multiple roles in the group. Groups represent data ownership and therefore control data access.



Key points

- Groups are defined along project lines, not functional lines, but can define third-party organizations such as suppliers.
- A group defines a list of selected roles for the group. There can be multiple roles per group.
- Groups can include multiple group members.
- A group member is a user with an assigned role in a group. A group member can be assigned to more than one role. A group member can be a member of many groups. For example, Robert Green can belong to the **high_performance** and **standards** groups.

What is a subgroup

A *subgroup* is a group with another group designated as its parent. A subgroup can also be designated as a parent group itself. The position of subgroups within the organization hierarchy can be managed by parenting and reparenting groups.

Example



Key points

- A subgroup must have a unique group name but only within the parent group. The combination of the parent group name and subgroup name becomes the unique identifier for the subgroup.
- Subgroups are an excellent way to organize your users.
- Subgroups inherit access, volumes, and preferences from their parent.

Group hierarchies

Groups are organized into one or more trees or hierarchies. Each group has exactly one parent group (unless it is at the top or root of the hierarchy, when it has no parent) and can have one or more child groups.

The following list indicates the functional areas in Teamcenter that use group hierarchies.

| Functional area | Group hierarchy use |
|-----------------|--|
| Access Manager | A group can inherit from its parent. |
| Authorization | Authorization rules are inherited within the group hierarchy. |
| Volumes | Groups can inherit access to a volume from parent groups; therefore, you must consider volume access when modifying or moving hierarchical groups. |
| Preferences | Group preferences can be inherited from the parent group. |
| Mail | Mail can be sent to members of a group's subgroups, subgroups of subgroups, and so on, as well as to the named group only. |
| EPM | Signoffs can be assigned to members of a group's subgroups and members of the named group. |

Moving groups within the hierarchy

Moving groups within a hierarchy can affect group member access to data depending on how volumes are assigned.

- Existing files

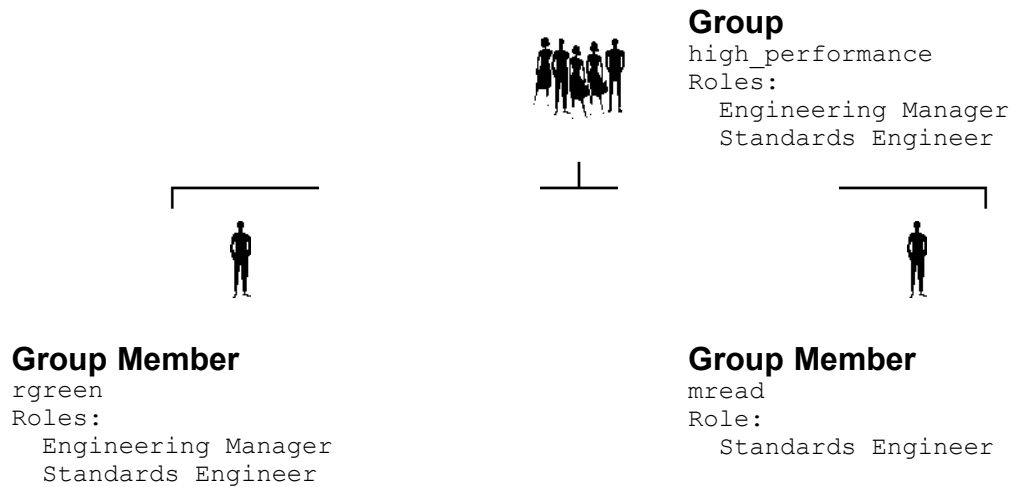
Restructuring groups makes no difference; the files continue to be available without any change in behavior.

- New files

If the group has its own assigned volume, restructuring makes no difference to volume access. Files continue to be saved to the assigned volume. However, if the group does not have its own volume, but instead inherits access to a volume from an ancestor group, there may be a difference after the restructuring. The restructured group now has a different set of ancestor groups; therefore, it either inherits access to a different volume or potentially has no volume access at all.

What is a role

A *role* is an object that models the type of work a user is expected to perform in a group.



Example

Robert Green is an Engineering Manager. In addition to his responsibilities as Engineering Manager, Robert must also perform standards work. Therefore, user **rgreen** has been assigned two roles in the **high_performance** group: **Engineering Manager** and **Standards Engineer**.

Key points

- One group can have many roles.
- One user can perform multiple roles in a group.
- Roles are created along functional lines.

Tip

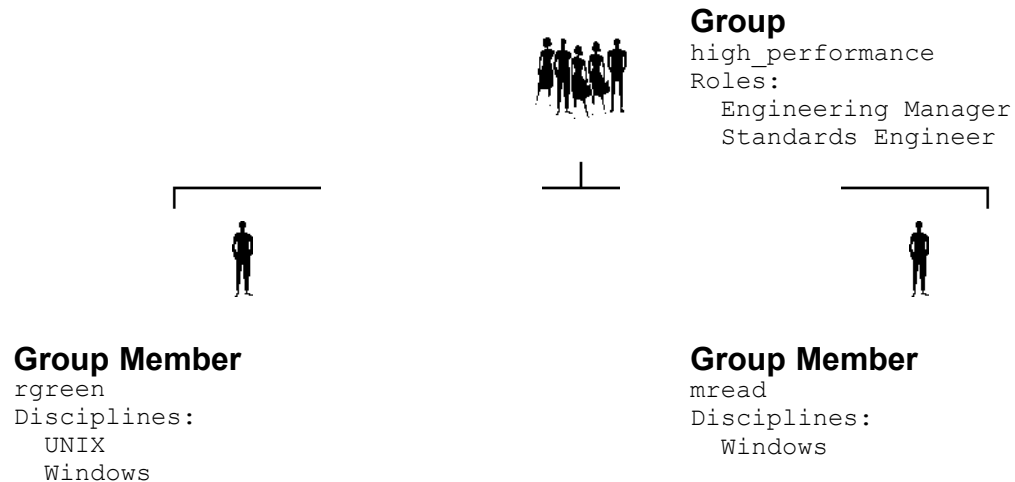
Use real-world descriptions, skills, and/or responsibilities.

What is a discipline

A *discipline* is an object that defines a set of users that have some behavior in a group.

Note

Discipline supports project management functionality.



Example

Robert Green is an expert in both UNIX and Windows operating systems. Therefore, user **rgreen** has been assigned two disciplines in the **high_performance** group: **UNIX** and **Windows**.

Key points

- One group can have many disciplines.
- One user can perform multiple disciplines in a group.

Passwords

Teamcenter enables companies to specify restrictions for passwords when creating user accounts. These password restrictions are controlled through preferences settings and take effect upon password creation. Existing passwords are not affected.

Companies can set the following restrictions:

- Minimum length required
- Mixed case required
- Minimum number of alpha or numeric characters required
- Special characters




Examples

```
PASSWORD_minimum_characters=0  
  
PASSWORD_mixed_case_required=false  
  
PASSWORD_minimum_alpha=0  
  
PASSWORD_minimum_digits=0  
  
PASSWORD_special_characters=#,*,%  
  
PASSWORD_minimum_special_chars=0
```

What is a volume

A *volume* is a location where files are stored. A volume equates to a directory on the operating system. Files stored in volumes are created by CAD applications or other third-party applications.

Example

| | |
|---|--------------------------------|
|  | <i>/disk 1/vols/user_vol</i> |
|  | <i>/disk 1/vols/group_vol</i> |
|  | <i>/disk 2/vols/group2_vol</i> |

Key points

- Teamcenter retains the volume location (directory) and the file name.
- Users should always access files in volumes through the Teamcenter product.

Activity

- Complete the organization worksheet.

In this activity, you fill out the training organization worksheet according to the scenario.

Organization worksheet example

| Person | User | high performance | standards |
|------------------|--------|---|--------------------|
| Green, Robert, M | rgreen | Engineering Manager Standards Engineer | Standards Engineer |
| Read, Mary, F | mread | Standards Engineer | |
| | | | |
| Default volume | | perf_vol | std_vol |

Review questions

1. What is the best logical order of creation for an organization?

Select one.

- Group, Role, User, Person
- Person, User, Role, Group
- Role, Group, Person, User
- User, Person, Group, Role

2. Where is it best to assign a volume?

Select one.

- Group
- Person
- Role
- User

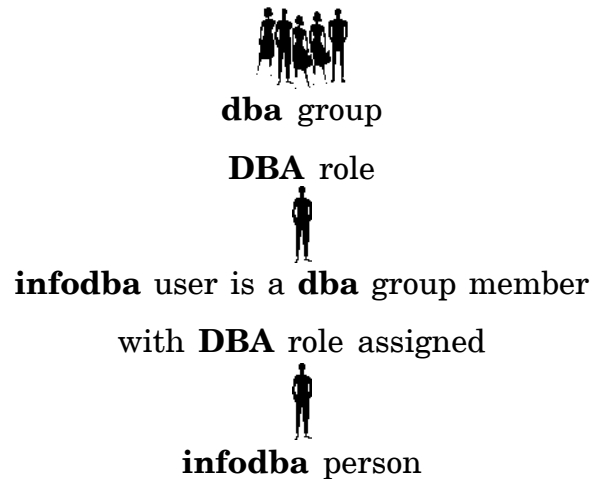
3. A volume is a location where files are stored.

- True

- False
4. A subgroup must have a unique group name.
- True
 - False

Define administrative privileges

infodba is the default account setup on installation. It consists of:



- **infodba** has all Teamcenter system-level privileges.
- **infodba** is the most powerful user in the system.

System administration accounts

Administrator privileges include the following settings:

| Setting | Privileges |
|--|---|
| Member of the dba group or another dba group | All system administration privileges. |
| Group administrator | Special access privileges for data owned by the group. |
| System administrator | Configures access authorization to administration applications and utilities based on group and role in group. |
| IP Admin role | Manages intellectual property licenses. For information about configuring and administering authorized data access, see the <i>Security Administration Guide</i> . |
| ITAR Admin role | Manages International Traffic in Arms Regulations licenses. For information about configuring and administering authorized data access, see the <i>Security Administration Guide</i> . |
| DBA role | Manages roles for that group and can assign the group administrator. |
| Member of the system group with the DBA role | Special access privileges for archive and restore. |
| Bypass option turned on | Overrides access protections and supersedes other privileges. |

Key points

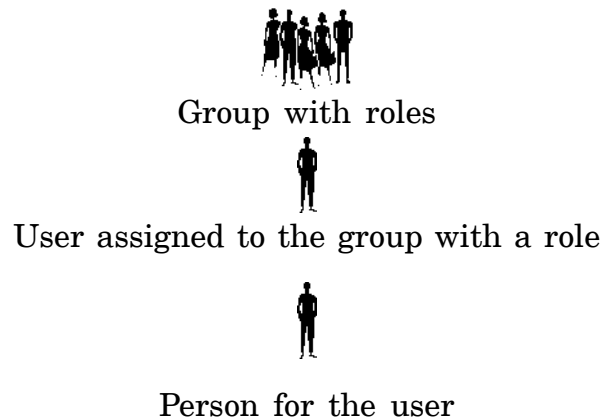
- Members of the **dba** group with the **DBA** role have complete system privileges.
- For each group, create at least one member who acts as the administrator for that group.
- The **DBA** role for a user in a non-**dba** group has no additional privileges over any other group member.

- For each group, create a **DBA** role to manage roles for that group and assign the group administrator.
- Group administrators can assign other group administrators in their group.

Create an account

Create an account by creating a user and adding them to a group.

- Users require a person.
- Groups require a role.
- Both users and groups can optionally specify a volume.

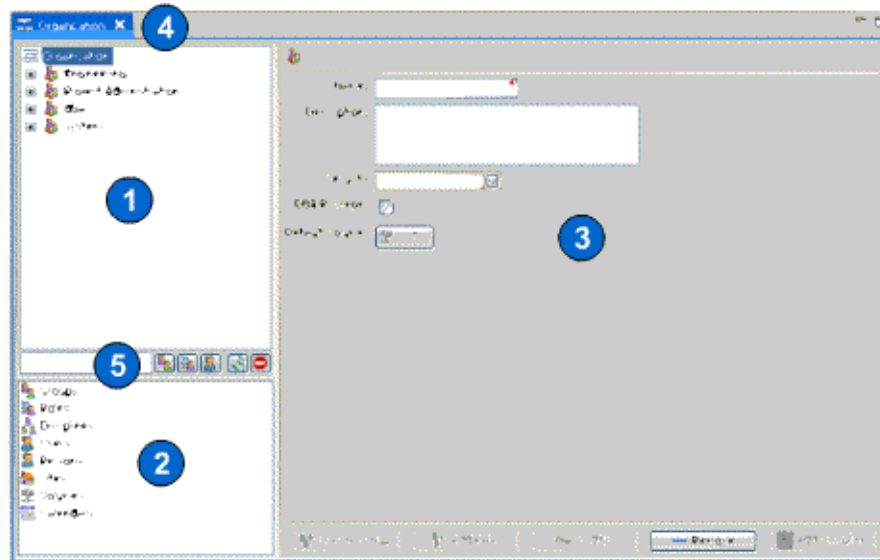


Key points

- Roles must be created before groups.
- Persons must be created before users.
- Volumes must be created before groups and users can reference a volume.

Organization interface

Three distinct panes make up the **Organization** pane:



1. The **Organization** tree enables you to view the structure of your organization at a glance. By expanding and collapsing branches of the tree, you can view and manage the organizational structure.
2. The **Organization List** tree enables you to view and manage the components of your organization by listing groups, roles, disciplines, users, and persons. Sites, volumes, calendars, and ADA licenses are also maintained through the **Organization List** tree.
3. The **Organization property** pane lists the organization object's properties and values. You can create, modify, or delete items as well as clear an object's property values.
4. **Tab title** displays **Organization**.
5. **Search field** is used to filter the **Organization** tree.

To view the organizations, from **My Teamcenter** window, choose **View→Organization**. You must have the appropriate permissions to view the **Organization** application.

The organization hierarchy can be built from bottom up or using a wizard to build the hierarchy from the top down.

Create the organization structure

Create an account by creating roles and groups.

1. Identify available volumes.
2. Create roles.
3. Create groups and subgroups.

Key points

- The system administrator is responsible for providing a list of volumes.
- Create new roles to be used in groups.

Warning

Do not delete the roles provided with Teamcenter. They are required for the product to function properly.

- Two groups are provided: **dba** and **system**.

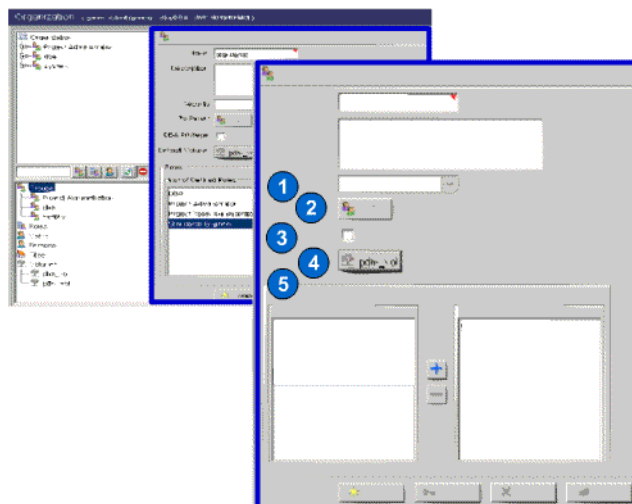
Warning

Do not delete the system group provided with Teamcenter. It is required for the product to function properly.

- Changing group or role names can impact workflow definitions.

Create groups

The **Group** pane has several important properties:



| | Property | Description |
|---|-----------------------|---|
| 1 | Security | Determines project-level security. Valid values are Internal , External , or the value can be left blank. |
| 2 | To Parent | Specifies the parent of this group, making this a subgroup. |
| 3 | DBA Privilege | Grants DBA privileges to members of this group. |
| 4 | Default Volume | Identifies default volume where files owned by this group are stored. |
| 5 | Roles | Identifies roles that apply to this group. |

Key points

- The group name becomes the volume's directory name when the **Default Volume** setting is used.

Example

Group Name: dba

Default Volume: dba_vol

Volume defined path: D:\dba_vol

Volume full path: D:\dba_vol\dba

Create persons

The **Person** pane has two important properties:

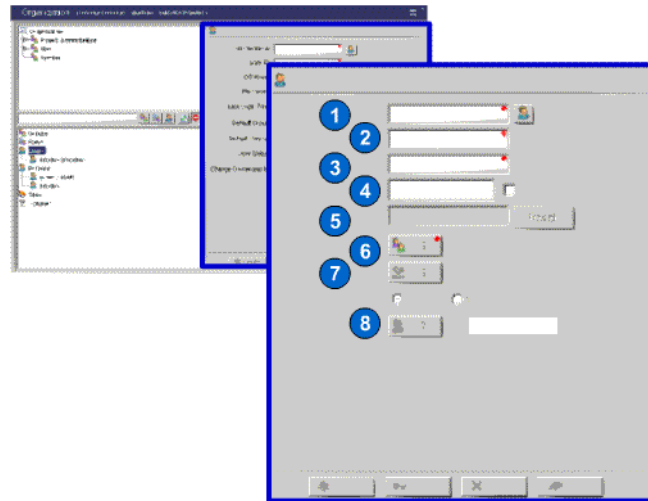
| Property | Description |
|---------------|--|
| Name | Enter the person's full name. Format the name as: <i>last name, first name, middle initial</i> Example <i>Green, Robert, M</i> |
| E_Mail | Enter the person's e-mail address. This is required for workflow notification. |

Key points

- The person name must be unique.
- The naming standard allows persons to be sorted by last name.

Create users

The **User** pane has several important properties:



| | Property | Description |
|---|----------------------------|--|
| 1 | Person Name | Specifies the person's name. Select from the list of predefined persons. |
| 2 | User ID | Specifies the unique Teamcenter user ID. |
| 3 | OS Name | Specifies the operating system user ID that is used to start Teamcenter. |
| 4 | Password | Specifies the password for the user ID. This must conform to password restrictions. |
| 5 | Last Login Time | Displays the last time this user logged onto Teamcenter. |
| 6 | Default Group | Specifies the group to which this user is assigned. |
| 7 | Default Volume | Specifies the volume to which this user is assigned. Defaults to the group's volume when left blank (recommended). |
| 8 | Change Ownership to | Changes ownership of this user's objects to another user. |

Key points

- **User ID** and **OS Name** are often the same.
- **Last Login Time** helps to determine when a user is deactivated. Reset will activate the user.
- Change ownership of the user's objects before deactivating or deleting an account. You can only change ownership if the user status is **Inactive**. A user cannot be deleted from the database if the user owns data and has approved a workflow.

Apply a new role to a group

Use the Organization Role wizard to add a role to a group.

1. Choose a group in the **Organization** tree.
2. Click the **Add Role** button in the **Group** pane.
3. Follow Organization Role wizard to add the new role.

Key points

- The first role added to a group becomes the default role for that group.

Apply new users to a group/role

Use the Organization User wizard to add a user to a group.

1. Expand a group in the **Organization** tree.
2. Choose a role under the group.
3. Click the **Add User** button in the **Role** pane.
4. Follow the Organization User wizard to add the new user.
5. Make the new group member the group administrator.

Group terms and concepts

| Term/Concept | Description |
|-----------------------------|--|
| Groups and subgroups | Groups and subgroups are project-oriented clusters of users. Each group has exactly one parent group (unless it is at the top, or root, of the hierarchy, when it has no parent) and may have one or more child groups (subgroups). |
| Default group | <p>When a user belongs to more than one group, one of them is designated as the default group. A default group must be designated for each user so that Teamcenter can store project files in a central location (for example, in the group volume).</p> <p>The user's default group is used at logon unless another group is specified.</p> <p>When Teamcenter Security Services is installed and the TC_SSO_SERVICE environment variable is set in the tc_profilevars file, logon uses the default Teamcenter group.</p> |
| Parent group | A parent group is used to organize a configuration of related subgroups. |
| System administration group | A Teamcenter administrator, also referred to as database administrator (DBA) or user with DBA privileges, is any member of a special system administration group and is the primary person responsible for maintaining the Teamcenter software, data volumes, and user accounts. Teamcenter creates one system administration group (dba) during installation. You can create others as the need arises. |
| system group | Teamcenter provides a special system group that is used for performing specialized tasks in an overall system administration strategy. Currently, members of the system group are primarily responsible for archiving and restoring objects. Although any Teamcenter user can mark an object for archive or restore, only members of the system group can perform the actual object archive and restore operations. This restriction also applies to members of a system administration group; they cannot perform archive and restore operations either. |

Warning

| Term/Concept | Description |
|---------------------|--|
| | <p>Do not delete the system group from the database under any circumstances. Teamcenter does not function properly without this group.</p> |
| Group administrator | <p>Although a user could belong to both the system group and a system administration group, members of the system group do not inherently have the privileges required to perform all Teamcenter administrative tasks.</p> <p>A group administrator is a group member with the following special privileges:</p> <ul style="list-style-type: none"> • Adding or modifying group members. • Adding or modifying person information for group members. |
| Group names | <p>Group administrators must be members of the group they are administering and these privileges are only valid within that group.</p> <p>A group is identified uniquely by the combination of its name and its parent. Two groups with different parents may have the same name, but two groups with the same parent may not.</p> <p>To distinguish between different groups with the same name, note the placement of the group within the Organization tree.</p> |

Activity

1. Create a new group and role.

In this activity, you create a new group called **high performance** and with a new **Manager** role.

2. Add a new user to a new group and role.

In this activity, you use the Organization User wizard to add **Maria Andretti** as the **high performance** group's manager.

Review questions

1. **infodba** is the most powerful user in the system.

- True
- False

2. The group administrator has what privileges?

Select one.

- All system administration privileges.
- Special access privileges for data owned by the group.
- Configures access authorization to administration applications and utilities based on group and role in group.
- Special access privileges for archive and restore.

Manual account generation

The **make_user** utility allows you to create your organization from a command line or script.

Why use the **make_user** utility script?

- **Increase productivity**
Load your entire organization with one command.
- **Disaster recovery**
Back up your organization.
- **Migration**
You can populate your test and production environments.

Key points

- You can run the **make_user** utility script in batch mode or multiple times.
- The **make_user** utility can be used to set group and volume defaults.

Warning

The **make_user** utility cannot be used to delete the organization objects.

make_user examples

Create a person and user and add them to a group with a specific role

Example

```
make_user -person="Smith, John" -user=smith -group=design  
-role=engineer
```

Note

The first group listed is the default group.

Warning

The group and role must already exist and the role must be in the group.

Assign a user to another group

Example

```
make_user -v -user=smith -group=dba -role=DBA
```

Note

User IDs must be unique.

Create a volume as a default to a group

Example

```
make_user -group=design -volume=design_vol -node=node1  
-path=/user/volumes/design_vol
```

Tip

See the **make_user** utility documentation for arguments.

Activity

- Create the organization hierarchy using the **make_user** utility.

In this activity, you use the **make_user** utility to add your organization.

Review questions

1. You can run the **make_user** utility script multiple times.
 - True
 - False

Import and export organization

The **dsa_util** utility allows you to distribute system administration data, such as organization data, from one site to another.

The **dsa_util** utility propagates Teamcenter administration data among multiple sites and allows administrators to:

- Manage system data from a central site.
- Support non-networked sites.
- Create reports of the results of a distribution operation.

Key points

- This utility should be used only for the initial migration of system objects. It is not recommended for use in maintaining system objects.

Export organization

Use the **dsa_util** utility with the following options to export:

| Argument | Description |
|------------------|--|
| -u | Specifies the user ID. This is generally infodba or another user with administration privileges. |
| -p | Specifies the password for the user ID. |
| -g | Specifies the group associated with the user. If used without a value, the user's default group is assumed. |
| -f | =exp to export the data. |
| -class | Identifies the system class or classes to be processed. One class argument is required for every class to export. |
| -filename | Specifies the path name of a text file to be used as output of system object information. If only the file name is given, the file is assumed to be in the user's current directory. Specify a file name using the .xml file extension. |

Example

```
dsa_util -u=infodba -p=infodba -g=dba -f=exp -class=Group  
-class=Role -class=Discipline -class=User -filename=OrgExp.txt
```

Import organization

Use the **dsa_util** utility with the following options to import:

| Argument | Description |
|------------------|---|
| -u | Specifies the user ID. This is generally infodba or another user with administration privileges. |
| -p | Specifies the password for the user ID. |
| -g | Specifies the group associated with the user. If used without a value, the user's default group is assumed. |
| -f | =imp to import the data. |
| -class | Identifies the system class or classes to be processed. One class argument is required for every class to export. |
| -filename | Specifies the path name of a text file to be used as input of system object information. If only the file name is given, the file is assumed to be in the user's current directory. Specify a file name using the .xml file extension. |

Example

```
dsa_util -u=infodba -p=infodba -g=dba -f=imp -class=Group  
-class=Role -class=Discipline -class=User -filename=OrgExp.xml
```

Search the organization

Expand the organization tree.



Organization



Group name



Role name








Person name (user ID)

Key points

- It is difficult to find a group, role, or person in large organizations.
- Use the search function to find groups, roles, or persons.

Search the Organization tree

1. Enter search text.
2. Click the group, role, or person button.

| Search text | Group | Role | Person | Refresh | Inactive members |
|-------------|---|---|---|---|---|
| |  |  |  |  |  |

Key points

- Wildcard values are valid.
- Search results appear in the **Organization** tree.
- To load the original **Organization** tree, click the refresh button with no search text.


Activity

- Search the organization.

In this activity, you use the search capability to find and update the organization hierarchy.

Review questions

1. Which search button is for a role?

- 
- 
- 

Summary

Topics learned in this lesson:

1. The organization hierarchy includes persons, users, roles, disciplines, and groups.
2. Create groups and roles to represent your projects and functional responsibilities.
3. Add users to groups to define their place in the organization.
4. Increase productivity by using a utility to load your organization.
5. The **dsa_util** utility allows you to import and export organization data.
6. Use the search function to find groups, roles, or persons.

Lesson

13 Query Builder definitions

Purpose

The purpose of this lesson is to create query definitions for a Teamcenter site.

Objectives

After you complete this lesson, you should be able to:

- Define query definitions basic concepts.
- Create and modify query definitions.
- Limit user access to query definitions.
- Import and export query definitions.

Help topics

Additional information for this lesson can be found in:

- [*Query Builder Guide*](#)

Define query definitions

Query definitions, also called *saved queries*, identify a search criteria used to find information in Teamcenter. Administrators define query definitions for end users.

Example

Find all folders named **Home**.

Find in the database all items that have been shipped.

Key points

- Saved queries allow you to quickly find product information in the database.
- To create queries you must be a user with system administration privileges or be granted authorization by a user with system administration privileges.
- Building query definitions requires knowledge of the Teamcenter POM schema, which is a hierarchical arrangement of classes, subclasses, and attributes.
- Limit the number of query definitions saved in the database and/or restrict the number of users allowed to create, modify, and delete query definitions to enhance Teamcenter usability.

Note

If there are more than 5,000 instances of the search class to be found, you must add an index to the attribute on which you are searching. Doing this helps the search performance. However, if the table is very small (that is, the class has less than 500 instances), a full-table scan is more effective than an index scan.

For more information, see *Application notes* in the *Query Builder Guide*.

Query Builder interface

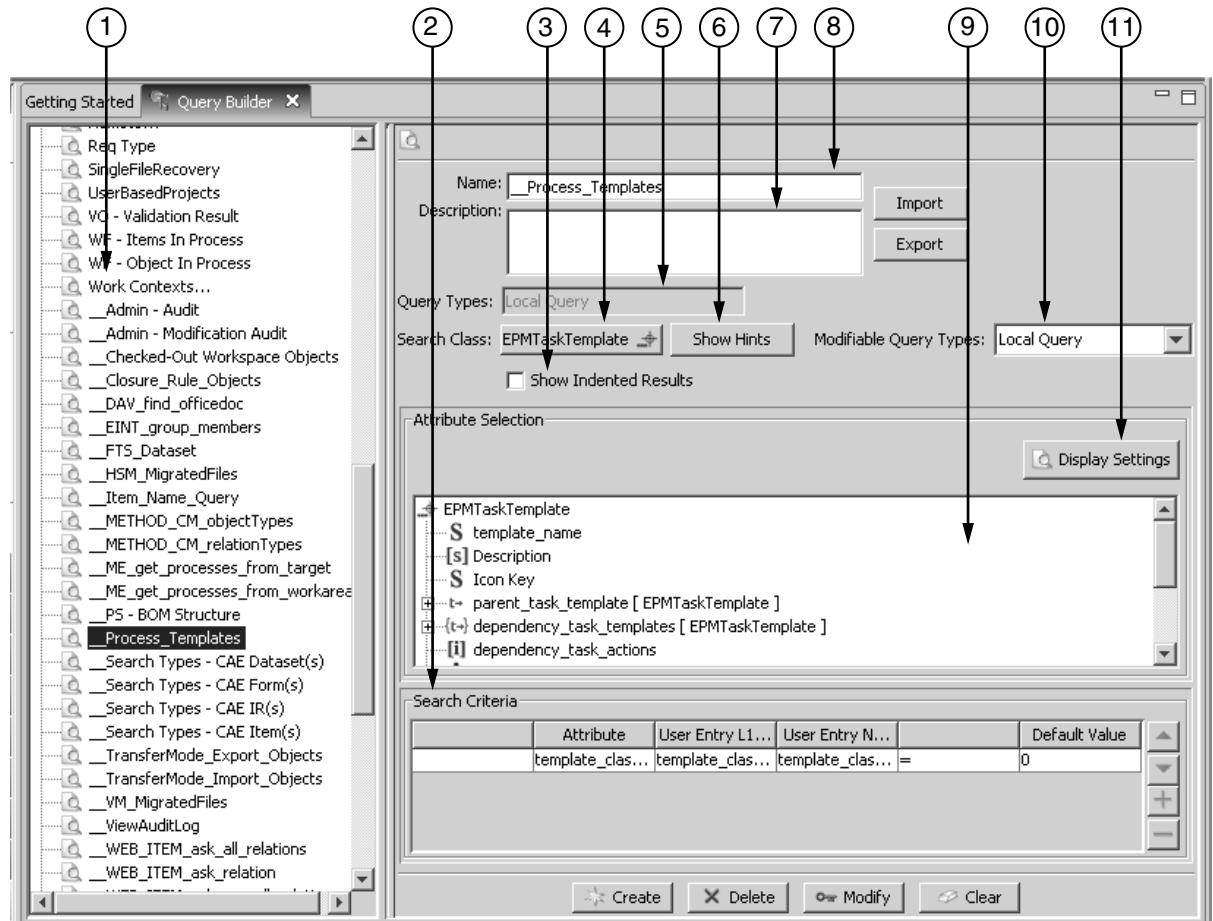


Figure 13-1. Query Builder interface

- 1 Displays all saved queries in the database. When you select a saved query in the tree, the details are displayed in the panes on the right side of the box.
- 2 Defines the search criteria clauses, using the following components:

Boolean Rules

The Boolean rules (**AND/OR**) are used to combine clauses to create a custom query.

Note

The indented search feature only supports **AND** clauses.







Attributes

The selected database attribute displays in this box.

| | | |
|---|--|--|
| | User Entry L10N Key | Specifies the localization key used to look up user entry names. The localization key-value pairs are defined in the qry_user_entry_names_locale.xml file. The value in this column can be modified and must be unique within the search criteria definition. |
| | User Entry Name | Displays the query box names as they will appear in the search form. The user name is the value of the localization key entered in the User Entry L10N Key column. If the key/value pair is not defined in the qry_user_entry_names_locale.xml file, the user entry name is the same as the key entered in the User Entry L10N Key column. The value in this column cannot be modified. |
| | Logical Operators | Matching values can be equal to, not equal to, less than, or greater than the value specified in your search clause. Matching values can also be null or not null. These conditions are called logical operators. |
| | Default Value | Unless the logical operator is_null or is_not_null is used, this box is required if you do not specify the User Entry Name when creating a query. You can specify the default value for the clause in any query you create. |
| 3 | Displays search results in a flat list from which you can navigate to related items for local and keyword query types. By default, this option is off. For more information about indented search results, see <i>My Teamcenter Guide</i> . | |
| 4 | Provides access to the Class Selection dialog box where you select your query class. | |
| 5 | Displays the query type of the currently selected query. | |
| 6 | Displays the dialog box used to select query hints that assist in navigating the schema and object relationships. | |
| 7 | Describes the purpose of the query or provides other pertinent information about the query. | |
| 8 | Displays the unique name of the query. | |

- 9 Displays the attributes of the selected class and either all inherited class attributes or only the direct attributes of the class, depending on the display setting you select.

The following icons indicate the attribute type:

| Icon | Attribute type |
|---|--------------------------|
| c | Character |
| [c] | Character array |
|  | Date |
|  | Date array |
| <i>d</i> | Double |
| <i>[d]</i> | Double array |
| <i>f</i> | Float |
| <i>[f]</i> | Float array |
| i | Integer |
| [i] | Integer array |
| b | Logical |
| [b] | Logical array |
| sh | Short |
| {sh} | Short array |
| S | String |
| [s] | String array |
| t→ | Typed reference |
| {t→} | Typed reference array |
| → | Untyped reference |
| [→] | Untyped reference array |
| e→ | External reference |
| {e→} | External reference array |
|  | Note |
|  | Note array |
|  | Typed relation |
|  | Untyped relation |



Class



External link default



vi overlay

10 Allows you to choose from the following valid query types:

Local Query Runs against the local database.

Keyword Search Query Performs keyword searches using a search engine.

The following query types are readable from Query Builder but can only be created, modified, or deleted using the **query_xml** utility.

User Query Calls **#USER_execute_saved_query** that returns objects for display.

Remote Query Runs against the published objects in the ODS.

User Exit Query Calls **#USER_query_execute** that returns column-value triples for display.

BOM Structure Query Generates BOM reports.

eIntegrator Query Created through the Global Services application; runs against an external database.

11 Indicates how query details are displayed in the box with the following values:

Class Attributes Displays only the class attributes.

All Attributes Displays all of the attributes.

Display Names Displays the presentation name (description) of the attribute.


Real Names Displays the actual attribute name as it appears in the database.

Note

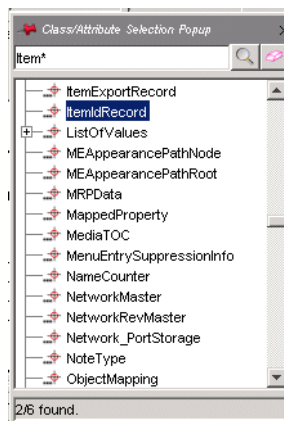
The **Display Names** and **Real Names** options are only available to the users with DBA privileges.

Class Selection dialog box

The **Class Selection** dialog box allows you to navigate and search the Teamcenter POM schema and choose classes and attributes for use in query definitions. To locate a class:

1. Type the class name (or partial name of the class along with a wildcard) in the box at the top of the dialog box.
2. Click the search button .

The number of classes matching your search are displayed at the bottom of the dialog box, and the first result is highlighted in the tree.



To navigate through the results:

Go to next match Press F3, Page Down, or Down Arrow key.

Go to previous match Press the Page Up or Up Arrow key.

Go to first match Press the Home key.

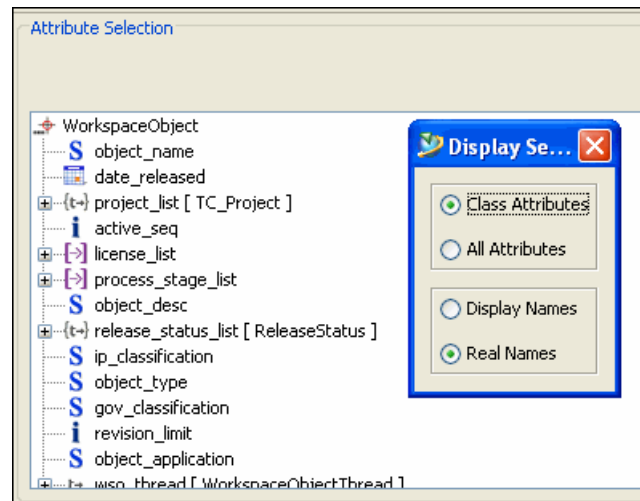
Go to last match Press the End key.

Properties for workspace objects

To create query definitions, you must understand the attributes defined on each class. Because there are many classes in the database, each with many attributes, you should understand the **WorkspaceObject** class hierarchy first.

The **WorkspaceObject** class defines the *common attributes* of all the workspace objects: folder, form, dataset, item, and item revision.

- The attributes without the plus \oplus symbol can be directly set when searching for objects in the **WorkspaceObject** class.
- The attributes with the plus \oplus symbol refer to another class in the class structure, which may have direct attributes, or may have more attributes with \oplus symbols.



Key points

- Use Query Builder to create customized searches for objects called *query definitions*.
- Query definitions are based on objects attributes.
- The **Name** and **Type** attributes are defined directly to the **WorkspaceObject** class.
- Some other attributes for the **WorkspaceObject** class are inherited from parent classes, but where these attributes are actually defined is not a concern at this time.

Query Builder creation features

Depending on your needs, you can create queries based on the following features:

- Queries using the hints feature

Query hints are provided to assist you in navigating the schema by presenting a relationship to traverse (for example, item-to-item master form) and the steps required to build that relationship into your query definition.

- Queries that include a keyword search

Including the **Keywords** attribute in a query definition allows you to develop queries to find dataset files that contain a specific keyword or combinations of keywords.

The Autonomy server must be installed and configured at your site to enable full-text keyword search functionality.

- Queries based on an existing definition

You can create different queries based on existing ones.

- Queries using the **IS_NULL** or **IS_NOT_NULL** operators

The **IS_NULL** and **IS_NOT_NULL** operators allow you to create queries to find objects with null attribute values, such as items that do not have descriptions, or objects with an attribute containing any value other than null. Clauses that use this operator are treated as having a fixed value; therefore, there is no need to enter a name and default value for the clause.

- Referenced-by queries

You can create queries using clauses on a reversed-reference relationship. For example, create a query using clauses on a reversed-reference relationship to find dataset objects that are referenced through an **IMAN_specification** relationship by an item revision.

- Queries based on classification attributes

With this function, and appropriate privileges, you create a saved query on Classification attributes, using Query Builder. In particular, you can create a saved query that combines Classification attributes and Teamcenter attributes.

Note

For more information, see the *Query Builder Guide*.

Create a query based on an existing query definition

To search in My Teamcenter, select a search type, which specifies criteria for a database query and distinguishes the databases to be queried. For example, an **Item ID** search queries the Teamcenter database for the specified item ID. You can create a new query for custom objects based on queries already defined in the database.

1. Select an existing query from the **Saved Queries** tree.
2. Type a unique name for the new query in the **Name** box.
3. Change the information in the **Name**, **Description**, or **Search Class** box, and/or **Search Criteria** table columns.
4. Click **Create**.

The query is created, and the name appears in the **Saved Queries** tree of the Query Builder application and in the list of **All Queries** in the Find application.

Activity

- Save as a new query definition.

In this activity, you create a general and a custom item queries definitions for a Teamcenter site.

You create the queries using existing ones and test them in My Teamcenter.

Review questions

1. What are query definitions?

Select one answer.

- Folders named **Home**
- Search criteria used to find information in Teamcenter
- Object attributes

2. To enhance Teamcenter usability, you need to _____ and _____.

Fill in the blanks.

- Limit the number of query definition saved in the database
- Avoid customizing query definitions
- Restrict the number of users allowed to create, modify and delete queries

3. A **Local Query** runs against the local database.

- True
- False

4. To navigate and search the POM schema you must use _____.

Fill in the blank.

- The **Search** dialog box in My Teamcenter
- The **Saved Queries** tree
- The **Class Attribute Selection** dialog box

5. You can create a new query based on an existing query definition.
- True
 - False

Create new query definitions

When creating a query definition, you must provide the following information:

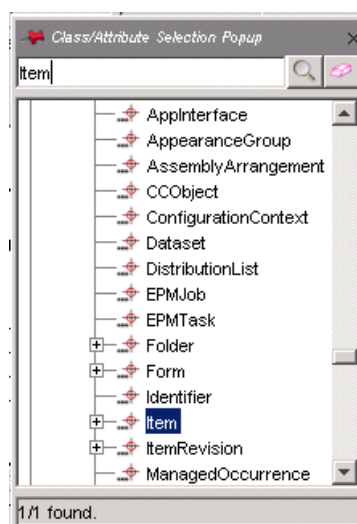
| Query definition | Example |
|------------------------------|--------------------------|
| Name | Find Home Folders |
| Modifiable Query Type | Local Query |
| Search Class | Folder |
| Search Criteria | Name = Home |

1. Remove any existing information from the Query Builder boxes by clicking **Clear**.
2. Type the name of the new query.
3. Add additional information about the query in the **Description** box.
This box is optional; it can be left blank.
4. Select the **Modifiable Query Type**.

The **Modifiable Query Type** box defaults to **Local Query** unless you click **Clear**.

5. Click **Search Class** .


The **Class Selection** dialog box appears.



6. Specify the desired search class by selecting an entry from this dialog box.

Tip

You can significantly expand or narrow the focus of your query depending on the class you select. It is best to limit the search to the lowest possible class in the hierarchy.

To locate a class, type the class name (or partial name and wildcards) in the box at the top of the window, and click the search button . The number of classes matching your search are displayed at the bottom of the dialog box, and the first result is highlighted in the tree. To navigate through the results:

Go to next match Press F3, Page Down, or Down Arrow key.

Go to previous match Press the Page Up or Up Arrow key.

Go to first match Press the Home key.

Go to last match Press the End key.

7. To display the search results in an indented or hierarchical form, select **Show Indented Results**.

8. Double-click to select at least one of the attributes in the **Attribute Selection** tree.

Direct attributes of the class are displayed in the tree. Reference classes and attributes can be accessed by double-clicking the **Referenced By** node in the attribute tree.

The attribute is added to the **Search Criteria** table.

9. Specify the desired search criteria. You must specify the following required search criteria boxes:

- **Attribute**
- **User Entry L10N Key**
- **Logical operators**

10. Click **Create**.

The query is created, and the name appears in the **Saved Queries** tree of the Query Builder application and in the list of **All Queries** in the Find application.

Using search criteria clauses

Search criteria is specified using statements or clauses in the **Search Criteria** table. Each clause searches the class and examines a specific attribute in that class and each clause can examine only one attribute. Therefore, if you want to build a complex query that examines multiple attributes, you must build several search clauses (one for each attribute you want to examine).

When you perform your search (run your query), Teamcenter examines the attribute specified in each of your search clauses and looks for values that match your search criteria.

| Search Criteria | | | | |
|-----------------|------------------|-----------------|---|---------------|
| | Attribute | User Entry Name | | Default Value |
| | person.user_name | Person Name | = | |
| AND | user_id | User Id | = | |
| | | | | |

Table 13-1. Search Criteria table element descriptions

| Element | Description |
|-----------|---|
| Attribute | <p>Entries in the Search Criteria table are based on attributes that you have selected for that query. To define an attribute for a query, double-click the desired attribute in the Attribute Selection tree. It is then added to the Search Criteria table. To add a node to the clause table, select the node and use the plus (+) button located to the right of the clause table. For example, to define a query to find classified workspace objects, first select the WorkspaceObject class, and then expand the Referenced_by node. Double-click the lcm0 node in the tree, expand it and display the eref node. Select the eref node and click the plus (+) button. The eref node is added to the clause table.</p> |

Table 13-1. Search Criteria table element descriptions

| Element | Description |
|----------------------------|--|
| User Entry L10N Key | <p>Defines the user entry name that appears when a query is chosen in the My Teamcenter Search pane. Type the name of a key defined in the qry_user_entry_names_locale.xml file or any other key name.</p> <p>Note</p> <p>Key names can be any alphanumeric string but must be unique within the search criteria definition.</p> <p>If you enter a key that has a corresponding value in the qry_user_entry_names_locale.xml file, the corresponding value is displayed in the User Entry Name column of query forms in the My Teamcenter Search pane. If you enter a key/text that does not have a corresponding value in the qry_user_entry_names_locale.xml file, the key/text is displayed in the User Entry Name column of query forms in the My Teamcenter Search pane.</p> |
| User Entry Name | <p>Displays the name that appears when a query is chosen in the My Teamcenter Search pane. The name displayed is the value of the localized key entered in the User Entry Key column if the key-value pair is found in the qry_user_entry_names_locale.xml file. If the key-value pair is not contained in the qry_user_entry_names_locale.xml file, the User Entry Name box displays the text entered in the User Entry Key box.</p> |
| Default Value | <p>Default values can be specified for the query clauses. Default values can be entered as a text string or selected from the associated list of values, where applicable. After the value is set, press the Enter key to save the default value. This box is only required when you do not specify the user entry name.</p> <p>The following keyword variables can be used to display default values in the query form:</p> <ul style="list-style-type: none"> • \$USERID • \$USERNAME |

Table 13-1. Search Criteria table element descriptions

| Element | Description | | | | | | | | | | | | |
|-------------------|---|------------------|-------------|---|-----------|----|---------------|---|---------------|----|---------------------------|---|------------|
| | <ul style="list-style-type: none"> • \$GROUP • \$TODAY <p>The values displayed in boxes for which the \$USERID, \$USERNAME, and \$GROUP variables are used as a default value correspond to the end user who is running the query. The \$TODAY variable displays the current date. These variables are used in the default Teamcenter queries. If you make any change to the default queries, the modified values are displayed unless you explicitly enter the variable name over its displayed value.</p> | | | | | | | | | | | | |
| Boolean rules | One or more search clauses are combined using Boolean (that is, AND/OR) rules to create a custom query. When you use AND clauses together, both must be satisfied in order to return a match (both this clause and that clause). When you use OR clauses together, either can be satisfied in order to return a match (either this clause or that clause). Keyword clauses do not support the OR rule. | | | | | | | | | | | | |
| Logical operators | <p>The find feature provides both power and flexibility in determining what is a matching value. Matching values can be equal to, not equal to, less than, or greater than the value specified in your search clause. These conditions are called logical operators. You must specify one of the following logical operators in each search clause:</p> <table> <tr> <th>Logical operator</th><th>Description</th></tr> <tr> <td>=</td><td>Equal to.</td></tr> <tr> <td>!=</td><td>Not equal to.</td></tr> <tr> <td>></td><td>Greater than.</td></tr> <tr> <td>>=</td><td>Greater than or equal to.</td></tr> <tr> <td><</td><td>Less than.</td></tr> </table> | Logical operator | Description | = | Equal to. | != | Not equal to. | > | Greater than. | >= | Greater than or equal to. | < | Less than. |
| Logical operator | Description | | | | | | | | | | | | |
| = | Equal to. | | | | | | | | | | | | |
| != | Not equal to. | | | | | | | | | | | | |
| > | Greater than. | | | | | | | | | | | | |
| >= | Greater than or equal to. | | | | | | | | | | | | |
| < | Less than. | | | | | | | | | | | | |

Table 13-1. Search Criteria table element descriptions

| Element | Description |
|-------------|---|
| <= | Less than or equal to. |
| IS_NULL | Indicates that the reference attribute value must be blank (not set). |
| IS_NOT_NULL | Indicates that the reference attribute must have a value. |

Note

Logical operators can only be used for string attribute types.

You can search for ranges of values using the >, >=, <, <= logical operators or invert search criteria using the != logical operator.

Activity

- Create a new query definition.

In this activity, you create a new administrative query to find all the **Home** folders existing in the database.

This is a useful query for Teamcenter administrators.

Review questions

1. What information is needed to create a query definition?

Select one answer.

- A query name, a user entry name, and a description
- A query name, a description, and a type
- A query name, a type, a search class, and a search criteria

2. You can significantly expand or narrow the focus of your query depending on the class you select. It is best to limit the search to the lowest possible class in the hierarchy.

- True
- False

3. What do you need to specify a search criteria?

Select one answer.

- Selected keywords
- Statements or clauses in the **Search Criteria** table for each attribute
- A real name and a display name

4. You can use Boolean rules and logical operators to define the **Search Criteria**.

- True
- False

5. What is the purpose of **IS_NULL** operator?
- To indicate that the reference attribute value must be less than or equal to a value
 - To create a custom clause using **AND/OR** rules
 - To find objects with undefined attributes values

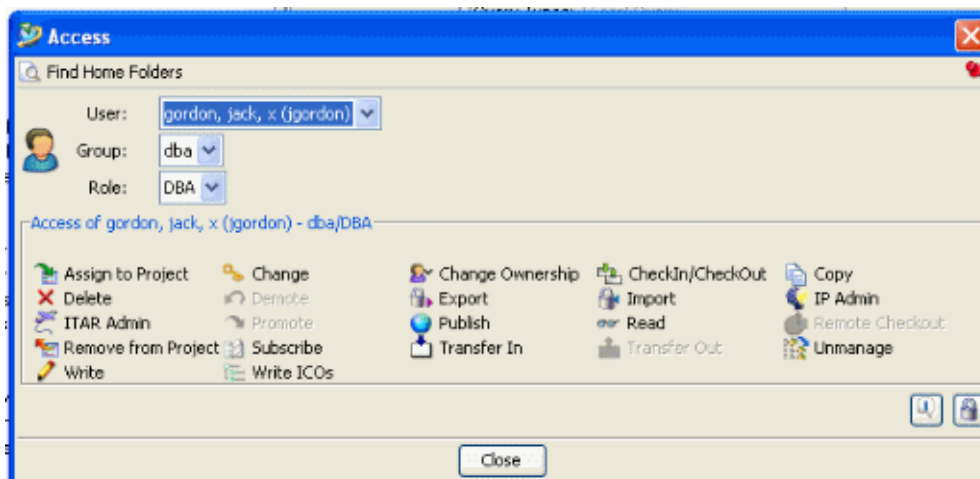
Limit access to query definitions

When query definitions are created, all users have read access to see and execute the queries.

Some query definitions, like the **Find Home Folders** query, may only be applicable to administrators. Because of this, administrators may need to restrict access to some query definitions.

Query access interface


Right-click the query definition and choose **Access** to see the **Access** dialog box.



Access control list

You use the **Access Control List**  to modify the current access.

Add the following accessors to the bottom of the list to grant access to administrators and revoke access for everyone else.

| Type of accessor  | Read  | Named ACL  |
|--|--|---|
| World | ✗ | Object |
| System Administrator | ✓ | Object |

Key points

- Using named ACLs, you control who can execute query definitions.
- The additional **World** named ACL revokes read privileges from all users.
- The additional **System Administrator** named ACL allows only administrators to use the query.

Optional activity

- Limit access to a query definition.

In this activity, you change the protection settings on a query definition to make it available only to administrators.

Key points

- You control who can execute query definitions.

Review questions

1. Saved queries are not subject to standard object protection.

- True
- False

2. How can you control user access to searches?

Select all that apply.

- Add accessors to grant or revoke access
- It is not possible to control the access to searches in My Teamcenter
- Create an **Object Named ACL**

Custom item type query definitions

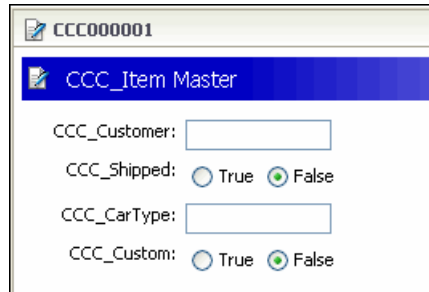
The general purpose of the query engine is to search the database for many kinds of data. It is often necessary to build queries to find items and item revisions based on the item master and item revision master form attributes and list of values.

Query forms are dialog boxes that provide user access to saved database searches.

Example

Create a query search based on the object type, object name, owning user, and owning group attributes, or you can also based the search on custom form attributes such as customer name, shipped date, and supplier.

For example, follow this procedure to build a query to find all custom CCC items based on the master form **CCC_Customer** and **CCC_Shipped** attributes.



1. Name the query: **CCC_Item**
2. Select the class: **Item**
3. Build the type, part number, CCC_Customer, and CCC_Shipped attributes clauses in the **Search Criteria** table:

| | Search attribute | User Entry L10N key | User entry name | Op | Default Value |
|-----|------------------|------------------------|--------------------|----|------------------|
| AND | object type | | Type | = | CCC_Item |
| AND | item_id | Part Number | Part Number | = | |
| AND | CCC_Customer | Customer | Customer | = | |
| AND | CCC_Shipped | Shipped | Shipped | = | TRUE |

4. Create the query definition.
5. If necessary, define the user access to the query definition.

Key points

- By clearing the **User Entry L10N Key** box, the clause is hidden from the user.
- **CCC_Customer** and **CCC_Shipped** are custom attributes found on the custom **CCC_Item Master** form.
- Use the **ACL Control List** dialog box to modify the query access and restrictions.

Activity

- Create a custom item type query definition.

In this activity, you modify a query to add custom form attributes to find different types of data existing in the database.

Review questions

1. The general purpose of the query engine is to search the database for one kind of data.

- True
- False

2. What are query forms?

Select one answer.

- Dialog boxes that provide user access to saved database searches
- A class structure
- Custom named ACLs

3. By clearing the **User Entry L10N Key** box, the clause is hidden from the user.

- True
- False

Import and export query definitions

Query definitions can be exported and saved as XML files that can be shared with other Teamcenter sites. Conversely, query data saved in XML files can be imported into Teamcenter. The XML files are parsed and verified before the data is imported.

Key points

- Export query definitions

Query definitions can be exported and saved as XML files that can be shared with other Teamcenter sites.

- Import query definitions




Conversely, query definitions saved in XML files can be imported into Teamcenter.

- The **Verify** button is used to validate that the POM class matches existing classes in the database before importing.

Note

It is possible that data that is correctly formatted in the XML file may be incompatible with the local database schema, which results in errors when you attempt to create the query definition using the imported data.

Export query definitions


1. Select the query in the **Saved Queries** tree that you want to export.
The system displays the query definition in the right pane of the **Query Builder** pane.
2. Click the **Export** button.
The system displays the query, in XML format, in the **Print** dialog box.
3. Click the **Save** button  to save the definition to a user-specified file.
The system displays the **Save** dialog box.
4. Determine the directory to which the file is saved.
5. Type a name in the **File name** box, including the **.xml** file extension.
6. Click the **Save** button  .
The system saves the file to the specified location and closes the **Save** dialog box.
7. Click the **Close** button  in the **Print** dialog box.

Import query definitions

Perform the following steps to import a query definition from an XML file and create the corresponding query in the Teamcenter database:

1. Click the **Import** button in the **Query Builder** dialog box.

The system displays the **Import** dialog box. The last query definition file that was imported to Teamcenter is displayed.

2. Click the **Browse** button  to locate the XML file containing the definition you want to import.

The system displays the **Read Query Definition** dialog box.

3. Locate the XML file and click the **Import** button.

The system displays the contents of the XML file in the **Import** dialog box.

4. Click the **Verify** button.

If the file format is valid, the query data is displayed in the **Query Builder** pane. If parser errors are encountered, an informational message describing the nature of the errors is displayed.

5. Click **OK** to load the query in the **Saved Query** tree and dismiss the **Import** dialog box.

6. Optionally, modify the name, description, or query clauses.

7. Click the **Create** button.

The system verifies that the definition is compatible with the local database schema. If so, the query is saved in the database. If not, an error message describes the discrepancies.

Activity

1. Import a query definition.

In this activity, you import the **user_data_3_equal_MAKE.xml** and **is_rejected** query definitions to the database. These queries are used as conditional queries for workflow processes.

2. Export a query definition.

In this activity, you export the **Find Home Folders** query definition to share it with other Teamcenter sites.

Review questions

1. Query definitions can be shared with other Teamcenter sites.

- True
- False

2. How are the query definitions exported?

Select one answer.

- Using XML file format
- Creating a compressed file
- It is not possible to export query definitions from Teamcenter

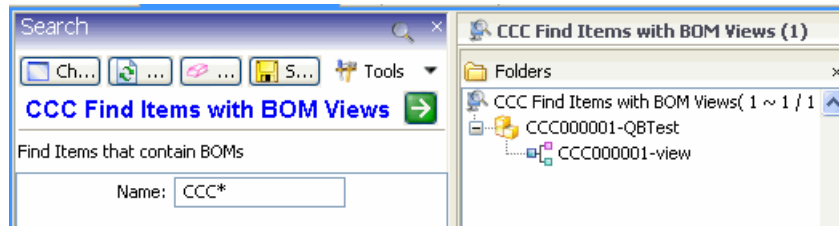
3. When importing query definitions, the XML files are parsed and verified before the data is imported.

- True
- False

Create a referenced-by query

You can create queries using clauses on a reversed-reference relationship.


You can use the **Show Indented Results** option to display the query results in an expanded tree format. The results hierarchy shows as many sublevels as determined by the query definition in the results pane.



The following example illustrates the process of creating a query using clauses on a reversed-reference relationship. In this case, the purpose of the query is to find dataset objects that are referenced, through an **IMAN_specification** relationship, by an item revision with a specific name.

1. In the **Name** box, type a name for the query. Query names must be unique.
2. Optionally, type a description of the query in the **Description** box.
3. Select **Show Indented Results**.

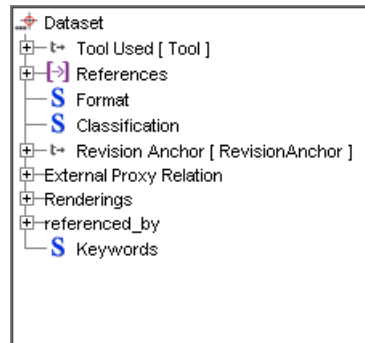
The **Show Indented Results** option allows the results to appear in an expanded tree format showing as many sublevels as determined by the query definition.

4. Click the **Search Class** button  to select the target class for the query.

The **Class Selection** dialog box displays the POM schema in tree format.

5. Select **Show all Attributes** under display settings.
6. Expand the **POM_application_object** class and locate the **WorkspaceObject** class.

- Expand the **WorkspaceObject** class and select the **Dataset** class by double-clicking it. The dataset is now displayed on the **Search Class** button. The **Dataset** class and its attributes are displayed in the **Attribute Selection** pane.



- Double-click the **Referenced By** node in the **Attribute Selection** pane. The **Class Attribute Selection** dialog box displays.

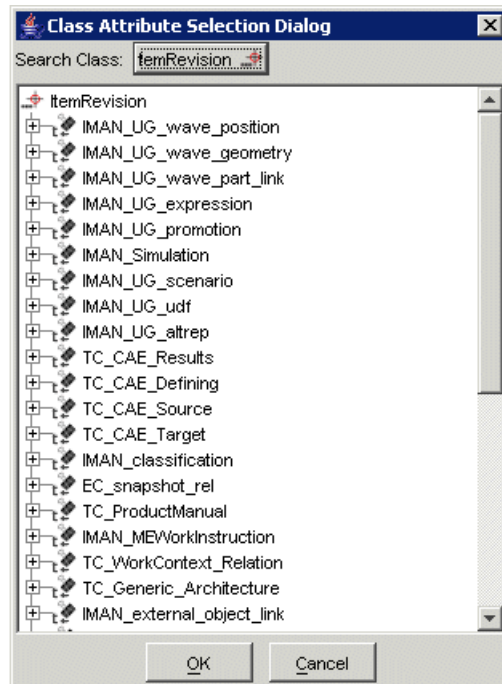
- Click the **Search Class** button  to select the referencing class for the query.

The **Class Selection** dialog box displays the POM schema in tree format.

- Expand the **POM_application_object** class and locate the **WorkspaceObject** class.

- Expand the **WorkspaceObject** class and select the **ItemRevision** class by selecting it and closing the dialog box.

The referencing class and its attributes are displayed in the **Class Attribute Selection** dialog box.

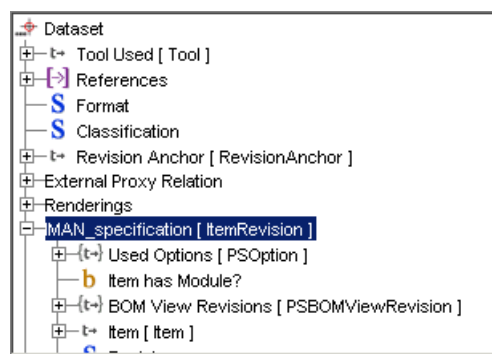


Note

Only those attributes that may reference the objects of the class being queried, in this case, the **Dataset** class, are displayed in the dialog box.

- Select a referencing attribute, in this case **IMAN_specification**, by double-clicking the node.

The referencing attribute, **IMAN_specification**, and class, **ItemRevision**, are displayed in the **Attribute Selection** area.



13. Select the attributes of the referencing class on which you want to build query clauses. In this case, locate and double-click the **object_name** attribute. The attribute is displayed in the **Search Criteria** table. Note that the display name of the attribute is **ItemRevision←IMAN_specification.object_name**. The \leftarrow symbol indicates a reversed-reference relationship.
14. Type a key name in the **User Entry Key** box.
15. Click the **Create** button to create the query.

The query is added to the **Saved Queries** tree of the **Query Builder** pane and is also available in the **Select a Query** list in My Teamcenter. For more information, see *My Teamcenter Guide*.

Activity

- Create a referenced-by query to find items that contain BOM views.

In this activity, you create a query to find items with BOM views using clauses on a reversed-reference relationship. You use the **Show Indented Results** option to display the query results in an expanded tree format.

Review questions

1. You can create queries using clauses on a reversed-reference relationship.

- True
- False

2. What is a referencing attribute?

Select one answer.

- Any object attribute
- An attribute that may reference the objects of the class being queried
- An attribute with an **IS_NULL** value.

3. What is the **Show Indented Results** option used for?

Select one answer.

- To display the results in an expanded tree format
- To display object attributes
- To save the search history

Summary

Topics learned in this lesson:

1. Define saved queries.
2. Create saved queries.
3. Limit access to saved queries.
4. Import and export query definitions.
5. Create referenced-by queries.

Lesson

14 *Preference management*

Purpose

The purpose of this lesson is to manage Teamcenter preferences.

Objectives

After you complete this lesson, you should be able to:

- Define preferences.
- Identify key Teamcenter preferences.
- Modify preferences.
- Import and export preferences.

Help topics

Additional information for this lesson can be found in:

- [*Getting Started with Teamcenter 2007*](#) (see *Managing options and preferences*)
- [*Preferences and Environment Variables Reference*](#) (see *Introduction to preferences*)

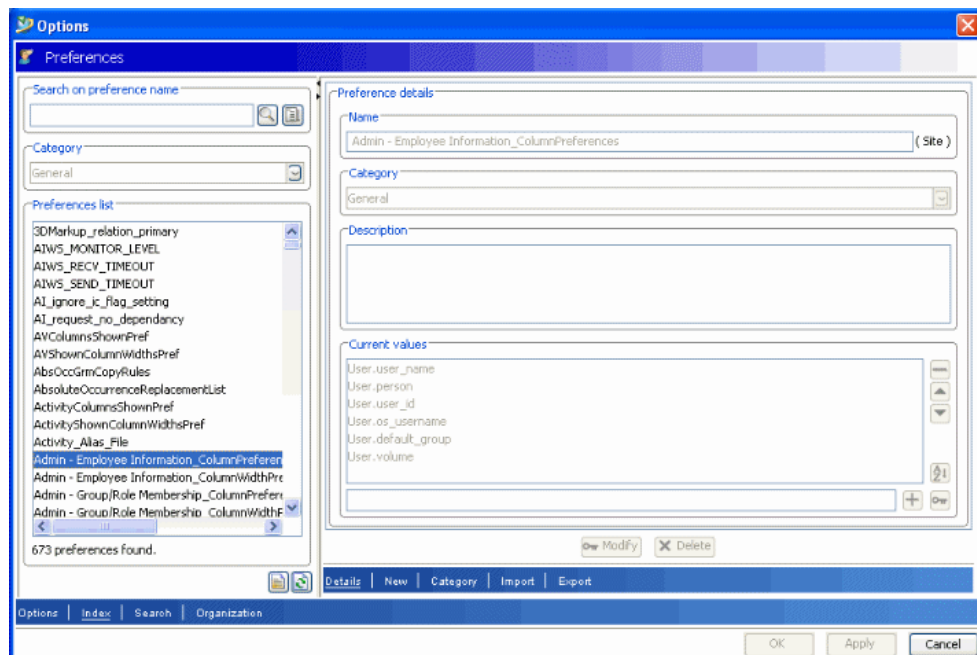
Introduction to preferences

Preferences are special environment variables stored in the Teamcenter database. They are read during Teamcenter application usage. Every application may have preferences associated with it.

Preferences allow administrators and users to configure many aspects of a session, such as:

- Logon mode and password requirements
- User interface
- Business rules

Choose **Edit® Options** to open the **Options** dialog box. Use the **Options** dialog box to search for preferences, set preference values, create new preferences, and remove existing preferences.



Note

The **tc_preferences.xml** file is read during system startup for loading the site preferences.

Do not to modify the COTS **tc_preferences.xml** file. You can create a new XML file to create or modify preferences.

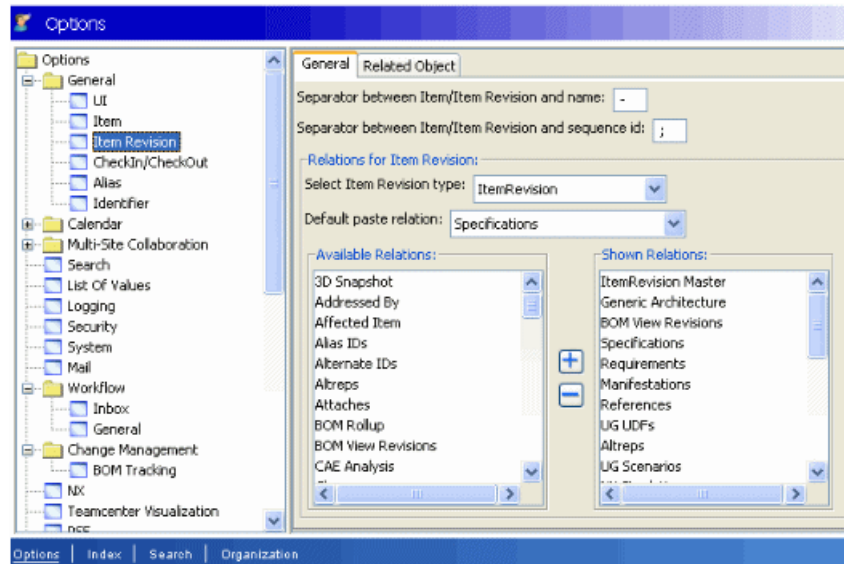
Key points

- Preferences are configuration variables stored in the database that are read when a Teamcenter session is initiated.
- Preferences are categorized according to the technical area of Teamcenter they affect.
- Only a system administrator can view and edit all preferences in Teamcenter.
- You can search for a preference in the database, modify the preference values, delete the preference, or add a preference to the system.
- You can work directly with the preferences and set values in text format.
- You can generate a report that lists all preferences that are customized at the site, or for selected group, role, or user.

Creating and editing preferences

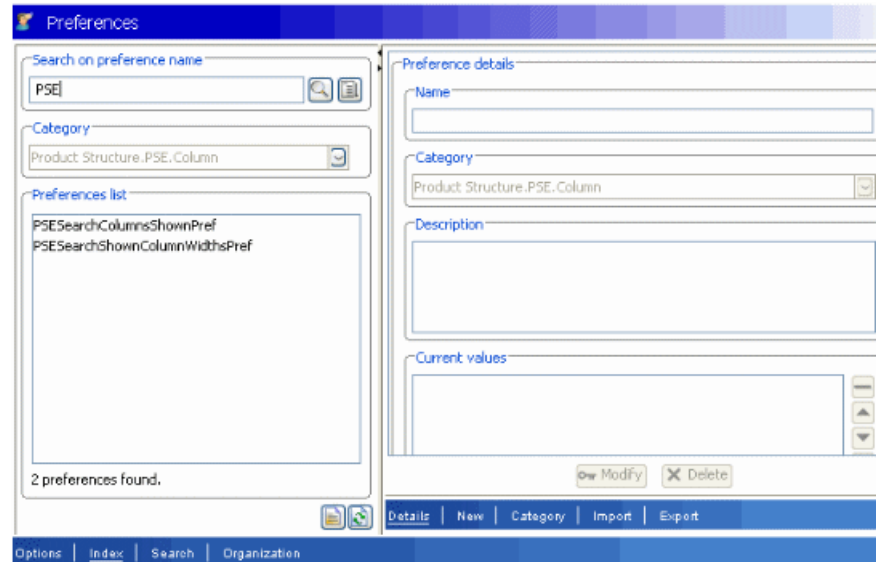
To create and edit preferences you open the **Options**, **Preferences**, **Search**, and **Organization** panes.

The **Options** dialog box enables you to work with a limited set of preferences in the form of *options*, which are presented by category. These are available from the **Options** tab of the dialog box.



Preferences pane

You can also work directly with the entire preference set, available from the **Index** tab of the dialog box. From this view, you can search for preferences in the database, modify preference values, delete preferences, and add preferences to the database.

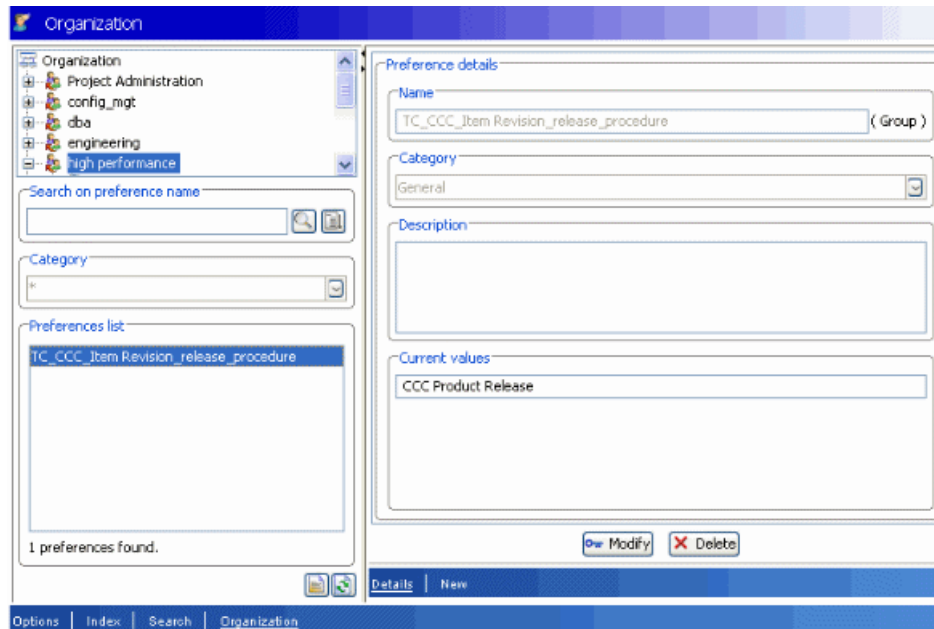


Organization pane

The **Organization** pane displays the preferences applied to groups and roles in the organization.

Users can see the preference applicable to their site, group, roles and user scopes, but they cannot see other users' preferences. Group administrators can see the preferences for the site, the group that they administrate and roles and users belonging to the group.

A system administrator can see all the preferences for the site, groups, roles, and user.



Key points

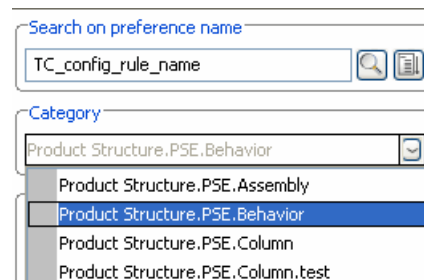
- In the **Options** pane, you work with a limited set of preferences in the form of options.
- In the **Index** pane, you search for preferences based on a preference name.
- In the **Search** pane, you search for preferences based on keywords and wildcards.
- In the **Organization** pane, you view the preferences applicable at various scopes like site, group, role, and user.

Preference categorization

- Preferences are categorized according to the technical area of Teamcenter they affect.
- Searches for preferences can be limited to certain categories to obtain results more quickly.
- Categories may contain subcategories to allow easier location and identification of a specific preference.

Example

The **Product Structure** category contains the **PSE** subcategory that also contains the **Behavior** subcategory. The **TC_config_rule_name** preference is one of the preferences categorized in this group.



The screenshot shows a web-based interface for managing preferences. At the top, there is a search bar labeled "Search on preference name" with the text "TC_config_rule_name" entered. Below the search bar, there is a "Category" dropdown menu. The dropdown is open, showing a list of categories: "Product Structure.PSE.Behavior", "Product Structure.PSE.Assembly", "Product Structure.PSE.Column", and "Product Structure.PSE.Column.test". The "Product Structure.PSE.Behavior" category is currently selected and highlighted in blue.

- The Teamcenter administrator can create new categories or subcategories that are appropriate for your business practices.

Key Teamcenter preferences

Table 14-1. Configuration preferences

| Preference | Category | Default value | Description |
|-------------------------------|-----------------------------------|----------------|---|
| PSE_default_view_type | Product Structure.PSE.Behavior | view | Sets the default BOM view used when opening PSE or pasting in an assembly when more than one BOM view is used at the site. |
| AE_dataset_default_keep_limit | Interface.Rich Client | 3 | Sets the number of old dataset versions to store in database. When this number is exceeded, the oldest dataset version is deleted if it is not referenced. |
| TC_config_rule_name | Product Structure.PSE.Behavior | Latest Working | Sets default revision rule that is used when opening and printing BOM views or BOM view revisions. |
| TC_auto_login | Interface.Rich Client | true | Enables (true) or suppresses (false) autologin feature for the entire site. Autologin uses operating system user names and passwords to log on to Teamcenter. |

Table 14-1. Configuration preferences

| Preference | Category | Default value | Description |
|-------------------------------|-------------------------------|-------------------------|--|
| Incremental_Change_Management | Product Structure.PSE.Display | false | Enables (true) or disables (false) incremental change functionality. |
| ICS_classifiable_types | Classification | Item and Item Revision | Specifies the list of Teamcenter types that can be classified. |
| QRY_dataset_display_option | General | 1 | Determines whether the latest version or all versions of a dataset objects are displayed when query result are returned. |
| QRYColumnsShownPref | Interface.Rich Client | Any valid queries names | Determines the queries available for selection from the Search function. |

A complete list of preferences can be found in the *Preferences and Environment Variables Reference*.

Preference scope

When working with preferences, you must take each preference's scope into consideration. The scope of a preference determines both user access to the preference and precedence of its behavior when there are multiple incidences of the same preference. Preference scope is displayed in the **Preference details** view of the **Options** dialog box. When an existing preference is selected, its scope displays to the right of the **Name** box.



It may be required to have different values for the same preferences for different groups, roles or even users. To accommodate this requirement, Teamcenter allows for preferences to be set to handle each of these situations. The following list is shown in order of the precedence hierarchy used to resolve setting conflicts in multiple locations, with 1 being the highest.

1. User preference
2. Role preference
3. Group preference
4. Site preference
5. All

Preference precedence

Preferences are stored as XML strings in specific tables in the database. Preference scope determines in which table a preference is stored. Data from the site table is loaded during server startup and data from the other table is loaded at user login. Only data from the current logon credentials are loaded. Preference scope also determines the order in which the system searches through the tables for the preference and its values.

The following list defines how the preferences are searched from the Teamcenter database:

User

When the scope is **User**, the system searches through preference data in the following order: user, role, group, overlay, site.

Role

When the scope is **Role**, the system searches through preference data in the following order: role, group, overlay, site.

If the user changes group or role settings during a session, the appropriate group and role preferences are loaded.

Group

When the scope is **Group**, the system searches through the preference data in the following order: group, overlay, site.

If the user changes group or role settings during a session, the appropriate group and role preferences are loaded.

Site

When the scope is **Site**, the system searches through the preference data in the following order: overlay, site.

All

When the scope is **ALL**, the system searches through the preference data in the following order: environment variables, user, role, group, overlay, site.

Activity

In this activity you modify and create preferences with different scope.

1. Modify a site preference

In this activity, as a member of the DBA group, you modify an existing site preference.

2. Create a user preference

In this activity, as a user, you create a user preference that takes precedence over the site preference.

3. Create a group preference

In this activity, you use a group administrator account to create a group preference.

Review questions

1. What are preferences?

Select all that apply.

- Special environment variables stored in the Teamcenter database
- Special workspace objects
- XML strings in specific tables in the database

2. Preferences allow you to configure many aspects of a session such as user logon names and basic system behavior such as business rules and security access.

- True
- False

3. What menu do you select to modify preferences?

Select one answer.

- **Edit® User Setting**
- **View® Organization**
- **Edit® Options**

4. What does the **AE_dataset_default_keep_limit** preference do?

Select one answer.

- Determines if you keep (true) or not keep (false) old dataset versions in the database
 - Sets the default BOM view
 - Sets the default number of preferences for datasets
 - Sets the number of old dataset versions to store in database
5. Categories may contain subcategories to allow easier location and identification of a specific preference.
- True
 - False
6. What precedence hierarchy is followed for preferences when the scope is **User**?

Select one answer.

- **Site, Group, Role, User**
- **Group, Role, User, Site**
- **User, Group, Role, Site**
- **User, Role, Group, Site**

Generating preference reports

You can generate reports that list the changes made to site preferences at your site and reports that list all the logged-on user's group, role, and user preferences in the database. The following reports can be created:

- The **Site** preference report lists the difference between the off-the-shelf (COTS) preferences delivered in the **tc_preferences.xml** file and any new preferences added to the database. This allows you to determine what custom preferences have been created at your site.
- The **Group** preference report lists all preferences stored for your current logged-on group.
- The **Role** preference report lists all preferences stored for your current logged-on role.
- The **User** preference report lists all preferences stored under your current user ID.

Create preference reports

Generate reports that list the user, group, role, or customized site preferences in your database. If desired, you can export the report to a file.

1. Click **Report**  in the **Organization** tab of the **Options** dialog box.

The **Report** dialog box appears.

2. Select the scope for which you want a report: **User**, **Role**, **Group**, or **Site**.

The **Report Changes** dialog box lists all the preferences of the selected scope which have been changed, providing the original and current values. Use the forward and back buttons at the top of the dialog box to view the various preferences.

3. (Optional). To export the report, type a path and file name in the **Export File Name** box (or browse to an existing file) and click **Export**.

The preference report is exported to the specified file in XML format.

Activity

In this activity, you create and export modified preference reports for the **Site** and the **Group** scopes.

1. Create a modified **Site** preference report

In this activity, as a member of the DBA group, you create a site preference report.

2. Create a modified **Group** preference report

In this activity, as a member of the standards group, you create a group preference report.

Review questions

1. What information is obtained from a preference report?

Select all that apply.

- Changes made to site preferences at your site
- Organization passwords
- Custom preferences
- Access control rules

2. What is required to create a **Group** preference report?

Select one answer.

- To log on to the system as a user of the group
- To log on as a DBA administrator of the site
- To log on to the system as the group administrator

3. How many types of preference reports can be created?

Select one answer.

- **Site** preference reports only
- **Site** and **User** preference reports
- **Site**, **Group**, **Role**, and **User** preference reports
- **Site**, **Group**, and **User** preference reports

4. You can export a preference report in XML file format.

- True
- False

Import and export preferences

Preferences can be imported and exported in two ways.

- Using the **Import** and **Export** links in the **Options** dialog box.
- Using the **preferences_manager** utility.

Key points

- The **Import** and **Export** links on the **Options** dialog box allow you to import the preferences and their values in to the database according to the specified handling options and export the scope-based preferences to the specified XML file.
- Import and export of preferences can be done based on a category in addition to the scope.

Export preferences from the database to an XML file

1. Choose **Edit® Options**.
2. In the **Options** dialog box, click the **Index** or **Search** link.
3. Click the **Export** link at the bottom of the **Preference details** pane.
4. Type a name for the export file in the **Name** box or click the button to the right of the box and locate a file in your directory structure.
5. Select the scope of the preferences to be exported. You can export **Site**, **Group**, **Role**, or **User** preferences.
6. Optionally, select the **Open on Export** box to open the XML file when the preference export operation is complete.
7. Click **Export**. Teamcenter exports the scope-based preferences to the specified XML file.

Import preferences into the database

1. Choose **Edit® Options**.
2. Click the **Index** or **Search** link at the bottom of the dialog box.
3. Click the **Import** link at the bottom of the **Preference details** pane.
4. Type the name of the XML input file in the **Import File Name** box, or click the button to the right of the box and locate the file in your directory structure.
5. Select the scope of the preferences to be imported. For example, if you choose the **User** option, the imported preferences are stored as preferences for the logged in user.
6. Select one of the following import modes:

Automatic

Imports all preferences in the specified file.

Selective

Compares the values of the preferences in the XML file with the preference values in the database for the specified scope and displays the differences in values in a separate dialog box. You can browse through the preferences and modify values before importing the preferences in to the database.

7. Specify one of the following options for handling duplicate preferences encountered during the import operation:
 - Override preference values in the database with values in the XML file
 - Merge preference values in the database with values in the XML file
 - Skip the preference
8. Click **Import**.

The preferences and their values are imported into the database according to the specified handling options.

preferences_manager utility

You can create a custom preferences XML file to create or modify preferences, and then use the **preferences_manager** utility to:

- Import preferences to the database
- Export preferences from the database
- Migrate the legacy **upfiles**, **rpfiles**, and **gpfiles** preference files to the database

The **preferences_manager** utility is location in **TC_ROOT/bin** directory.

You can use the following command examples:

Table 14-2. Preferences utility examples

| Command line | Results |
|--|--|
| <code>preferences_manager -h</code> | Displays the help information on the console. |
| <code>preferences_manager -mode=generatexml -h</code> | Displays the advanced options that can be used with -mode=generatexml option. |
| <code>preferences_manager -mode=category -h</code> | Displays the advanced options that can be used with -mode=category option. |
| <code>preferences_manager -u=userID -p=password -g=dba -mode=generatexml -file=%Tc_DATA%\tc_env -scope=SITE -context=Teamcenter -out_file=C:\temp\site_pref.xml</code> | Generates the site preferences XML file from the legacy site preference file. |
| <code>preferences_manager -u=infodba -p=password -g=dba -mode=import -scope=SITE -file=c:\temp\site_pref.xml -categories=general</code> | Imports the site preferences in an XML in the specify category name. Categories are a comma separated list. |
| <code>preferences_manager -u=infodba -p=password -g=dba -mode=import -scope=SITE -file=c:\temp\site_pref.xml -action=SKIP</code> | Imports the site preferences in an XML file and skip the processing for all the preferences in the XML file that exist in the database. |
| <code>preferences_manager -u=infodba -p=password -g=dba -mode=import -scope=SITE -file=c:\temp\site_pref.xml -action=MERGE</code> | Imports the site preferences in an XML file and merge the values for the preference in the database with the values for the same preference in the XML file. |

Table 14-2. Preferences utility examples

| Command line | Results |
|--|---|
| <pre>preferences_manager -u=infodba -p=password -g=dba -mode=import -preference=TestPreference -scope=SITE -values=Val1,Val2,Val3 -action=OVERRIDE</pre> | Imports a preference and override the values in the database with the values specified for the preference on the command line. |
| <pre>preferences_manager -u=userID -p=password -g=groupName -mode=export -scope=GROUP -out_file=c:\temp\group_pref.xml</pre> | Exports all group preferences in the database for the user's default group. |
| <pre>preferences_manager -u=userID -p=password -g=groupName -mode=export -scope=USER -out_file=c:\temp\user_pref.xml</pre> | Exports all user preferences in the database for a user when the utility is run as that specific user. |
| <pre>preferences_manager -u=userID -p=password -g=dba -mode=migrate -dir=c:\TC_data \rpfiles\Designer.iman_env</pre> | <p>Migrates preferences specified by the -dir option. The -dir option is required. This example migrates the legacy Designer role preferences file. If the directory does not contain the upfiles, rpfiles, and gpfiles folders, you can set the following environment variables:</p> <p>TC_GROUP_PFILE= location of gpfiles TC_ROLE_PFILE= location of rpfiles TC_USER_PFILE= location of upfiles</p> |

For information, see *Configuration utilities* in the *Utilities Reference*.

Optional activity

- Import and export categorized preferences

In this activity, you create custom categories and preferences to test the import and export functionality and the **preferences_manager** utility.

Review questions

1. You only can import and export preferences using the **preference_manager** utility.
 - True
 - False
2. What is a the **Selective** import mode?
 - A mode that imports all preferences in the specified file
 - A mode that selects the scope of the preferences to be imported
 - A mode that compares the values of the preferences in the XML file with the preference values in the database for the specified scope
3. What is the **preference_manager** utility used for?
Select all that apply.
 - To import and export preferences to and from the database
 - To create categories and subcategories
 - To migrate legacy preference files to the database

Summary

Topics learned in this lesson:

1. Define preferences and their scope.
2. Identify key Teamcenter preferences.
3. Modify and create preferences.
4. Import and export preferences.

Lesson

15 Report Builder definitions

Purpose

The purpose of this lesson is to create and manage data reports using Report Builder.

Objectives

After you complete this lesson, you should be able to:

- Create specific report definitions.
- Manage report definitions.
- Deploy report definitions throughout the enterprise.

Help topics

Additional information for this lesson can be found in:

- [*Report Builder Guide*](#)

Report Builder interface

Report Builder is a Teamcenter rich client application and uses standard menus, buttons, tabs, and symbols.

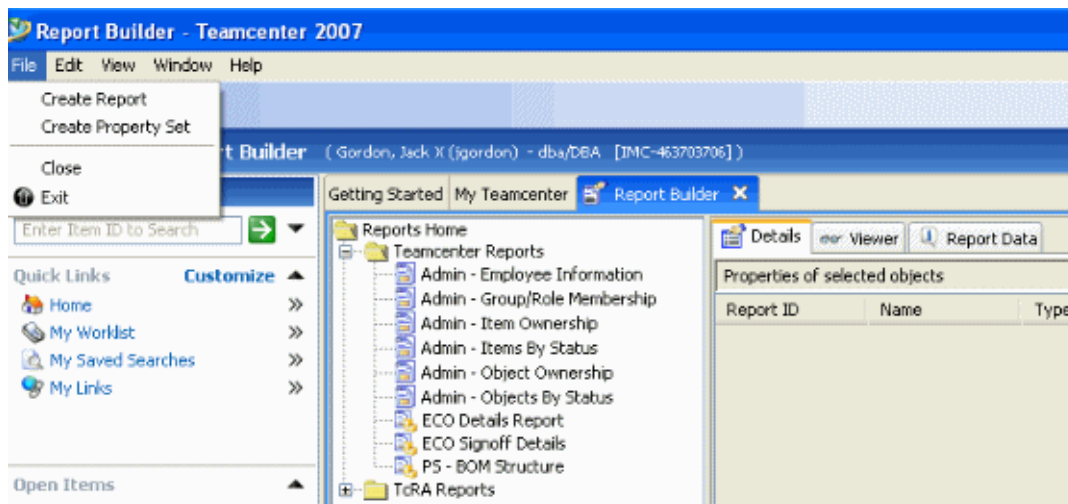
The **Reports Home** folder displays the report definitions available in Teamcenter. The **Reports Home** has two subfolders:

- **Teamcenter Reports**

Contains report definitions included or created in Teamcenter.

- **TcRA Reports**

Contains report definitions created with Teamcenter Reporting and Analytics (TcRA).



Report Builder definition types

Use Report Builder to create report specification templates. These templates are called **report definitions**. A report definition contains the name, description, query sources, property sets, and style sheets.

Note

Only Teamcenter administrators can create report definitions.

Report definitions are categorized as these types:

- **Summary reports**

Summary reports are generated from Teamcenter saved queries. When you select a summary report from a list of available summary reports, you are prompted to input query criteria. You can leave default values or enter new values. If default values are not given, you are prompted to type the values when you generate a report.

- **Item reports**

Item reports are executed in the context of an individual object, such as an item revision. Each item report object is associated with a Teamcenter class and transfer mode object. Transfer mode objects are created in the PLM XML Export Import Administration application.

The difference between item and summary reports is that item reports require a context and summary reports are context free. For example, when you select an item type document, only the reports applicable to the document item and its relationships are available to generate the report. Reports that guarantee a context for the selected object are called *item reports*.

- **Custom reports**

Custom reports address special cases such as complex processing or calculations done through custom code or when data is coming from external sources. Each custom report object is associated with a custom program. When a custom report is selected from a list, the server launches the program and the custom process.

Report definition structure

The process of creating report definitions varies slightly based on the type of report definition you choose to create.

To create a summary report definition, you must identify:

- A unique report ID that can be assigned automatically by the system.
- A report name and description.
- A saved query source specification.

And to create an item report definition, you must identify:

- A unique report ID that can be assigned automatically by the system.
- A report name and description.
- A particular class in the data model.
- A PLM XML transfer mode object.

To further complete the report definition, you add the following information:

- A property set for PLM XML for import or export operation
- A style sheet dataset to reformat the report data for output

Note

Basically, a query source and transfer mode with closure rules are sufficient to define the report fit data. An optional property set defines the properties that appear in the report output.

Basically, report and query source are sufficient to define a report definition. (Saved query or transfer modes closure rules define the report fit data). An optional property set defines the properties that appear in the report output. By default, Report Builder produces reports in native PLM XML format. The end user has the option to attach different style sheets to produce different types of reports in HTML or Microsoft Excel format. Refer to the Report Builder sample style sheets for examples of how to format summary and item reports. The sample stylesheets are available on the corporate server in the *tc_data/crf/Resources* directory.

Standard Teamcenter report definitions

The report definitions shown in the following table are delivered with the standard Teamcenter installation.

| Report definition | Description |
|------------------------------------|---|
| Admin-Employee Information | Returns employee information for a specified person/user. |
| Admin-Group/Role Membership | Returns membership information for a specified group/role. |
| Admin-Item Ownership | Returns items of a specified type that are owned by a specified user/group. |
| Admin-Items by Status | Returns items of a specified type released to a specified status. |
| Admin-Object Ownership | Returns objects of a specified type that are owned by a specified user/group. |
| Admin-Objects by Status | Returns objects of a specified type released to a specified status. |
| ECO Details Report | Returns change object information. |
| ECO Signoff Details | Returns change object signoff information. |
| PS - BOM Structure | Returns BOM information for specified item revisions. |

Activity

- Create a summary report definition.

In this activity, you create a summary report using the **CCC_Item** query. The **CCC_Item** query finds all the CCC items with a particular part number, customer name, and whether the part has been shipped or not. You also customize an existing style sheet for HTML output.

Review questions

1. What are report definitions?

Select one answer.

- PLM XML import or export operations
- Report specification templates
- Saved queries

2. Summary reports are generated from saved queries.

- True
- False

3. What type of reports requires an object context?

Select one answer.

- Custom reports
- Item reports
- Summary reports

4. What do you need to create an item report definition?

Select one answer.

- A custom application
- A property set and a saved query
- A report ID, a report name, a class, and a transfer mode

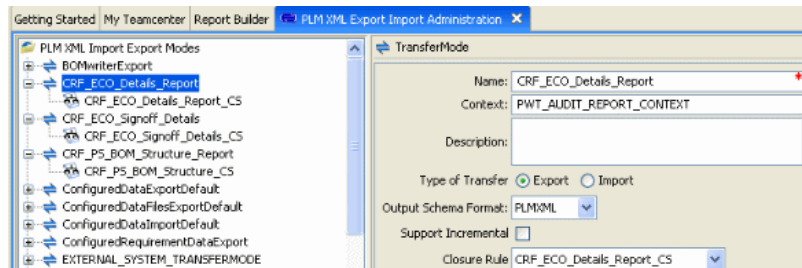
5. By default, Report Builder produces reports in HTML format.

- True
- False

PLM XML report data

The Report Builder application leverages on the PLM XML schema to produce report data. You can use PLM XML Export Import Administration to:

- Create transfer mode objects to define the report export context.
- Create closure rules to define the scope of the data translation.
- Define a list of property clauses or property set that extends the Teamcenter data model needed in the report in PLM XML format.



Teamcenter data model quick review

- Classes are the persistent representations of the data model schema and provide attributes to business objects.

Classes are also known as the persistent object model (POM).

- Business objects are the fundamental objects used to model business data.

Note

Business objects are also known as types.

- Attributes are item characteristics, such as name, number, description, and so on. Attributes are attached to classes, and business objects inherit the attributes from classes as properties.
- Properties are essentially attributes that are inherited by business objects.
- Properties contain information such as name, number, description, and so on. In addition to the properties that are derived from the persistent storage class, business objects can also have additional properties such as run-time properties, compound properties, and relation properties.
- Form business object are the only business objects to which you can directly add properties.

For more information, see the *Business Modeler IDE Guide*.

Creating transfer mode objects

Transfer mode objects allow you to import and export objects or system data. They contain the rules that configure the export operation. You can edit, remove or change the rules, property sets, configuration objects, and actions that define a transfer mode.

To create a transfer mode for a Report Builder definition:

1. Open the PLM XML Export Import Administration.
2. Select **TransferMode** in the lower left pane of PLM XML Export Import Administration.
3. Type a unique name and description.
4. Type the context string that maps the transfer mode object to the customized processor for the report type.
5. Select the **Export** in the **Type of Transfer** option.
6. Select a closure rule.

Note

If necessary, you can create a closure rule before creating the transfer mode object.

7. Select a property set.

Note

If necessary, you can define a property set before creating the transfer mode object.

8. Click **Create**.

For more information, see the *PLM XML Export Import Administration Guide*.

Creating closure rules

Closure rules specify how to process the data structure to obtain the report information. Closure rules are defined as a set of rules that are evaluated in order for each target object.

Closure rules are conformed of five elements:

- Primary object selector
- Secondary object selector
- Relation selector
- Action
- Optional conditional clause

To create a closure rule for a Report Builder definition:

1. Open PLM XML Export Import Administration.
2. Select **ClosureRule** in the lower left pane of PLM XML Export Import Administration.
3. Type a unique name and description.
4. Select **Export** in the **Scope of Transversal** option.
5. Click the **Add clause** button to create the ordered clauses that specify how the data structure is traversed.
6. Click **Create**.

For more information, see the *PLM XML Export Import Administration Guide*.

Defining property sets

Property sets are collections of Teamcenter data, such as class attributes or business object (type) properties. You can create a property set that contains just the information you want to use in a report. For example, you can create a property set that lists the name, ID, and description of items.

When you choose **File→Create Property Set**, the New Property Set wizard launches, which contains the same functionality as the **Property Set** tab in PLM XML Export Import Administration application.

Example

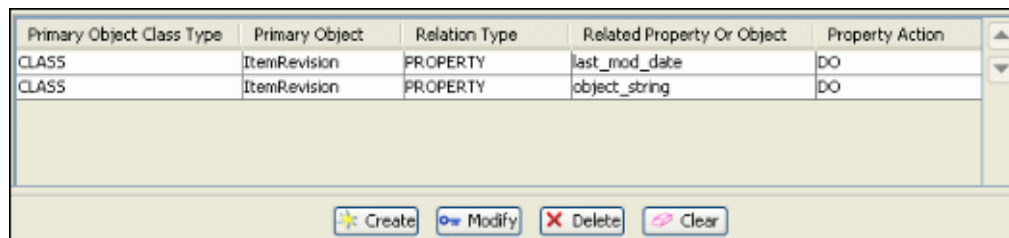
You define a property set using the following syntax:

- `CLASS.ItemRevision:PROPERTY.last_mod_date:DO`
- `CLASS.ItemRevision:PROPERTY.object_string:DO`

The data transformed in the PLM XML file looks like this:

```
<ProductRevision id="id12" name="assy" accessRefs="id5" masterRef="#id20"
revision"A">
  <UserData id="id15">
    <UserData value="02-Dec-2003 12:37" title="last_mod_date"/>
    <UserData value="000338/A-assy" title="object_string"/>
  </UserData>
</ProductRevision>
```

In the **New Property Set** dialog box, you can define property set clauses without writing code.



- A property set adds the Teamcenter custom data into a PLM XML file that may not exist in the PLM XML schema.
- The property sets controls the **UserData** element.
- The **UserData** element is a container for a list of name-values pairs that allows you to add any attribute or property of a Teamcenter object to the PLM XML operations.

To create a property set in the Report Builder:

1. Choose **File**→**Create Property Set**.

The system displays the New Property Set wizard.

2. In the **Property Set Name** box, type a name for the property set.
3. In the **Description** box, type a description of the property set.
4. In **Scope of Property Set** section, select **Export** because you want the data to be exported from the database and placed into reports.
5. To locate the class or business object (type) whose properties you want to compile into a set:

- a. In the **Search** box, select **Class** or **Type**.

- b. Click the **?** button and type the name of the class or business object (type) whose properties you want to compile into a set, for example, **ItemRevision**.

- c. Click the magnifying glass button or press Enter.

The class or business object appears in the pane. To close the selection box, click the **X** in the upper right corner.

The properties of the class or business object appear in the pane.

- d. To change how the attributes display in the pane, click the **Display Settings** button. You can see all attributes or just the attributes for the class.

- e. Select the properties you want to place into the set by double-clicking them in the pane.

The properties appear at the bottom of the dialog box.

For example, if you searched for the **ItemRevision** class, double-click the **ItemRevision** properties in the pane.

6. To change the order that the properties are placed into the set, click the up or down arrows to the right of the bottom pane.

To remove a property, click the minus button (–) to the right of the bottom pane.

7. You can also add properties manually by clicking the plus button (+) to the right of the bottom pane (also known as the clause table):

- a. Click the **+** button located to the right of the clause table.

- b. Click the arrow in the **Primary Object Class Type** cell and choose **CLASS** or **TYPE**.
- c. Type the name of the class or business object in the **Primary Object** box. For example, type **ItemRevision**.
- d. Click the arrow in the **Relation Type** cell and select either **PROPERTY** or **ATTRIBUTE**.
- e. Type the name of a property in the **Related Property Or Object** box.
- f. Click the arrow in the **Property Action Type** cell and choose one of the following:
 - **DO**
Specifies that the action be executed.
 - **SKIP**
Specifies that the relationship should not be followed. This directive is useful to eliminate special cases before a more general rule is reached.

Note

The **SKIP** action cannot be combined with any other action.

8. Click the **Create** button.

The system displays the new rule in the left pane. This property set can now be used when creating an item or summary report definition.

Key points

- **Primary Object Class Type** specifies the primary object class or type to be translated. The class or type name is preceded by the **CLASS**, **TYPE**, or ***** keyword and a period.

Caution

Use caution when using the wildcard ***** because it indicates that all classes types fit the primary or secondary object criteria.

Example

```
CLASS.Item.  
TYPE.Item.  
TYPE.*.
```

- The **Relation Type** options are **ATTRIBUTE** and **PROPERTY**.

Both options refer to the name of the attribute or property of the primary object that refers to the secondary object.

Example

```
CLASS.Item:ATTRIBUTE.revision_number  
TYPE.Item:PROPERTY.type_name  
CLASS.Item:PROPERTY.item_id
```

- The **Property Action Type** options are **DO** and **SKIP**.

DO specifies that the action be executed.

SKIP specifies that the relationship should not be followed. This directive is useful to eliminate special cases before a more general rule is reached. The **SKIP** action cannot be combined with any other action.

Example

This clause gets all the occurrence notes from a BOM line except manufacturing notes:

```
TYPE.BOMLine:TYPE.MfgNote.PROPERTY.bl_all_notes:SKIP  
TYPE.BOMLine:TYPE.*.PROPERTY.bl_all_notes:DO
```

For more information and examples, see the *PLM XML Export Import Administration Guide*.

Activity

- Create an item report definition.

In this activity, you create an item report definition that reports the items and item revisions found in a folder object.

1. Define closure rules and create a transfer mode.
2. Create an item report definition and test the PLM XML output.
3. Attach a style sheet and validate the report HTML output.

Review questions

1. The Report Builder application leverages on the PLM XML schema to produce report data.

Select one answer.

- True
- False

2. What is the purpose of a transfer mode object?

Select all that apply.

- To contain the rules that configure the export operation
- To import and export Teamcenter objects or system data
- To save dynamic queries

3. To specify how to process the data structure to obtain the information, you create _____ that are defined as a set of rules that are evaluated in order for each target object.

Fill in the blank.

- closure rules
- property sets
- transfer mode objects

4. What do you control with property sets?

Select one answer.

- Query sources
- Transfer mode objects

- **UserData** elements

5. What do you specify with the **SKIP** action?

Select one answer.

- That the action is executed
- That the first action is not executed, but then can be processed
- That the relationship should not be followed

import_export_reports utility

The **import_export_reports** utility allows report definitions and their associated style sheets to be exported from one Teamcenter server and imported to another.

Use the Report Builder **import_export_reports** command line utility to export and import report definitions in files including saved queries and associated style sheets.

- To export a report definition from a Teamcenter server, run the following command on the command line:

```
Import_export_reports -export -u=username  
-p=password -g=group  
-stageDir=data-directory -reportId=report-name
```

This command exports report definitions and associated data (style sheets) to the directory specified by the **stageDir** argument. You must specify the full path with the **stageDir** argument, and the directory must exist before running the utility. After you run the utility, the directory contains a subdirectory containing an XML file that matches the **reportId** argument.

- To import a report definition from a Teamcenter server, run the following command on the command line:

```
Import_export_reports -import -u=username  
-p=password -g=group  
-stageDir=data-directory -reportId=report-name
```

When you import a report definition, the **stageDir** argument specifies the path, and the **reportId** argument specifies the directory name where the report resides.

Activity

- Export a report definition.

In this activity, you use the **import_export_reports** utility to export a report definition.

- Import a report definition.

In this activity, you use the **import_export_reports** utility to import a report definition.

Review questions

1. What do you use to deploy report definitions to other Teamcenter sites?

Select one answer.

- The **import_export_reports** utility
- The **File® Export** menu in Report Builder
- The PLM XML Export Import Administration window

2. When you export and import report definitions, the saved queries and associated style sheets are also processed.

- True
- False

Summary

Topics learned in this lesson:

1. Define types of reports.
2. Create summary and item report definitions.
3. Define PLM XML report data.
4. Import and export report definitions.

Lesson

16 Access Manager

Purpose

The purpose of this lesson is to establish unique data access requirements.

Objectives

After you complete this lesson, you should be able to:

- Configure the rule tree.
- Import and export the Access Manager rule tree.

Help topics

Additional information for this lesson can be found in:

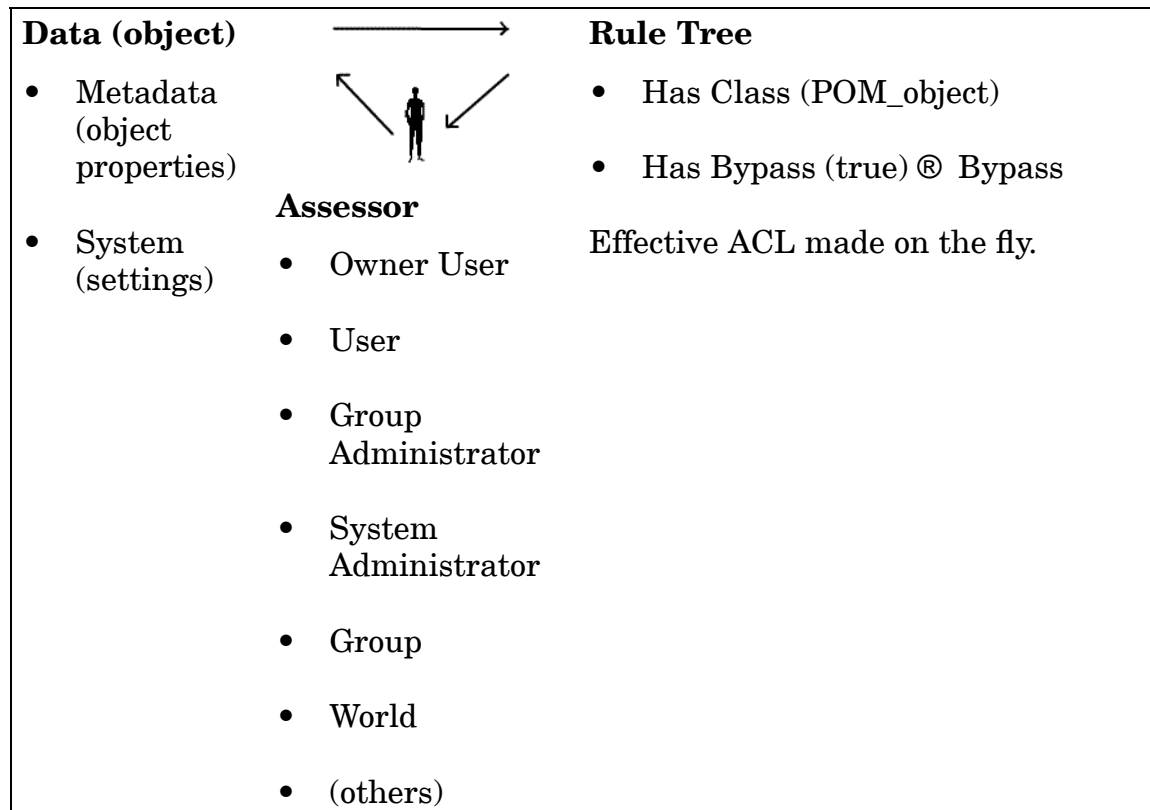
- [*Access Manager Guide*](#)
- [*Security Administration Guide*](#)

Protecting Teamcenter data

Object protection and ownership are extremely important in a distributed computing environment. Objects represent actual product information in the database and must be protected from unauthorized or accidental access, modification, and deletion. Teamcenter implements two different tiers of data protection:

- Rules-based protection
- Object-based protection

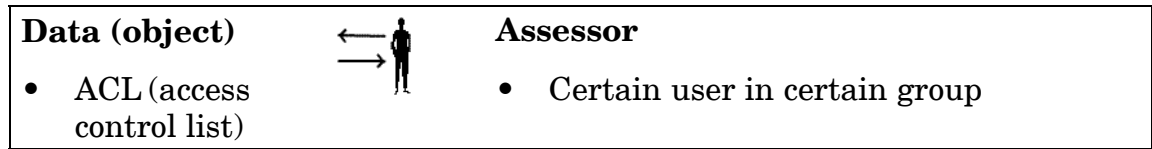
Rules-based protection



As an administrator, you define various conditions or rules that control who can or cannot access various objects. These rules are global affecting your entire Teamcenter site and are defined using Access Manager.

As a user interacts with Teamcenter data, access is determined by using the user's accessor information, the object's metadata (properties), and some system information. This information is tested in the conditional rule tree, which returns the access results.


























Object based protection



Any Teamcenter user having change access to objects can set object-based protection access control lists (ACLs) on objects to create exceptions to rules-based protection. Object ACLs are most useful when it is required to either grant wider access or limit access to a specific object. With object-based protections, settings are added to and controlled by the object to implement case-by-case exceptions to the access defined by the Access Manager rule tree.

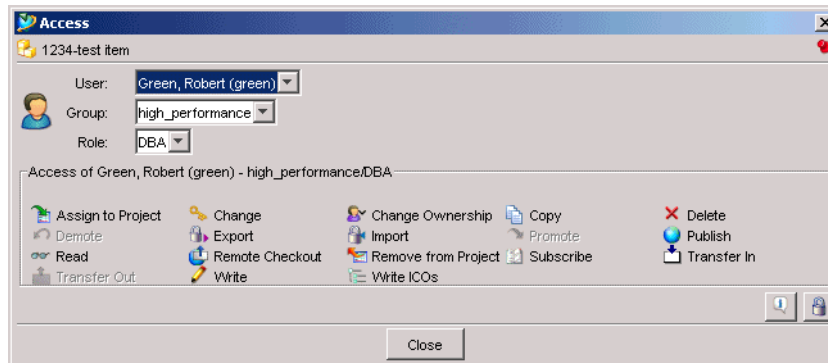
Object ACLs

Access control lists display the current protections for an object and allow the user to change those protections. Initial protections are determined by the rule tree.



















|  |  |  |  |  |  |  |  |  |  |  |
|---|---|---|---|---|---|--|---|---|---|---|
| System Administrator | | |  |  | | | |  | |  |
| World |  |  |  |  |  |  |  |  |  |  |

Access Manager interface

The **Access** dialog box shows permission for the selected object.



Symbol Description

| | |
|---|--|
|  | Read: open and view an object. |
|  | Write: modify an object. |
|  | Delete: delete an object. |
|  | Change: modify object protections. |
|  | Promote: Move a task ahead in a workflow. |
|  | Demote: move a task back in a workflow. |
|  | Copy: copy an object. |
|  | Change Ownership: grant change, or restrict ownership rights on objects. |
|  | Publish: grant or restrict publish privileges to users or groups. |
|  | Subscribe: subscribe to an event on a specified workspace object. |
|  | Export: export objects from your database. |
|  | Import: import objects into your database. |
|  | Transfer Out: transfer ownership of objects exported from your database. |
|  | Transfer In: become the new owner of objects imported into your database. |
|  | Write Classification ICO: write classification ICOs. |
|  | Assign to Project: Assign objects for which you have privileges, to a project. |
|  | Remove from Project: Remove objects, for which you have privileges, from a project. |
|  | Remote Checkout: Remotely check out the object. |

Access Manager rule tree

Rules are organized in the Access Manager rule tree and are evaluated based on their placement within the tree structure. The default rule tree included in your Teamcenter installation assumes that users are granted access unless access is explicitly denied. The rules are evaluated from the top of the tree to the bottom of the tree, with rules at the top of the tree taking precedence over rules at the bottom of the tree. The rule tree acts as a filter that an object passes through when a user attempts to access the object. When conditions that apply to the selected object are met, the privileges defined in the access control list are applied.

Note

Subbranches always take precedence over parent branches in the tree.

The **Access Manager** application displays the rule tree.

- Has Class(POM_object)
 - Has Bypass(true) -> Bypass
- Has Class(POM_object) -> System Objects
- Has Class(WorkspaceObject)
 - In Job(true)
 - Has Status(TCM Released) -> TCM Released Rule
 - Has Status() -> Vault
 - Has Object ACL(true)
 - Has Class(POM_application_object) -> Import/Export
 - In Project() -> Projects
 - Owning Group Has Security(Internal) -> Internal Data
 - Owning Group Has Security(External) -> External Data
- Has Class(POM_application_object) -> Working

Note

For a list of default rule conditions, see the *Access Manager Guide*.

How rules work

The simplified view of the default rule tree (below) is used in the example that follows.

- 🔑 Has Class(POM_object)
 - 🔑 Has Bypass(true) → Bypass
 - 🔑 Has Status() → Vault
 - 🔑 Has Class(POM_application_object) → Import/Export

A user, Jim Smith (**jsmith**), attempts to open the released **MyDataset** text dataset. To perform this action, Jim Smith needs read privileges on the dataset.

The rule tree is evaluated, as follows:

- The **Has Bypass(true) → Bypass** rule is evaluated. This is a high-level rule that grants system administration privileges to users.

If Jim Smith has bypass set, the **Bypass** ACL is applied.

Jim is not a system administrator, nor does he have bypass set; therefore, the system concludes that this rule does not apply and moves down the tree to the next branch.

- The **Has Status() → Vault** rule is evaluated. This rule asks if the object has a status type attached to it. If yes, the **Vault** ACL is applied.

The **MyDataset** dataset is released; therefore, there is a status type attached to the dataset. The system concludes that the rule is applicable and applies the **Vault** named ACL.

The **Vault** ACL explicitly grants read privileges to the **World** accessor. The **World** accessor represents all users.

- The **Has Class(POM_application_object) → Import/Export** rule is evaluated. This rule asks if the object is of the **POM_application_object** class. If yes, the **Import/Export** ACL is applied to the object. All workspace objects, including datasets, are subclasses of the **POM_application_object** class; therefore, the system concludes that the rule applies and the **Import/Export** named ACL is applied to the object.

The ACL is evaluated to determine if Jim Smith can open the dataset. If the **Import/Export** ACL explicitly grants read privileges, Jim can open the dataset. If the ACL explicitly denies read privileges, access to the dataset is denied.

Add an Access Manager rule

1. Select the parent tree rule to which the new node will be added.
2. Set the **Condition**, **Value**, and **ACL Name** for the new rule.

Note

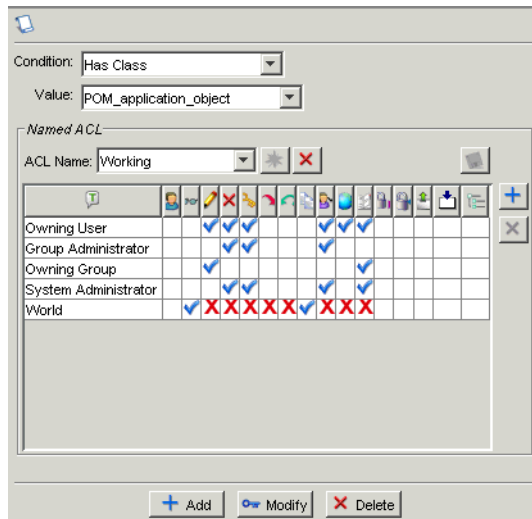
ACLs can be referenced in more than one rule.

3. Click **Add**.
4. Click **Save**.

This creates the new rule and adds it to the selected parent in the rule tree. An asterisk appears next to the Access Manager name indicating that the application has been modified.

Create and manage named ACLs

The right pane of the Access Manager application displays the condition and value being evaluated with the named ACL to be applied.



There are numerous predefined named ACLs with the default Access Manager setup. By using the list to choose a different named ACL (for example, **ImportExport**), you can view/edit the other predefined named ACLs.

Named ACLs are managed by accessor. Accessors are listed in the first column of the **Named ACL** table. An accessor is a user or group of users who share certain traits, such as membership in the group that owns the object or membership in the project team.

| Accessor | Description |
|-----------------------------|--|
| Owning User | Users who initially created an object. Ownership can be transferred and additional privileges (for example, delete) are usually granted to an object's owner that are not granted to other users. |
| Group Administrator | User who has special maintenance privileges for the group. |
| Owning Group | Group that owns the object. Usually, it is the group of the user creating the object. Additional privileges (for example, write) may be granted to the owning group, because it is common for users to share data with other members of their group. |
| System Administrator | Users who are members of the system administration group. |
| World | Any user, regardless of group or role. |

See the complete list of accessor types and precedence in the *Access Manager Guide*.

Build an effective ACL

The effective ACL is the cumulative buildup of all the named ACLs appropriate to that object within the rule tree.

1. Remove from the rule tree all lines with rules that do not apply to the object.

The effective ACL is now built up from those named ACLs left in the trimmed tree.

2. Order the traversal of lines (rules).
3. Build up the effective ACL by combining the named ACLs on the lines (in order of traversal).

ACL precedence

| | |
|----|------------------------------|
| 1 | Condition {Value}@ Named ACL |
| 2 | Condition {Value}@ Named ACL |
| 15 | Condition {Value}@ Named ACL |
| 9 | Condition {Value}@ Named ACL |
| 3 | Condition {Value}@ Named ACL |
| 4 | Condition {Value}@ Named ACL |
| 7 | Condition {Value}@ Named ACL |
| 5 | Condition {Value}@ Named ACL |
| 6 | Condition {Value}@ Named ACL |
| 8 | Condition {Value}@ Named ACL |
| 14 | Condition {Value}@ Named ACL |
| 10 | Condition {Value}@ Named ACL |
| 13 | Condition {Value}@ Named ACL |
| 11 | Condition {Value}@ Named ACL |
| 12 | Condition {Value}@ Named ACL |

When determining the order of precedence in the rule tree, follow this simple guideline:

- Children take precedence over parents.

Evaluating which privileges are finally granted depends on two criteria:

- Precedence of the conditional statements in the rule tree
- Accessor precedence in the named ACL

The result is an effective ACL, which controls the access to the object.

ACL guidelines

- Do not modify named ACLs referenced by rules on or within the **System Objects** branch. These rules are used for internal purposes and changes could result in unpredictable behavior or loss of data.
- Do not delete or change the order of the branches in the upper area of the rule tree. This could result in unpredictable behavior or loss of data.
- Add new rules for working data below the **Working** branch. This is the proper location to add new rules for working data types at your site.
- When adding a new rule, if you do not enter a named ACL, you create an inactive stub that can be further customized by adding additional rules as subbranches. Until you add additional subbranches or modify this rule to apply a named ACL, it has no effect on object protections.
- Do not modify a COTS rule, but create a new one in the tree to replace it.
- Export the rule tree before and after making changes.

Activity

1. View default protections.















In this activity, you create a new item and view the default Access Manager defined privileges.

2. Modify a named ACL.

In this activity, you define object specific access control list data to remove write access from a single user.

Review questions

1. Determine the order of precedence.

-   Condition {Value}@ Named ACL
-   Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL
-   Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL
-   Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL
-  Condition {Value}@ Named ACL

Group security

There are many facets to group security, but here we focus on internal and external group security. It is important to understand how to configure security to limit access by suppliers or other partners outside of your company.









| Accessor | Description |
|-----------------------------|--|
| Owning Group | Group that owns the object. Usually, it is the group of the user creating the object. Additional privileges (for example, write) may be granted to the owning group, because it is common for users to share data with other members of their group. |
| Groups with Security | Users who have the given security value, either Internal or External . This value is used to distinguish between groups in the parent company (internal) and suppliers (external). |

Configuring security to prevent suppliers from viewing internal data

In this example, ABC Part Company wants to grant users in their internal groups, Design, Development, and Manufacturing, read access to data owned by any internal group while denying access to Supplier 1 and Supplier 2. Use the following rule that specifies privileges for all data owned by users who are members of internal groups:

Owning Group Has Security(Internal) -> internal_group_acl

The **internal_group_acl** ACL controls read access and also allows the owning user to modify access privileges (define object ACLs for exceptions to the rule-based security).

|  |  |  |  |
|---|---|--|---|
| Owning Group | |  |  |
| Groups with Security | Internal |  | |
| Groups with Security | External |  | |









Configuring security for data owned by a supplier (external data)

In this example, ABC Part Company wants to define access privileges for data owned by one of their suppliers, Supplier 1. The group at Supplier 1 (external group) that owns the data needs to have read and write access to their data. Users at ABC Part Company (internal groups) need read access to the data. Users belonging to any other supplier group must be denied access to the data owned by Supplier 1.

Use the following rule that defines privileges for all data owned by users who are members of internal groups:

Owning Group Has Security(“External”) -> external_group_acl

The **external_group_acl** ACL controls read and write privileges for the owning group and grants read privileges to internal groups. External groups are denied read privileges.

|  |  |  |  |
|---|---|---|---|
| Owning Group | |  |  |
| Groups with Security | Internal |  | |
| Groups with Security | External |  | |

Import and export the access manager rule tree

The Access Manager rule tree can be exported as an ASCII file to a directory outside of Teamcenter and, conversely, be imported back into the Teamcenter environment.

The export and import feature can be used for the purpose of archiving or safeguarding a rule tree prior to making extensive modifications (such as, adding or deleting branches, relocating branches, and so on). If, after many modifications, it is found that the original rule tree is desired, it can be reinstated.

This capability is also useful for copying the Access Manager rule definitions from one Teamcenter site to another.

Rule tree edits from the Access Manager application are automatically saved in the exported file format when the user clicks the **Save** button.

Import and export guidelines

- Never modify rule tree files with a text editor.

Although rule tree files are simple ASCII files that can be read using any text editor, you must never modify them with a text editor. These files must conform to a particular format and can be easily corrupted.

- To successfully load a new rule tree from a different Teamcenter site, your site must have the same types, roles, and groups as those referenced in the rule tree file.

If there is any incompatibility, Teamcenter immediately terminates the read operation (at the first discrepancy) and displays an error message. If you encounter schema compatibility problems when loading a rule tree from a file, open that rule tree file with a text editor. Either print the file or make note of the types, roles, and groups referenced in that file. Then, define these exact types, roles, and groups for your site.

Activity

1. Export the Access Manager rule tree.

In this activity, you export Access Manager rules before you make a change in the rule tree.

2. Add an Access Manager rule to the rule tree.

In this activity, you add an access manager rule to the rule tree to give members of the standards group write access to items.

3. Import the Access Manager rule tree.

In this activity, you import the original Access Manager rules.

Review questions

1. Internal groups with security represent what?

Select one.

- Company group
- Owner group
- Supplier group

2. External groups with security represent what?

Select one.

- Company group
- Owner group
- Supplier group

3. When is the best time to export the rule tree?

Select all that apply.

- After making changes
- Never
- Prior to making changes

Summary

Topics learned in this lesson:

1. The rule tree controls access to system data.
2. Named ACLs uniquely define object access.
3. Group security protects data from being accessed by outside suppliers.
4. Export rules to archive or import to another site.
5. Imported rules must conform to the site's organization.

Lesson

17 *Projects to control access*

Purpose

The purpose of this lesson is to define project-level security.

Objectives

After you complete this lesson, you should be able to:

- Define projects.
- Create projects.
- Assign objects to projects.

Help topics

Additional information for this lesson can be found in:

- [*Project Guide*](#)
- [*My Teamcenter Guide*](#) (see *Working with projects*)
- [*Business Modeler IDE Guide*](#) (see *Working with internal extensions*)
- [*Security Administration Guide*](#) (see *Configuring project-level security*)
- [*Authorization Guide*](#)

Define projects

The *Project* application allows Teamcenter sites to control access to a Teamcenter database by multiple organizations, such as:

- Project teams
- Development teams
- Suppliers
- Customers

Key points

- A *project* is the basis for identifying a group of objects made accessible to users at a supplier's site for a particular piece of work (the project).

Who can create projects

Projects are defined by project administrators in the Project application in the rich client.

While Project is an administrative application, the specific group/role of **Project Administration/Project Administrator** is required to create and modify projects and project team members.

Key points

- Although Project does not need to be enabled before you use it, there are some tasks your Teamcenter administrator may choose to perform to configure the way Project works at your site.
- A Teamcenter administrator must populate the group/role of **Project Administration/Project Administrator** in the Organization application before users intended as project administrators can log on to the rich client with this group/role and create projects.

Project naming rules

Naming rules can be assigned to the project ID and project name boxes, enforcing consistent naming conventions at your site.

To put a naming rule into effect, you must attach it to a property of a business object. For example, if you want to use a naming rule to define how all projects are named, attach the naming rule to the **project_id** property of the **TC_Project** business object.

You might create a naming rule like **CCC_Project_&&&&**. Here, you want every project ID to start with **CCC_Project_**. The user can add their own alphanumeric four-digit code at the end. There are many ways to define a standard project ID.

Define team members for projects

Project team members are typically external to the organization running the Teamcenter site.

Team members for a particular project can include:

- Project team administrator
- Privileged team members
- Regular team members

Key points

- After projects are created, privileged team members must assign Teamcenter data to the project using My Teamcenter in the rich client and the project functionality buttons in the thin client.

Apply project security

Access is controlled on a project-by-project basis.

Project-level security functionality provides an additional level of security while allowing non-site employees access to objects in the Teamcenter database.

Key points

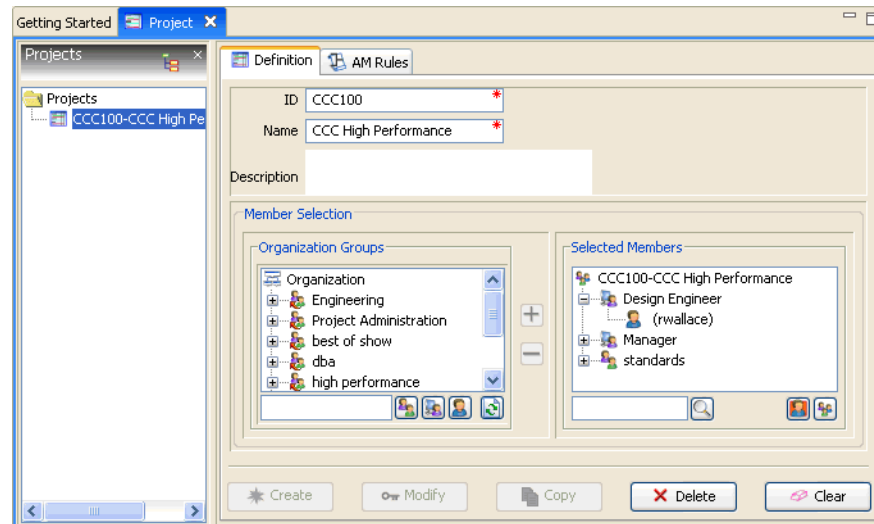
- Default security rules for project-level access are automatically defined when a project is created.
- Additional security rules can be configured in Project by a project administrator or in Access Manager by a Teamcenter administrator.

Creating projects

Project creation and administration is done in the following order:

1. A project administrator is added to the **Project Administration** group by the Teamcenter administrator.
2. A project is created with specific groups, users or roles assigned as team members or privileged team members.
3. As database objects are created, they can be assigned to the project automatically or manually by privileged team members.

Project interface









Project administration tabs

| Tab | Description |
|-------------------|---|
| Definition | Displays the Project Definition pane. Use this pane to create the project, to assign project team members, and to designate privileged team members. |
| AM Rules | Displays the Access Manager Rules pane. Use this pane to apply access rules to a project. |

Note

Project administrators only have access to the **In Project** branch of the rule tree. You cannot modify other branches of the rule tree from within Project. In addition, moving the **In Project** branch to a different position in the tree requires Teamcenter administrative privileges and must be done using the Access Manager application.

Project administration buttons



| Button | Purpose |
|--|---|
|  Find groups | Searches for a group in the organization tree when a name or partial name and wildcard characters are entered in the text box. |
|  Find roles | Searches for a role in the organization tree when a role or partial role and wildcard characters are entered in the text box. |
|  Find users | Searches for a user in the organization tree when a user name or partial user name and wildcard characters are entered in the text box. |
|  Refresh tree | Refreshes the organization tree. |
|  Select privileged team members | Opens the Select Privileged Team Members dialog box where you can designate privileged team members. |
|  Select a team administrator | Opens the Select a Team Administrator dialog box where you can designate a team member to be a team administrator. |

Project quick links

There are two links in the **Quick Links** area of the navigation pane that are specific to Project:

| | |
|------------------------------------|---|
| Project Administration | Provides access to the project administration interface used to create and activate projects, assign users membership in projects, and designate project administrators and privileged team members. |
| Smart Folder Administration | Provides access to the smart folder filter configuration interface used to define filtering criteria based on the smart folder hierarchy. These filters control how project data is displayed to users. |

These links appear in both the **Project administration** window and the **Smart folder configuration** window.


| Button | | Purpose |
|---|---------------|--|
|  | Add filter | Adds a new row to the filter table that is used to define additional filters for data-driven folders. Additional filtering criteria can only be applied to data-driven folders. Abstract folders in the hierarchy can only apply a single filter criteria. |
|  | Remove filter | Removes the selected row from either of the filter tables. |

Project terms and concepts

The following table describes the objects, terms, and concepts that you use to create and manage project members and objects.

| Concept | Definition |
|----------------------------|---|
| Project | Projects organize data and are the basis for controlling access (typically read access) by project members. Only privileged team members can assign data to projects. |
| Supplier | Specifies a group of users external to the organization that owns the data. |
| Relation propagation rules | Defines how secondary objects are assigned to projects based on the defined relation to the primary object. |
| Project administrator | Teamcenter user with privileges to administer projects, including creating, modifying, and deleting projects and adding and removing team members. These privileges apply to the project metatdata, not to the data assigned to projects. |
| Project team administrator | Project team member with privileges to modify project information and to add and remove team members in the projects in which they are members. These privileges apply to the project metatdata, not to the data assigned to projects. There can be only one project team administrator per project. |
| Privileged team members | Project team members with privileges to assign or remove objects from their projects. |
| Team members | Users with read privileges to objects in a project. |

Create a project

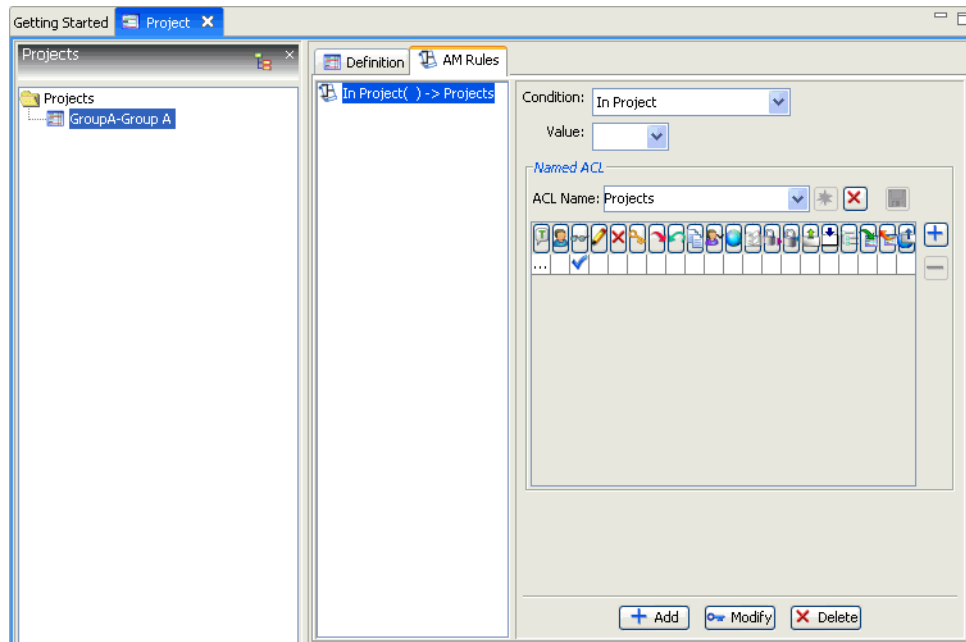
1. Open the Project  application.
2. Define the project name and ID.
3. Select the project team members.
4. Add additional security rules through the **AM Rules** pane.
5. Click **Create** to create the project.

Key points

- Entire groups, individual users, and roles can be added as team members of a project.
- Users are first assigned as team members, with a project team administrator and privileged team members being selected from the list of users assigned to the project.
- A user with the group/role of **Project Administration/Project Administrator** is automatically added as the team administrator.

Project security rule tree

The **AM Rules** pane provides security data for the project.



- **Condition**

Determines users privileges in the project. By default, users assigned as team members are in the project and have read access to objects in the project.

- **Value**

Specifies the project that these rules apply. By default, this rule is applied to all projects, but you can make it specific to your project.

- **Named ACL**

Sets up the form to create a new named ACL.

- **ACL Name**

Specifies the named ACL to use for this AM rule. By default, the project named ACL is used, but you can select the blank field or another named ACL to use.

- **ACL settings**

Specifies the detail ACL for this named ACL. By default, it is set to the named ACL selected. If you create a new named ACL, you specify the settings.

- Use the **Add**, **Modify**, and **Delete** buttons to update the AM rules for this project.

Manage projects

You can modify an existing project when you want to retain the majority of the project settings. Typically, you retain the same project ID and project name, modifying team member assignments or the project's security rules.

1. Select an existing project from the **Project** tree.
2. Modify any of the project settings, such as team member assignments or security rules.
3. Click **Modify**. Your modifications to the project information are saved to the database.

Assign data to projects

After the project administrator creates a project and assigns its team members, the project data (such as items, item revisions, datasets, and forms) must be assigned to the project.

These database objects can be assigned by:

- Project administrator
- Project team administrator
- Privileged team members

These members can also assign folders, or folders and their contents, to projects.

Key points

- Additional project security can be achieved using the **ASSIGN_TO_PROJECT** and **REMOVE_FROM_PROJECT** privileges.
- Consider that privileged members are able to assign objects to a project as long as they have read privilege to the object.

Example

Suppose a project member is a privileged member of two projects; Project A is more sensitive than Project B. This privileged project member is able to assign objects in the sensitive Project A to the less-sensitive Project B, thus giving other Project B members read access to the data in Project A.

This capability to assign objects to a project can be restricted using the **ASSIGN_TO_PROJECT** privilege. The exact behavior of these privileges is controlled by the **TC_project_validate_conditions** site preference.

Automatically assign new objects to a project

Teamcenter can be configured so that certain object types are automatically assigned to a project when they are created.

Example

When a new item revision is created, it is automatically assigned to the project on which the user is currently working.

Teamcenter administrators define which objects can be automatically assigned to projects using the Business Modeler IDE to configure the **autoAssignToProject** extension located in the **Extension Definitions** folder under **Rules**.

The following object types can be added to this extension:

- Item and item revision subtypes, such as engineering changes, and so forth
- Forms
- Datasets

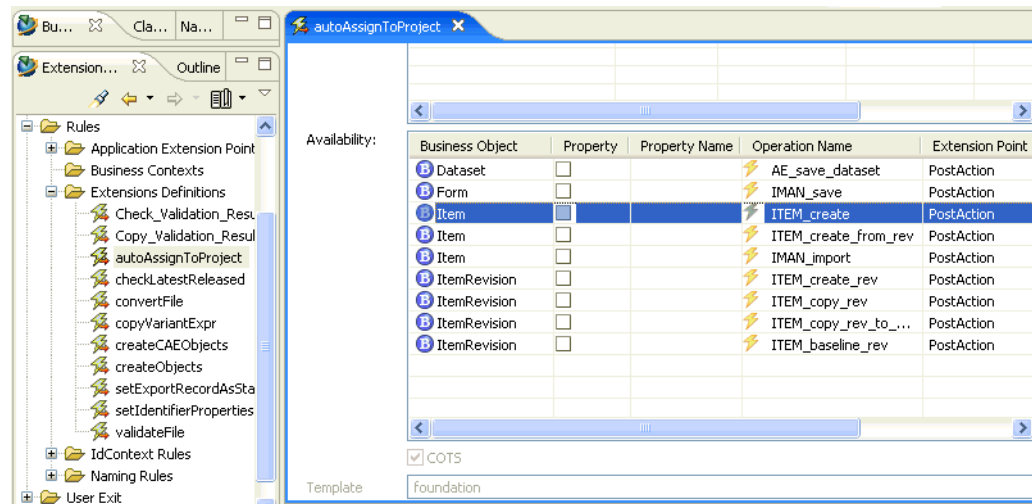
Key points

- Work contexts define a particular project as the current project. Defining a project as current enables the system to automatically assign new objects to the project. It also enables the system to enforce access if the **In Current Project** access rule is defined at your site.
- The current project is defined in the **User Settings** dialog box, accessed by choosing the **Edit→User Settings** menu command.

autoAssignToProject availability

From the **Business Modeler IDE** window, look up the availability of the **autoAssignToProject** extension on business objects.

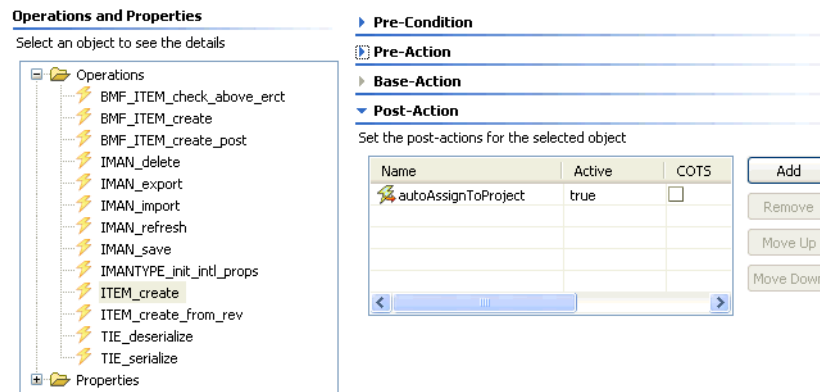
In the **Availability** table, view the business object, operation, and extension point for which the extension can be used. Decide which business object and operation for which you want to use the extension, and observe the extension point where it can be used.



There is a **PostAction** available when you create **Items** to automatically assign the item to the current project.

Configure automatic project assignment

1. Right-click the item and choose **Open Extension Rules**.
2. Select the operation for the post-action.
3. Add the **autoAssignToProject** post-action.



For any item created, the project property is automatically populated with the current project defined for the user.

Propagation rules

Configuring the **autoAssignToProject** extension for an object type has implications on the project *propagation rules*.

Project propagation rules determine which secondary objects are assigned to a project when a primary object type is assigned.

When there is a conflict between a propagation rule and the execution of the **autoAssignToProject** extension, the extension takes precedence.

Key points

Consider the following points when configuring the **autoAssignToProject** extension:

- The **autoAssignToProject** extension applies only to newly created objects; whereas, propagation of related objects to projects occurs whenever a relation between two objects is created, modified, or deleted.
- The **autoAssignToProject** extension explicitly assigns objects to projects; therefore, the objects can only be removed from the project by explicitly selecting the object and choosing the **Project→Remove** menu command in one of the Teamcenter applications.
- Propagation rules implicitly assign secondary objects to projects. Therefore, when the primary object is explicitly removed from the project, the secondary object is also removed from the project.
- The **update_project_bom** utility can be used to assign all items in a complete product structure to a project.

Activity

1. Configure automatic assign to project.

In this activity, you configure automatic assign to project for the **CCC_Car** business object.

2. Create a project with team members.

In this activity, you create the **CCC High Performance** project.

3. Test automatic project assignment.

In this activity, you create a new **CCC_Car** item and verify the **CCC High Performance** project is assigned.

Review questions

1. Who can create a project?

Select one.

- Privileged team members
- Project administrators
- Project team members

2. When **autoAssignToProject** extension is configured for an item, the user can select a project from a list upon creation.

- True
- False

Summary

Topics learned in this lesson:

1. A *project* is the basis for identifying a group of objects made accessible to users at a supplier's site for a particular piece of work (the project).
2. The **autoAssignToProject** extension allows the current project to be automatically assigned to newly created objects.

Lesson

18 Workflow process modeling

Purpose

The purpose of this lesson is to define Workflow process templates.

Objectives

After you complete this lesson, you should be able to:

- Define Workflow templates.
- Create Workflow process templates.
- Import and export process templates.
- Deploy Workflow templates to users and groups.

Help topics

Additional information for this lesson can be found in:

- [*Workflow Designer Guide*](#)

Basic concepts about Workflow Designer

Workflow stems from the concept that all work goes through one or more processes to accomplish an objective. Workflow is the automation of these business processes. Using Workflow, documents, information, and tasks are passed between participants during the completion of a particular process.

There are two types of templates that you can use to create a Workflow in Teamcenter: *process* templates and *task* templates.

Workflow process template

A *process template* is a blueprint of a process defined by placing Workflow and/or Change Management tasks, such as **Do**, **Perform Signoff**, **Route**, and **Checklist**, in the required order of performance. Additional process requirements, such as quorums and duration times, are defined in the template using Workflow handlers.

Workflow task template

A *task template* is a blueprint of a Workflow task. A task is a fundamental building block used to construct a process. Each task defines a set of actions, rules, and resources used to accomplish that task.

Workflow in Teamcenter

- Two applications are used to accomplish Workflow objectives:
 - **Workflow Designer**

A system administrator uses this application to design Workflow process templates that incorporate your company's business practices and procedures into process templates.
 - **Workflow Viewer**

Use this application to view the Workflow progress, even if you are not a participating member of that particular process.

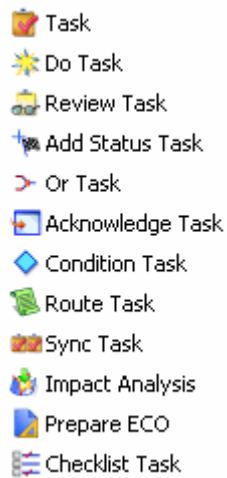
Tip

For ease of use, Siemens PLM Software recommends using My Teamcenter to initiate and complete workflow processes because the entire procedure can be accomplished from within your inbox in **My Worklist**.

Key points

- A process template is a blueprint of a Workflow process. It predefines a process that must be followed and defines a set of tasks.
- A *task* is a fundamental building block used to construct a process. Each task defines a set of actions, rules, and resources used to accomplish that task.

- Example tasks include **Task Do**, **Review**, **Add Status**, **Route**, and **Checklist**.



Warning

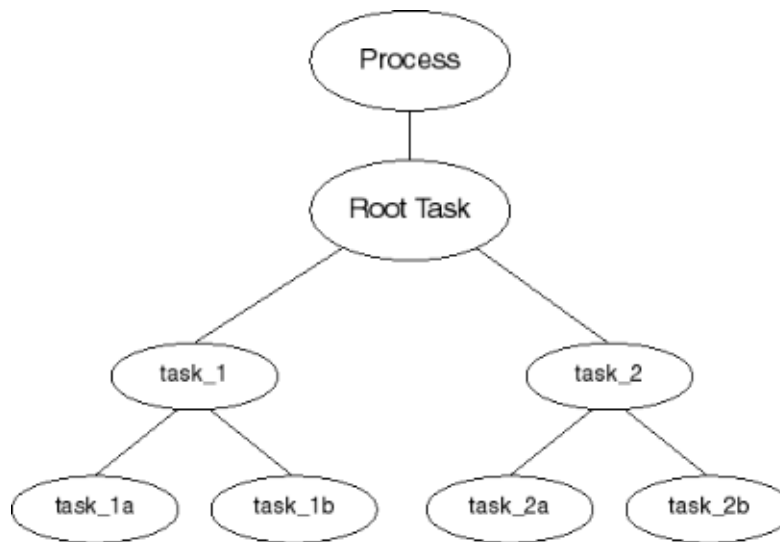
The **Checklist** task template is for use in change processes only. It cannot be used on a Workflow process.

- An *attachment* is an object associated with a process job.
 - **Target object**
Any object that requires a release status, for example an item revision or a dataset.
 - **Reference object**
An object that allows you to provide information to the signoff team.
- You can create, view, add tasks, modify, and link templates.
- Additional process requirements, such as quorums and duration times, are defined in the template using *workflow handlers*.
- *Handlers* are the lowest level building blocks in workflow.

Enterprise Process Modeling

Enterprise Process Modeling (EPM) is used to model Workflow processes, allocate resources, and manage data according to business rules. In other words, EPM is the software engine that Teamcenter uses to accomplish workflow objectives.

Each EPM process is a group of nested tasks. In fact, the process itself is a task. The top-level task of every process is referred to as the *root task*. The root task contains the process definition and the process name is the same as the root task name.



Key points

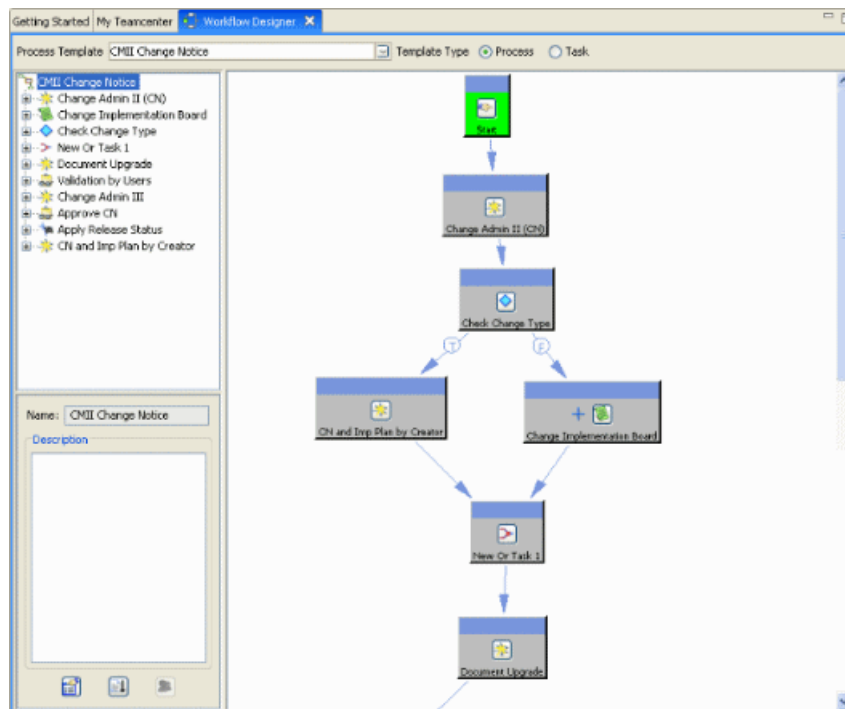
- In Enterprise Process Modeling (EPM), tasks have both temporal (time) and hierarchical (structure) relationships, which allows individual tasks to complete sequentially (serially) or asynchronously (in parallel).
- Essentially, each EPM process is a group of nested tasks. In fact, the process itself is a task. The top-level task of every process is referred to as the root task. The root task contains the process definition and the process name is the same as the root task name.
- A Workflow process contains defined tasks to automatically notify selected users requesting work signoff. The requests are tracked through an electronic worklist and each request maintains pointers to the data being approved. The exact task must be defined based on current company procedures for data signoff.
- A Workflow process can have any number of tasks arranged in a serial or parallel progression. At signoff, each review task has a list of users allocated for signoff.
- To modify the privileges on the data for users who need to work in the process, use Workflow access control lists (ACLs). You can create and attach different rules to the process tasks.


Note


To avoid modifying the database existing named ACLs, create new named ACLs for Workflow.

Workflow Designer interface

To start Workflow Designer, click **Workflow Designer**  in the navigation pane.



Browse mode is the default mode when you first access the Workflow Designer. Click **Browse**  to view process data and the details of the process. You cannot make any modifications in this mode.

Click **Edit Mode**  to edit templates. To use the Workflow Designer in edit mode, you must be a member of the system administration group.

Note

Access may be restricted even if you have administrator privileges.

Note

The **infodba** account is an administrative super-user account and should not be used for production work.

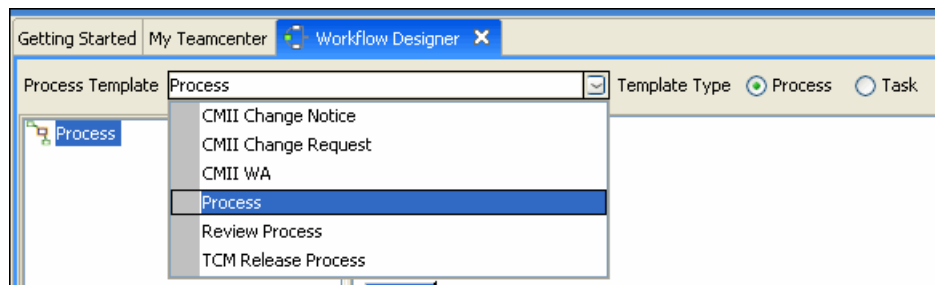
Workflow Designer interface key points

- **Process Template**

Lists either the process templates or the task templates depending on the **Template Type** setting.

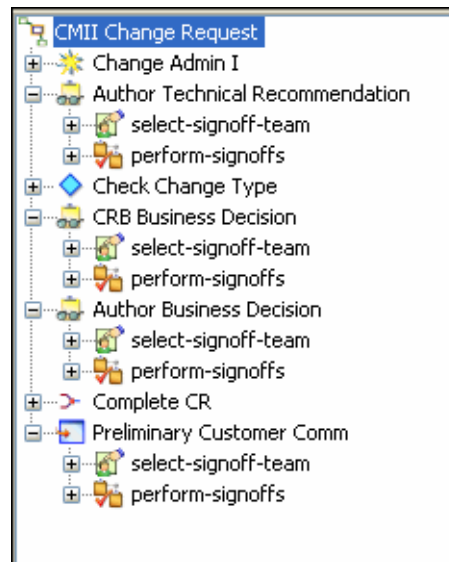
- **Template Type**

Toggles to show either process templates or task templates.



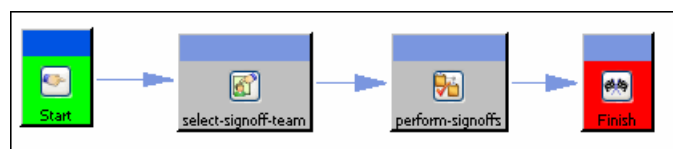
- **Template tree**

Lists the featured process template and its tasks.



- **Process flow**

Graphically shows the selected process template and its tasks.



Releasing data with a process

Most common user interaction data such as folders, forms, datasets, item revisions, and BOM view revisions can be initiated and released with a process.

When data is released, the following typically occurs:

- The release status and date released attributes get values.
- The released data becomes read only.
- Releasing flat objects, such as datasets or forms, only releases the object.
- Releasing a folder only releases the folder but not its contents.
- Releasing an item revision releases the item revision, BOM view revision, and all of the specification relation objects.

This is done through the use of a handler in the release process template. You can remove or change the handler to get other desired functionality.

Using quick-release process templates

Releasing data without electronic reviews or signoffs is sometimes necessary for the following reasons:

- When you import data into Teamcenter from existing systems where it was previously released.
- When you have data that has been reviewed outside of Teamcenter and is simply released in Teamcenter.
- When you are gradually phasing into the electronic workflow system.
- When you must release intermediate or baseline data for users to review or consume.

Note

An intermediate data release or *baseline* of in-progress data is sometimes called a preliminary data indicator (PDI).

To create a quick-release process, you can either create a process based on the **Review Process** template as it has the **create-status** and **add-status** handlers already built in, although this template also contains unnecessary handlers. Or you can create a process based on the **Empty Process** template and add the **create-status** and **add-status** handlers.

Batch release utility

Use the **release_man** utility to release objects in batch mode without creating jobs and audit files.

- You must belong to the DBA group to execute the **release_man** utility.
- The status type must be a valid status type defined for your site.
- The release folder must be a single folder directly inside the executing user's workspace **Home** folder.
- If the objects in the folder are item revisions, the contents of the item revision with the **IMAN_specification** relation and BOM view revision objects can also be released by using the **-spec** argument.
- The **release_man** utility does not release invalid objects or objects locked by other processes.
- The **release_man** utility can also be used to remove or delete the status from objects.

Examples

To apply the **Released** status type to all objects in the **my_folder** folder (including item revision specifications and BOM view revisions), type the following command:

```
TC_ROOT/bin/release_man -u=user-id -p=password -g=dba -spec  
-status=Released -folder=my_folder
```

To remove the **Released** status type from all objects in the **my_folder** folder (including item revision specifications and BOM view revisions), type the following command:

```
TC_ROOT/bin/release_man -u=user-id -p=password -g=dba -spec -unrelease  
-status=Released -folder=my_folder
```


Arguments

-spec

Indicates that specifications and BOM view revisions of an item revision in the release folder are released along with the item revision.

-unrelease

Removes the specified status type.

-retain_release_date

Specifies that if the object to be released is already released, the original release date is retained.

-status

Specifies the status type to be applied to all objects.

-folder

Specifies the name of the release folder.

-acl

Specifies the object protections to apply. Accessors are separated by commas. To apply the same object protections to more than one accessor, supply the appropriate colons. If the **-acl** argument is not supplied, the default protection is read-only for the executing user.

Activity

1. Create a quick-release process to release data.

In this activity, you release data using quick-release template.

2. Release data in batch mode.

In this activity, you release data using a batch process.

Review questions

1. When data is **released**, the following typically occurs:

Select all that apply.

- The release data is available for editing
- The release status and date attributes get values
- The released data become read only
- Users cannot access the released data

2. What is used to release data in a process template?

Select one answer.

- A handler
- An attribute
- An environment variable

3. A **quick-release** process is based on an **empty process** template.

- True
- False

4. What is **release_man**?

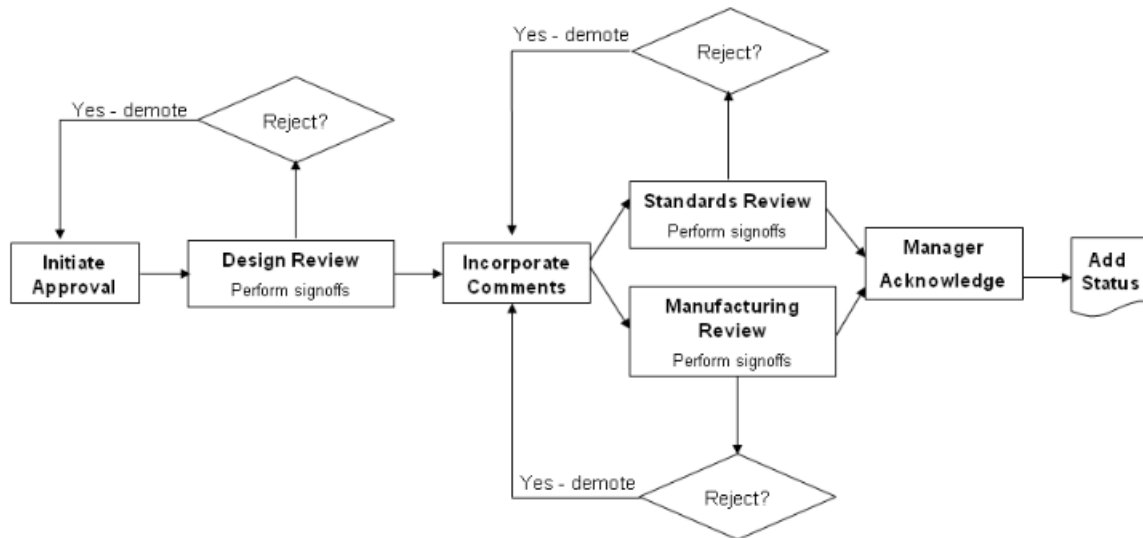
Select one answer.

- A release status
- A review task
- A workflow utility to release data in batch mode

5. You can use the **release_man** utility to add or delete a release status.
- True
 - False

Defining a process model

A *process* is made up of tasks. Some tasks may be subtasks associated with a main task.



Workflow processes pass documents, information, and tasks between participants during the completion of a particular process. A Workflow process can be large and complicated but can also be simple and straightforward.

Workflow process terms

The following table describes the objects, terms, and concepts that you use to create and manage workflow processes.

| Concept | Definition |
|-------------------|--|
| Process templates | Defines a Workflow process. A process defines a set of tasks and a user profile (group and role) for each task. The user profile is an abstract and does not correspond to a real person in a process. |
| Job | Defines an instance of a process. A new job is created each time a process is run. Jobs define the data required for that job and assigns real persons to perform tasks according to their user profile. |
| Process initiator | Defines a user who initiates a Workflow process. |
| Approver | Defines a user responsible for approving a task. |
| Attachment | Defines a Teamcenter object associated with a process job. There are two kinds of attachments: target objects and reference objects. |
| Target object | Defines any object (for example, item revision, dataset) that is released via a process job. Typically, target objects are assigned a release status when the process job completes. |
| Reference object | Defines an object attached to a process job that allows you to provide information to the signoff team. Reference objects are not released when the process job completes, but are kept in the job object for historical purposes. |
| Tasks | Defines actions used to perform within a job. When all tasks defined for a job are completed, the job is complete. |

Creating Workflow process templates

A *process* describes the individual tasks and the task sequence required to model the workflow process. In the process, you define a set of tasks and a user profile of groups and role for each task. *Templates* define a blueprint of a process or task to be performed at your site.

There are two different approaches to using Workflow in Teamcenter:

- As a review and approve mechanism

In this case, the user creates and completes the updates to any data and then starts a review process of the data.

- As a task controller for work in progress

As the work process progresses, the data is updated and added to the process to obtain a certain status.

Tasks are used to perform actions within a job. When all tasks defined for a job are completed, the job is complete.

Examples

- Create a process template outlining the process required for a final design review, name the process template final design review, and design the review process.
- Create a task template for implementing review edits of all design reviews.

Create a generic process template


1. Create a new root template by choosing **File® New Root Template**.
2. Give the template a unique name.
3. Define the template as a process.
4. Select an existing template or the empty template option on which to base the new template.
5. Click **OK** to create the template.
6. Configure the template by adding and linking tasks.

The process remains in **Under Construction**  state until it is made available. Before you exit Workflow Designer, you can set the template to available by adding it to the list of available processes or by selecting the **Set Stage to Available** option under the **Template** tree.

Note










Templates with the under construction designation are visible only to system administrators within Workflow Designer.




Key points

- You can create new a process template or a template based on an existing one.
- The system lists the template as **Under Construction**  until you select the **Set Stage to Available** option and exit Workflow Designer.

Workflow template tasks

This table lists the task templates available in Workflow Designer.

| Symbol | Task template | Definition |
|---|-------------------------|---|
|  | Task | Uses the EPMTask template to define custom forms and other site-specific tasks for the user to complete. This template is the default template setting. |
|  | Do Task | Uses the EPM-hold handler, which stops the task from automatically completing when started. |
|  | Review Task | Uses the select-signoff-team and perform-signoff subtasks, each of which has its own dialog box. |
|  | Add Status Task | Creates and adds a release status to the target objects of the process. It is a visual milestone in a process. No dialog box is associated with this type of task. |
|  | Or Task | Continues the workflow process when any <i>one</i> of its multiple task predecessors is completed or promoted. There is no limit to the number of predecessors an Or Task may have. |
|  | Acknowledge Task | Uses the Acknowledged and Not Acknowledged subtasks, each of which has its own dialog box. |
|  | Condition Task | Requires that the succeeding task contains a check-condition handler that accepts a Boolean value of either True or False . |
|  | Route Task | Uses the Review , Acknowledge , and Notify subtasks, each of which has its own dialog box. |
|  | Sync Task | Uses the check-process-completion handler to allow interprocess synchronization. A synchronization task template is dependent on the completion of one or more processes before it starts. |

| Symbol | Task template | Definition |
|---|-----------------------------|---|
|  | Impact Analysis Task | <p>Provides an impact analysis for a user to complete for the associated EC revision. The task provides Reference, Impact Analysis Form, Viewer and Task Info tabs.</p> <p>The Impact Analysis Task template is for use in EC processes only. It cannot be used on a Workflow process.</p> |
|  | Prepare ECO Task | <p>Provides EC Requests or EC Orders for a user to complete. The task provides ECO Sample and Task Info tabs.</p> <p>The Prepare ECO Task template is for use in EC processes only. It cannot be used on a Workflow process.</p> |
|  | Checklist Task | <p>Provides a checklist for a user to complete. The checklist form is a form type with a number of logical fields. You can create a custom form type with a site-specific field list using Java code to represent the form as a checklist. The task provides Check List and Task Info tabs.</p> <p>The Checklist Task template is for use in EC processes only: it cannot be used on a Workflow process.</p> |

Task and Do task templates

You use an empty **Task template** to define custom forms and other site-specific tasks for the user to complete. This template is the default template setting. The **Task template** is synonymous with the **EPMTask** template.

When this task is performed in a process, the **Task** dialog box displays the defined custom form for a user to complete.

Use the **Do task template** to define action tasks for a user to complete. The **Do task template** uses the **EPM-hold** handler to stop the task from automatically completing when started.

When this task is performed in a process, the **Do task** dialog box displays task instructions and a check box for a user to select, indicating that the task's instructions have been completed. When the check box is selected, the **Do task** dialog box sets the **EPM-hold** handler's argument to **False** and changes the status to **Complete**.

If you require user authentication before a **Do task** can be performed, add the **require-authentication** handler to the **Perform** action of the task. When you implement user authentication for this task, a password box displays below the **Comments** box. Users must type their user password in this box before they can click **Apply** and complete the task.

Review task template

You use the **Review task template** to define the **Signoff Team profiles** that a user complies with to assign review responsibilities to other users. This template also provides the **perform-signoff** task for the signoff team members to complete.

The **Review task template** uses the **select-signoff-team** and **perform-signoff** subtasks, each of which has their own dialog box. The **Select Signoff Team** dialog box displays **Group**, **Role** and **User** lists. The process initiator assigns specific users from these lists to become members of the **Review** task's signoff team.

When this task is performed in a process, the **Perform Signoff** dialog box displays three decision commands: **Approve**, **Reject**, and **No Decision**. Signoff team members can choose one of the commands to perform the signoff.

If you want to require user authentication before this task can be performed, add the require-authentication handler to the **Perform** action of the **perform-signoff** task. When you implement user authentication for this task, a password box displays below the **Comments** box. Users must type their user password in this box before they can click **Apply** and complete the task.

Acknowledge task template

You use the **Acknowledge task template** to define the signoff team profiles with which a user complies to assign acknowledgement responsibilities to other users. This template also provides the **perform-signoff** task for the signoff team members to complete.

When this task is performed in a process, the **Acknowledge** dialog box displays two decision commands: **Acknowledged** and **Not Acknowledged**. Signoff team members choose one of these commands to perform the signoff.

Tip

Define the signoff profiles by group or role, not by individual users. You can use the wildcard (*) to leave both the group and role category undesignated.

Resource Pool Subscription

A **Resource Pool** is a group, role in a group, or a role that can be assigned tasks the same way an individual user is assigned tasks. **Resource Pool** is useful in modeling workflow tasks that cannot be directly assigned to a single user, either automatically, or at run time by the responsible user. Instead, the task is assigned to a pool of users, one of whom perform in full the task or assign the task to themselves.

Note

To view and perform tasks that are assigned to resource pools, users must add the display of the **Resource Pool Inbox** to their **My Worklist** tab using the My Teamcenter **Tools® Resource Pool Subscription** menu.

Add Status task

You use the **Add Status task template** to create and add a **Release** status to the target objects of the process.

This template is a visual milestone in the Workflow process. There is no action for the user to perform, and therefore, no dialog box associated with the **Add Status** task.

Modifying task behavior

You can modify the behavior of a task within a process template by using:

- Attributes

You can set requirements and restrictions on a task. Task attributes are:

- Named ACL
- Template name
- Signoff quorum
- Release status

- Handlers

You use handlers to extend and customize tasks. The following is a list of the types of functions you can add to a task:

- Set protections
- Assign reviewers
- Demote a task
- Perform a signoff
- Change a status

Controlling access in a process template

Workflow ACLs are created in Workflow Designer within the context of a specific task and are considered an attribute of the task.

Key points about Workflow ACLs

- Access privileges to data that is the target in a workflow process are controlled using the **In Job** rule condition.

An ACL is not associated directly with the **In Job** condition rule. If the condition is evaluated as being true, the system applies the ACL associated with the current task in the workflow process.

- The system uses the **EPM-set-rule-based-protection** handler to determine the appropriate ACL to be applied.
- Accessors you can use to control access to the process data are:

- **Approver (RIG)**

Users who are members of a sign-off team in a workflow process with a specific role in a specific group (RIG). This accessor is only used in workflow ACLs and must match the signoff role-in-group requirements for the release level associated with the workflow ACL.

- **Approver (Role)**

Users in a specific role who are members of a sign-off team in a workflow process.

- **Approver (Group)**

Users in a specific group who are members of a sign-off team in a workflow process.

- **Approver**

Users who are members of a sign-off team in a workflow process regardless of their role and group.

- **Task Owner**

User who is granted privileges for the task's target data.

- **Task Owning Group**

Group that is granted privileges for the task's target data.

- **Responsible Party**

Users responsible for performing a particular task. This ensures that only the user assigned as the responsible party is given privileges to the task's target data.

- You can specify the **Promote** and **Demote** privileges for the Workflow process.

Adding task handlers

You can customize task behavior by creating and modifying task handlers. A task handler is a small ITK program or function. Handlers are the lowest level building blocks in EPM and are used to extend and customize tasks.

There are two kinds of handlers:

- Action handlers

Extend and customize task actions. Action handlers perform such actions as displaying information, retrieving the results of previous tasks (inherit), notifying users, setting object protections and launching applications.

- Rule handlers

Integrate workflow business rules into EPM processes at the task level. Rule handlers attach conditions to an action.

Syntax for handler arguments

In the rich client user interface, you define arguments and values using the **Handlers** dialog box. Click **Task Handlers** to open the dialog box.

Select a handler name from the **Handler** tree. Existing arguments and values for that handler populate the argument table. Enter additional data by typing argument and value data into the table cells. To assign multiple values to a single argument, separate the values with commas. For example:

| Argument | Values |
|-----------|--------------------|
| -relation | IMAN_specification |
| -type | UGMASTER, UGPART |
| att_type | target |

Example of useful handlers

Each task has a number of folder actions. You can add action handlers to perform such actions as displaying information, retrieving the results of previous tasks, notifying users, setting object protections and launching applications. Rule handlers integrate business rules into the EPM process at the task level.

Action handlers

- **add-status, create-status and set-status**

- The **create-status** handler creates a status object and attaches it to the root task.
- The **add-status** handler takes the status object that is attached to the root task and applies it to the target objects (same as the append mode for the **set-status** handler.)
- The **set-status** handler applies the appropriate status objects to the processes' target objects.

Note

You can use the **delete** argument with the **set-status** handler to remove status objects from targets that were applied in Workflow processes.

- **CR-fill-in-reviewers**

Automatically assigns signoff reviewers that meet specified user, group, or role criteria for the specified **Review** task. This criteria populates the signoff profiles.

- **require-authentication**

Displays a password box in the perform dialog box or pane of the task within which it has been placed.

- **demote and demote-on-reject**

The **demote** handler clears all signoff decisions from the current and previous **Review** tasks. An optional argument allows the user to specify the task name that the process is demoted to.

The **demote-on-reject** handler demotes the current task to the previous task or to the task specified on the **-level** argument of the demote handler placed on the **Undo** action of the current task.

- **notify**

Informs users of a task's status through e-mail.

- **EPM-export-to-plmxmlfile**

Exports targets, references, and/or process information to a PLM XML file. Use this handler to export targets and references data to a PLM XML file during a workflow process.

- **EPM-tessellation-handler**

Tessellates NX datasets. It identifies which datasets to tessellate by reading the targets set in the **EPM_tessellation_target_type** site preference and comparing them against the targets identified for the Workflow process.

Rule handlers

- **check-status**

Allows a process to be initiated only if the current and the previous revisions have a valid release status.

- **CR-assert-targets-checked-in**

Verifies that all target objects in this process are checked in.

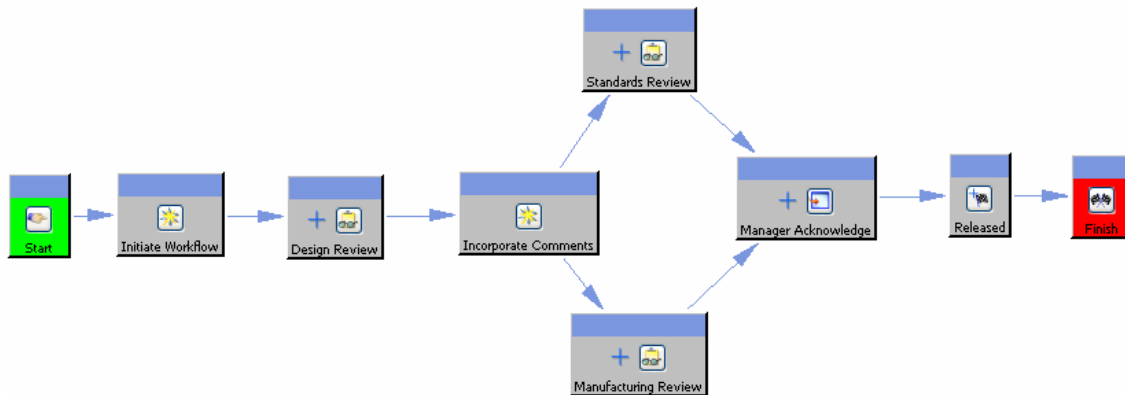
- **disallow-adding-targets** and **disallow-removing-targets**

The first handler disallows adding targets after a process is initiated. A switch can be used to specify the types of objects to be excluded. The **disallow-removing-targets** handler prevents targets from being removed from a process after the process is started.

Activity

- Create a multitask workflow process for release data.

In this activity, you create a new template with the required tasks to complete a review process.



CCC Product Release process template specifications

- The process signifies that the product is ready for production.
- The process targets **CCC_Item Revision**.
- Approvals:
 1. Seeks approval of Design Engineer from the high performance group.
 2. Seeks approval of Standards Engineer from the standards group.
 3. Seeks approval of Manufacturing Engineer from the high performance group resource pool.
 4. Notifies the Manager from the high performance group.
- Constraints:
 1. The target data is checked in.
 2. The reviewers are not allowed to add or remove data.
 3. The reviewers can demote the data to a certain task level.
- Adds **Released** status to the targeted data.

Review questions

1. What is **Workflow**?

Select one answer.

- A check list for signoffs
- Any object that is released via a process job
- Workflow is the automation of a business process

2. How many **types of workflows** are in Teamcenter?

Select one answer.

- **Do, Review, and Checklist** tasks
- Process and task templates
- Target and reference objects

3. To modify the user's privileges on the data, you add **Workflow ACLs**.

- True
- False

4. A **process** is made up of _____.

Select one answer.

- Attachments
- Conditions
- Tasks

5. What is a **task template**?

Select all that apply.

- A *task template* is a blueprint of a Workflow task
- An accessor to control access to the process
- An ITK program
- The fundamental building block used to construct a process

6. Where do you define a **signoff-team** profile?

Select one answer.











- In a **Do** task template
 - In a **Review** and **Acknowledge** tasks templates
 - In an **Or** task template
7. What is a **Resource Pool**?
- Select one answer.
- A group of users to perform **Acknowledge** tasks
 - A group, role in a group, or a role that can be assigned tasks the same way an individual user is assigned tasks
 - A task template
8. You can modify the behavior of a task within a process by using **attributes** and **handlers**.
- True
 - False
9. There are two kinds of handlers, condition and rule handlers.
- True
 - False
10. An example of an action handler is:
- Select one answer.
- **check-status**
 - **disallow-adding-targets**
 - **require-authentication**

Change Management process model

Change objects are items of type **EngChange** and are used to control all changes to a product's definition and configuration during its life cycle.

Change objects key points

- A change object is a special item type and is represented by a blue delta symbol.
- A change object is viewed in the Change Management Viewer.
- Contains default folders used to collect all the information about the change.

| Contents of:  CN0002/A;1-CN-WF0001 | | |
|---|----------------|---------------------|
| Object | Type | Relation |
|  Affected Items | | Collection |
|  Tasks to Track | | Collection |
|  CN0002/A | EngChange R... | ItemRevision Master |
|  Solution Items | | Collection |
|  Tasks to Perform | | Collection |
|  Reference Items | | Collection |
|  CN0002/A-CMII CN Form | CMII CN Form | Specifications |
|  Problem Items | | Collection |
|  Addressed By | | Collection |

Special workflow handlers are provided with Teamcenter for use in process models designed for the Change Management application.

- **ECM-add-affected-irs-as-target**

This handler attaches all item revisions in the **Affected Items** and **Solution Items** folders to the **Target** folder of the workflow process. This handler may be used several times in a process.

- **ECM-att-new-status-for-aff-revs**

This handler attaches a separate release status object for each affected item revision of the targeted change revision.

- **ECM-copy-end-item-effectivity**

This handler copies effectivity from one affected revision of the target change revision to any other affected revisions that do not share the same release status.

- **ECM-notify-competing-changes**

This handler notifies a site-defined list of recipients about competing changes for each affected revision.

- **ECM-start-new-sub-processes**

This handler starts a new process for all affected revisions of the targeted change revision.

- **EPM-attach-related-objects**

This handler attaches the specified related objects of the target objects as target/reference attachments to the process.

Activity

- Add special handlers to the **CCC Product Release** process template.

In this activity, you add **EPM-attached-related-objects** handlers to the tasks to attach the specified related objects of the target objects as target/reference attachments to the process.

Review questions

1. What is a change object?

Select all that apply.

- A handler to attach item revisions
- A special item type represented by a blue delta symbol
- A special workflow template
- An item of type **EngChange**

2. Inside the change object is a built-in folder structure for the change.

- True
- False

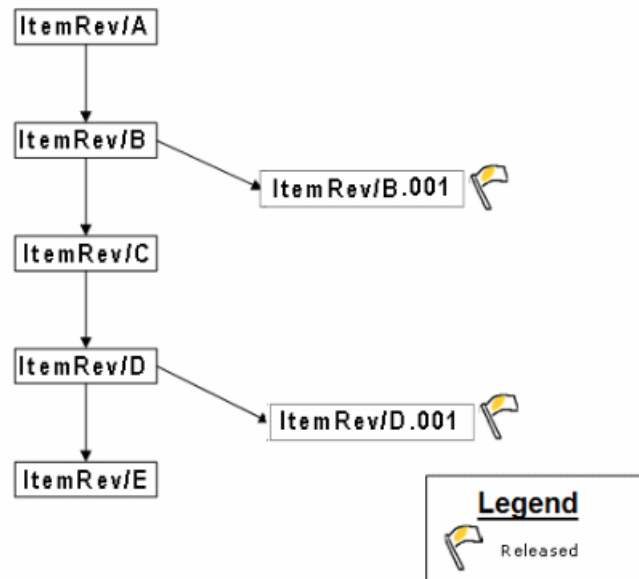
3. Why is a **EPM-attached-related-objects** handler added to a process template?

Select one answer.

- To attach a separate release status object for each affected item revision of the targeted change revision
- To attach the specified related objects of the target objects as target/reference attachments to the process
- To copy effectivity from one affected revision of the target change revision to any other affected revisions that do not share the same release status
- To start a new process for all affected revisions of the targeted change revision

Creating baseline process templates

The baseline feature allows you to create a baseline, or a snapshot, of a work-in-process item revision and its component objects without incrementing the revision of the item. This enables you to capture a product design at a particular stage without having to stop work or generate an undesired revision of the item.



Requirements for baseline process

To deploy a baseline release process, you must configure the following:

- A naming rule attached to the target item revision object
- A status option for baseline
- A quick-release process
- A value for the **Baseline_release_procedures** preference

Optional activity

- Create a baseline release process.

In this activity, you configure a new baseline process using a quick-release template.

- Create and attach a naming rule to the target item revision object using the Business Modeler IDE.
- Create a status for a baseline release using the Business Modeler IDE.
- Create a quick-release workflow procedure using the Workflow Designer.
- Adjust the environment in Teamcenter.
- Test the baseline process.

Review questions

1. What is a **baseline**?

Select all that apply.

- A review task
- A snapshot of a work-in-process data
- An engineering change object
- Product design at a particular stage

2. Creating a baseline process requires _____.

Select all that apply.

- A naming rule
- A quick-release process
- A status option
- An LOV object

Sharing process templates

You can export process and task templates from the Teamcenter database, storing the templates in a single export file you define in a directory you define.

This functionality is useful for transferring workflow templates between different Teamcenter sites. After exporting the templates to an export file, you can import the file into the Teamcenter database at another site.

Export a process template

1. Choose **Tools**→**Export**.

The **Export Workflow Templates** dialog box is displayed.

2. Type the path to the directory containing the objects you want to export in the **Export Directory** box, or click the **Browse** button to locate the directory.
3. Specify the name of the export file in the **File Name** box, for example, **template_export**.
4. In the **Templates** section of the dialog box, select the templates you want to export from the **All Templates** list. (Use the Ctrl key to select multiple templates).
5. Add the selected templates to the **Selected Templates** list. These are the templates the system exports.
6. If you want the system to continue the transfer if one or more templates fail to transfer, select the **Continue On Error** option. If one or more templates fail to transfer, the system records transfer errors in its log files, bypasses the failed templates, and transfers the remaining templates.

If you do not choose this option, the system stops the transfer process if one template fails to transfer and only includes in the transfer those templates that transferred successfully.
7. Click **OK** to export the templates in the **Selected Templates** list and close the dialog box.

The selected templates are exported to the file name you defined in step 3 in the directory you defined in step 2.

Import a process template

1. Choose **Tools**→**Import**.

The system displays the **Import Workflow Templates** dialog box.

2. Type the path to the directory containing the export file in the **Import File** box, or click the **Browse** button to locate the directory.
3. If you want the system to continue the transfer if one or more templates fail to transfer, select the **Continue On Error** option. If one or more templates fail to transfer, the system records transfer errors in its log files, bypasses the failed templates, and transfers the remaining templates.

If you do not select this option, the system stops the transfer process if one template fails to transfer and only includes in the transfer those templates that transferred successfully.

4. If you want the system to overwrite any template of the same name that already exists in the database, select the **Overwrite Duplicates** option. The system does not display or log any errors.

If you do not select this option, any importing template with the same name as an existing template is ignored and the import process continues. The system does not display or log any errors.

5. Click **OK** to import the templates contained within the file you selected into the Teamcenter database.

The imported template names now exist in the database and display in the **Process Template** list.

Activity

- Import process templates.

In this activity, you use the **Import** function to import Workflow processes.

Review questions

1. How do you transfer workflow between different Teamcenter sites?

Select one answer.

- Create a new template for each Teamcenter site
 - It is not a good practice to share Workflow templates
 - Use workflow import/export functionality
2. You can only **export** one template at the time.
 - True
 - False
 3. What is needed to overwrite an existing template during import?
 - Import existing templates into the system is not allowed
 - Nothing, because the system automatically overwrites the existing templates
 - Select the **Overwrite Duplicates** option

Adding Secure tasks

To secure a task you use the **require-authentication** action handler. You can add this handler to the **Perform** folder action of the **Do Task**, **Perform Signoff**, or **Condition Task**. A password box appears and the user must type the logon password to complete the task.

Use the **require-authentication** to secure the following task:

- **Do** tasks
- Perform-signoffs in **Review** tasks
- **Acknowledge** tasks
- **Manual Condition** tasks

Note

You must place the **require-authentication** on the **Perform** action folder of these tasks.

Activity

- Add security to process template tasks.

In this activity, you add security to the **Design Review** and **Manager Acknowledge** tasks of the **CCC Product Release** process template.

Review questions

1. How do you add security to a task?

Select one answer.

- Adding the **disallow-adding-targets** rule handler
- Adding the **require-authentication** action handler
- It is not possible to secure a task

2. Types of task that you can secure are _____.

Select one answer.

- **Automatic Condition** tasks
- **Review** and **Acknowledge** tasks
- **Or** tasks

3. To add security to the task, you must add the **require-authentication** handler to the **Start** action folder.

- True
- False

Adding Condition tasks

Use the **Condition Task** template to branch process flow according to defined criteria. **Condition** tasks have a result attribute that you can set to **True**, **False**, or **Unset**. The initial setting of the **Condition** task is **Unset** until it is either automatically or manually set to **True** or **False**. Successor tasks require the **Condition** task to be set to either true or false before they can start.

Dependency flow lines branch from the **Condition Task** template based on the true/false results of the **Condition** task, branching the workflow process template into one or more paths. Typically, one or more **Or** tasks are placed later in the process template to resolve the paths into a single path.

Use the **Condition Task** template to create a query that defines a true/false condition. Dependency flow lines branch out from the **Condition Task** template based on the true/false results of the query, branching the workflow process template into two or more parallel paths. Typically, one or more **Or** tasks are placed later in the process to resolve the parallel paths into a single path.

A **Condition** task can be configured to complete either automatically or manually. You need to determine which configuration is best suited for the process template you are defining. Typically, if a handler can determine the criteria, it is best to configure the task as automatic.

- **Automatic condition task**

Add an action handler that sets the task's result to true or false.

The simplest way to achieve this is to use the task template's interface to define a condition query; this automatically inserts the action handler. Alternatively, you can create a custom action handler that uses ITK to verify criteria.

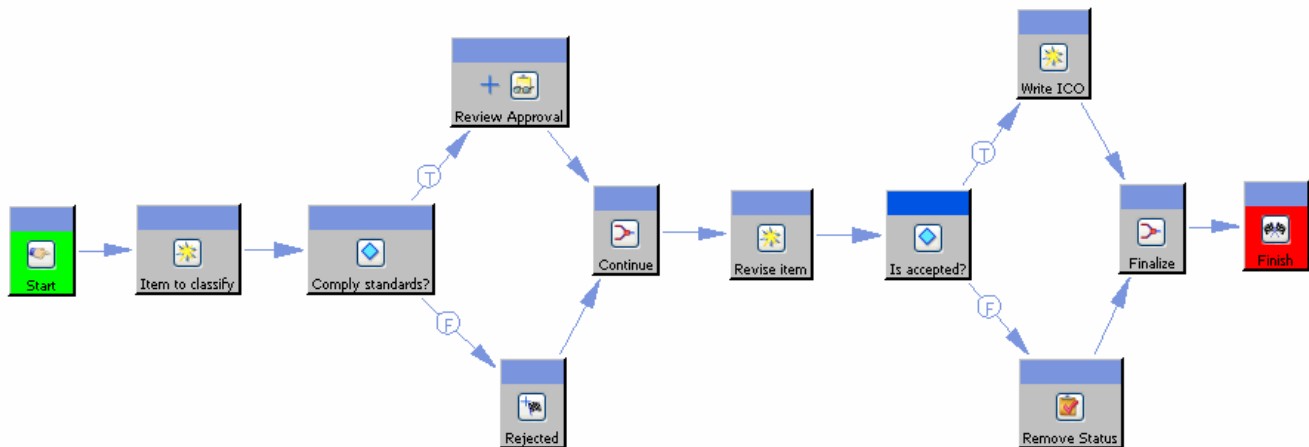
- **Manual condition task**

Does not define a query and does not add an action handler to the task template.

Optional activity

- Use **Condition** tasks in a process template.

In this activity, you configure manual and automatic tasks using the required handlers and a saved query. You also use a handler to add and remove status and exit the Workflow.



CCC Classify Item process template specifications

- The process signifies that the data is ready for library classification and includes two **Conditional** tasks.
- Data is received for classification.
- Check for standards compliance.
 - Complied items go to approval review.
 - Noncomplied items obtain **Rejected** status.
- Verify that the data is accepted.
 - Approved data is allowed for Classification.
 - If the status of the data is **Rejected**, data goes back to production.
- The process targets **CCC_Item Revision** and **CCC_Standard Revision**.
- Approvals:
 - Seeks approval of the Standards Engineer from the high performance group.

- Constraints:
 - A manual **Condition** task to check data.
 - An automatic **Condition** task to check the status of the data.
- The automatic **Condition** task requires the **is_rejected** query.

Review questions

1. What are the values of the **result attribute** in a **Condition** task?

Select one answer.

- **True** and **False**
- **True, False, and Unset**
- **Set** and **Unset**

2. How many types of **Condition** tasks are available?

Select one answer.

- **Automatic** and **Manual**
- **Automatic** only
- **Do, Review, and Acknowledge**

3. How do you configure an automatic **Condition** task?

Select all that apply.

- Adding a **Review** task
- Using a condition query
- Using an action handler that sets the task's result to true or false

4. Why is an **Or task** typically used in a process template?

Select all that apply.

- To make the process look better
- To reconcile task branches
- To resolve the parallel paths into a single path when using a condition task

Deploying process templates

Once you create Workflow templates, you can assign specific templates to a group. The group can have multiple assigned process templates with each process template based on the type of object being initiated from the **File® New® Process** dialog box.

Using the template filter

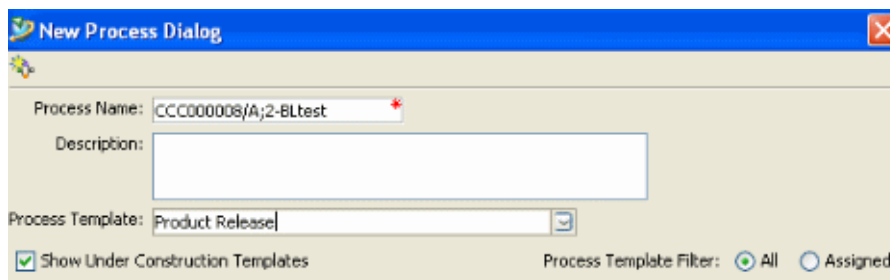
Process templates are assigned by using **Preferences** for the specified group.

In the Workflow Designer, choose **Edit® Template Filter** and then select templates for user groups and object types.

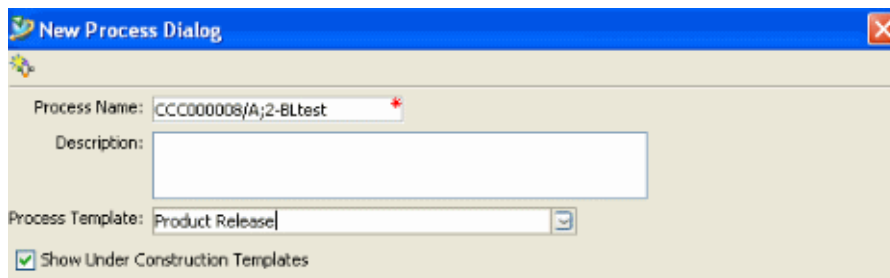
Blocking users from choosing from all process templates

After process templates are assigned to the groups, you can set the **CR_allow_alternate_procedures** preference to disable users from choosing any process template when choosing **File® New® Process**.

- Setting **CR_allow_alternate_procedures** to **any**, allows the **All** option in the **Process Template Filter** to be selected.



- Setting **CR_allow_alternate_procedures** to **none**, removes the **All** and **Assigned** options from the **New Process** dialog box.



Activity

In this activity, you use the **Process Template Filter** function to assign default templates to user groups.

- Assigning default process templates.

Review questions

1. How do you assign process templates to a group of users?

Select all that apply.

- Selecting templates for user groups and object types
- Setting the **CR_allow_alterate_procedures** preference to none
- Using the **Template Filter** function

Summary

Topics learned in this lesson:

1. Define basic concepts of Workflow templates.
2. Release data using a quick-release process.
3. Create multiple task Workflow process templates.
4. Configure a baseline process.
5. Add security to tasks in process templates.
6. Add conditional tasks to Workflow process templates.
7. Import and export process templates.
8. Assign templates to users and groups.

Lesson

19 Course summary

Course objectives

The overall objective for this course was to extend the data model by creating business objects, classes, options, list of values, constants, and rules and to configure the application for use by creating business data and processes.

1. To understand what *administrative tasks* are done using the Business Modeler IDE and rich client interfaces.
2. To configure the Business Modeler IDE.
3. To understand the basic Business Modeler IDE process.
4. To understand how to work with *business objects* and *classes*.
5. To edit the data model graphically.
6. To understand and attach a *list of values (LOV)* to a property.
7. To understand how configuration *options* are used.
8. To understand and configure *constants*.
9. To understand what *rules* do.
10. To understand how data model files are used.
11. To import a data model file.
12. To understand how file repository projects are used.
13. To import a file repository project by using the Import File Repository interface.
14. To create an *organization*.
15. To create *saved queries*.
16. To configure the working environment and managing *preferences*.

- 17. To create *report definitions*.
- 18. To configure *access permissions*.
- 19. To configure *projects*.
- 20. To create *workflow process templates*.

Index

A

Access manager
 Object based protection 16-4
 Rules based protection 16-3
Access Manager
 ACL 16-5, 16-10–16-13
 Add rule 16-9
 Export 16-18–16-19
 Group security 16-15
 Import 16-18–16-19
 Interface 16-6
 Protecting 1-28
Account
 Add role to group 12-23
 Add user to group 12-24
 Create 2-4, 12-17
 Generation 12-28
 Group 12-20
 infodba 12-14
 Organization interface 2-5, 12-18
 Organization structure 12-19
 Person 2-6, 12-21
 System administration 12-15
 User 2-7, 12-22
Action handlers 18-29
Active extension files 3-30
Adding
 Attributes 4-31
 Business Modeler IDE template
 projects 3-18
 Change options 8-20
 Classes 4-24
 Condition tasks 18-45
 Dataset business objects 7-6
 Form business objects 4-35
 Item business objects 4-10
 Lists of values (LOVs) 6-5
 Naming rules 10-13–10-14

Notes 8-4
Projects 3-15
Status objects 8-7
Tool objects 7-4
Units of measure 8-11
View objects 8-16
Adding rules 16-9
Administrative reports
 Admin-Employee Information ... 15-5
 Admin-Group/Role
 Membership 15-5
 Object Ownership 15-5
 Objects by Status 15-5
Approver 18-15
Architecture 1-34
Array column 4-30
Array Length column 4-30
Assigning default process
 templates 18-48
Attach
 Naming rules 10-13
Attaching
 LOVs 6-9
 Naming rules 10-16
Attachment 18-15
 Reference object 18-15
 Target object 18-15
Attribute Name column 4-28
Attributes 13-6, 13-15, 18-25
 Adding 4-31
 Data Model 15-8

B

Baseline process templates
 Create 18-37
Boolean rules 13-17
Boxes
 Class Attribute Selection 13-33

- Query Builder 13-3
- Browse mode 18-7
- Business Modeler IDE
 - Basic process 3-28–3-29
 - Configure 3-14
 - Definition 3-2
 - Eclipse 3-3
 - Enable 3-8
 - Extending the data model 3-26
 - Installing 3-9
 - Interface 3-4–3-6
 - Server profile 3-20, 3-24
 - Start 3-10–3-11
- Business Modeler IDE template
 - projects 3-18
- Business object constants 9-9
 - Examples 9-9
- Business object display rules
 - Adding 10-4–10-5
- Business objects 1-14
 - Constants 9-10
 - Dataset 7-2, 7-6
 - Form 4-34–4-35, 4-38
 - Item 4-10
 - Properties 4-13
 - Search 4-7
- Bypass switch 12-15
- C**
 - Candidate Key column 4-30
 - Cascading LOVs 6-15, 6-18
 - Change 8-18
 - Change Management 18-34
 - Change Management workflow
 - handlers 18-34
 - Change objects 18-34
 - Change options 8-20
 - Class Attribute Selection dialog
 - box 13-33
 - Class Selection dialog box 13-7
 - Classes 1-14
 - Adding 4-24
 - Closure rules 15-10
 - Compound property rules .. 10-20–10-21
 - Adding 10-22
 - Concepts
 - Custom reports 15-3
 - Item reports 15-3
 - Summary reports 15-3
 - Condition tasks, adding 18-45
 - Configuring supplier security for external
 - data 16-17
 - Configuring supplier security for internal
 - data 16-16
 - Constants 1-18, 9-2
 - Business object 9-9–9-10
 - Framework 9-3
 - Global 9-5–9-6
 - Precedence 9-4
 - Property 9-13–9-14, 9-17
 - Working with 9-2
 - Create a new Teamcenter UML diagram
 - wizard 5-10
 - Creating
 - Attributes 4-31
 - Business Modeler IDE template
 - projects 3-18
 - Business object constants 9-10
 - Classes 4-24
 - Dataset business objects 7-6
 - Form business objects 4-35
 - Global constants 9-6
 - Item business objects 4-8, 4-10
 - Projects 3-15
 - Property constants 9-17
- Custom reports
 - Concepts 15-3
- D**
 - Data model 1-12–1-13
 - Basic item data model 4-2
 - Business object model 4-2
 - Business objects 4-5
 - Classes 4-20
 - Deploying 3-34
 - Example 4-4
 - Extending 4-3
 - Files 11-2
 - Inheritance 5-4
 - Logical data model 4-2
 - Packaging 11-3
 - Saving 3-32

Dataset
 Tool 7-3
 Dataset business objects 7-2, 7-6
 Deep copy rules 10-26
 Adding 10-28
 hierarchy 10-27
 Defining search criteria 13-15
 Deploying data model
 changes 3-33–3-34
 Digital Dashboard 1-22
 Discipline 12-9

E

Eclipse 3-3
 Framework 3-7
 ECO reports
 ECO Details Report 15-5
 ECO Signoff Details 15-5
 Environment 1-38
 Export As String column 4-30
 Export report definitions
 Import_export_reports utility .. 15-17
 Extension files
 Setting 3-30
 Extensions rules 10-31
 Predefined 10-32, 10-34
 External groups 16-17

F

Follow on Export column 4-30
 Form business objects 4-34
 Creating 4-35
 Hide properties 4-39
 Properties 4-38
 Storage class form 4-37

G

Generic process templates
 Create 18-17
 Global constants 9-5–9-6
 Examples 9-5
 GRM rules 10-7
 Adding 10-9

Group 12-5, 12-20
 Add role 12-23
 Add user 12-24
 Group member 12-5
 Role 12-8
 Subgroup 12-6
 Group-level security
 Configuring for
 suppliers 16-16–16-17
 External groups 16-17
 Internal groups 16-16

H

Handlers 18-25
 Handlers arguments 18-28
 Hide properties 4-39
 Hierarchical LOVs 6-15, 6-18
 Hints, creating queries using 13-9
 How rules work 16-8

I

ID context rules 10-19
 Import files 7-9
 Import report definitions
 Import_export_reports utility .. 15-17
 Import_file utility 7-11
 Importing
 Model files 11-9
 Projects 11-7
 infodba 12-14
 Inherited column 4-16, 4-29
 Initial value column 4-29
 Installing
 Business Modeler IDE 3-9
 Templates 11-6
 Interface
 Business Modeler IDE 1-2
 Rich client 1-3
 Internal groups 16-16
 Item business objects
 Creating 4-8, 4-10
 Item Ownership report 15-5
 Item reports
 Concepts 15-3

- Items by Status report 15-5
- J**
- Job 18-15
- K**
- Keyword search queries 13-9
- L**
- Launching the Business Modeler
 - IDE 3-11
- List of values (LOV) 1-16, 6-2
 - Add, remove or clear 6-11
- Lists of values (LOVs)
 - Adding 6-5
 - Attaching 6-9
 - Cascading 6-12, 6-15, 6-18
 - Filter 6-12, 6-14
 - Hierarchical 6-15, 6-18
 - Interdependent 6-12, 6-17
 - interface 6-3
- Logical operators 13-18
- Lower Bound column 4-30
- M**
- make_user 12-28
 - Examples 12-29
- Modifiable queries 13-6
- N**
- Named references 7-9–7-10
- Naming rules 10-12
 - Examples 10-18
- No Backpointer column 4-30
- Note 8-3
- Notes 8-4
- Nulls Allowed column 4-30
- O**
- Open in UML Editor menu 5-7–5-8
- Options 1-17, 8-2
 - Change 8-18
 - Note 8-3
- Status 8-6
- Tool 7-3
- Unit of measure 8-9
- View 8-15
- Organization 1-24, 2-2, 12-2
 - Account generation 12-28
 - Export 12-31–12-32
 - Group 12-5
 - Group hierarchy 12-7
 - Import 12-31, 12-33
 - Passwords 12-10
 - Person 12-3
 - Role 12-8
 - Search 12-34–12-35
 - User 12-4
 - Volumes 2-3, 12-11
- P**
- Packaging extensions for
 - installation 11-3
- Palette, UML 5-8, 5-11
- Passwords 12-10
- Persistent object model (POM) 1-15
- Person 2-6, 12-3, 12-21
- PLM XML Export Import
 - Administration 15-7
- PLM XML report data 15-7
- POM (persistent object model) 4-24
- Preference
 - Definition 1-25
 - Import 14-19–14-20
 - Import and export 14-18
- Preference precedence 14-11
- Preference scope 14-10
- Preferences 14-2, 14-8
 - categories 14-7
 - Index 14-5
 - Options 14-5
 - Organization 14-6
 - subcategories 14-7
- Preferences options 14-2
- Prerequisites for Workflow
 - Designer 18-7
- Privileged team member 17-11
- Process template 18-2
- Process templates

-
- Create 18-16
 - Export 18-39
 - Import 18-39
 - Process Templates 18-15
 - Product Structure reports 15-5
 - Project 17-11
 - Assign data 17-15
 - Assign objects 17-16–17-18
 - Create 17-12
 - Manage 17-14
 - Privileged team member 17-11
 - Project administrator 17-11
 - Project team administrator 17-11
 - Project team member 17-11
 - Propagation rules 17-19
 - Security rule tree 17-13
 - Project administrator 17-11
 - Project team administrator 17-11
 - Project team member 17-11
 - Projects 1-29, 17-2
 - Business Modeler IDE
 - template 3-18
 - Create 17-7
 - Creating 3-15
 - Files 3-16
 - Importing 11-7
 - Interface 17-8
 - Naming rules 17-4
 - Security 17-6
 - Team members 17-5
 - Who can define 17-3
 - Propagation rules 17-11
 - Properties
 - Business objects 4-13
 - Constants 9-17
 - Data Model 15-8
 - Form business objects 4-38
 - Property constants 9-13
 - Examples 9-13
 - Reference 9-14
 - Update 9-16
 - Property Name column 4-14
 - Property sets 15-11
 - Protecting
 - Teamcenter data 16-2
 - Public Read column 4-30
 - Public Write column 4-30
 - Q**
 - Queries
 - Based on existing definitions ... 13-10
 - Creating using hints feature 13-9
 - Referenced-By 13-9, 13-31
 - With keyword search 13-9
 - Query
 - Create 13-13
 - Custom item type 13-24
 - Definition 1-26, 13-2
 - Workspace objects 13-8
 - Query Builder dialog box 13-3
 - Quick-release process 18-9
 - R**
 - Reference Class column 4-29
 - Reference object 18-15
 - Referenced-by queries 13-9, 13-31
 - Relation propagation rules 17-11
 - Report builder
 - Definition 1-27
 - Reports
 - Admin-Employee Information ... 15-5
 - Admin-Group/Role
 - Membership 15-5
 - Admin-Item Ownership 15-5
 - Admin-Items by Status 15-5
 - Admin-Object Ownership 15-5
 - Admin-Objects by Status 15-5
 - ECO Details Report 15-5
 - ECO Signoff Details 15-5
 - PS - BOM Structure 15-5
 - Resource Pool Subscription 18-23
 - Rich client
 - Applications 1-4
 - Interface 1-3
 - Role 12-8
 - Add to group 12-23
 - Rule handlers 18-30
 - Rule tree 16-7
 - Rule tree evaluation 16-8
 - Rule tree precedence 16-7
 - Rules 1-19, 10-2

- Adding 16-9
 - Business object display 10-4–10-5
 - Compound property 10-20–10-22
 - Deep copy 10-26–10-28
 - Extensions 10-31–10-32, 10-34
 - GRM 10-7, 10-9
 - ID context 10-19
 - Naming 10-12
 - Subbranch precedence 16-7
 - Tree 16-7
- S**
- Samples 3-11
 - Saving data model 3-32
 - Scripts 1-39
 - Search criteria
 - Attributes 13-15
 - Boolean rules 13-17
 - Default value 13-17
 - Defining 13-15
 - Logical operators 13-18
 - User entry L10N key 13-16
 - User entry name 13-16
 - Search for classes 13-7
 - Server
 - Deploying changes to 3-34
 - Server profile 3-20
 - Two-tier 3-24
 - Set active extension files 3-30
 - Size column 4-29
 - Source Class column 4-29
 - Source column 4-16
 - Start the Business Modeler IDE 3-10
 - Start the IMR 3-10
 - Starting the Business Modeler
 - IDE 3-11
 - Status 8-6
 - Status objects 8-7
 - Storage Type column 4-15
 - Subscriptions 1-32
 - Summary reports
 - Concepts 15-3
 - Supplier 17-11
- T**
- Target object 18-15
 - Task handlers
 - Creating 18-28
 - Task template 18-2
 - Task templates
 - Handlers 18-28
 - Tasks 18-15
 - Attributes 18-25
 - Rich Client Administration 1-23
 - Team members 17-11
 - Privileged team member 17-11
 - Project administrator 17-11
 - Project team administrator 17-11
 - Team member 17-11
 - Teamcenter Environment Manager
 - (TEM) 3-9
 - Templates 1-21
 - Adding extensions 11-5
 - Deploying 3-34
 - In general 3-27
 - Installing 11-6
 - Packaging 11-3
 - Workflow process 18-2
 - Workflow task 18-2
 - Tool 7-3
 - Tool objects 7-4
 - Transfer mode objects 15-9
 - Transient column 4-30
 - Tutorials 3-11
 - Type column 4-14, 4-28
 - Types of modifiable queries 13-6
- U**
- UML editor 1-14, 5-2
 - Add business object 5-11
 - Add class 5-11
 - Data model 5-6
 - Preferences 5-5
 - UML diagram 5-3
 - Unique column 4-30
 - Unit of measure 8-9
 - Units of measure objects 8-11
 - Upper Bound column 4-30

-
- User 2-7, 12-4, 12-22
 - Add to group 12-24
 - User entry L10N key 13-16
 - User entry name 13-16
 - User passwords 12-10

 - V**
 - View 8-15
 - View objects 8-16
 - Volumes 2-3, 12-11

 - W**
 - Welcome window 3-11
 - Workflow ACLs 18-26
 - Workflow Designer
 - Definition 1-30
 - Workflow Designer interface 18-7
 - Workflow process template 18-2
 - Workflow task template 18-2
 - Workflow tasks 18-18
 - Workflow templates 18-14
 - Create 18-16
 - Create baseline process templates 18-37
 - Create generic process templates 18-17
 - Enterprise process modeling (EPM) 18-5
 - Exporting 18-40
 - Importing 18-41
 - Model a process 18-14
 - Process terms 18-15
 - Tasks 18-18
 - Workflow Designer interface 18-7
 - Workspace 3-18

This page left blank intentionally.

Appendix

A Classroom system information

Application and data model administration class data sheet

| Data item | Data value | Domain |
|---------------------|------------|----------------|
| OS user ID | | Local computer |
| OS password | | |
| OS user ID | | Virtual image |
| OS password | | |
| Teamcenter user ID | | Virtual image |
| Teamcenter password | | |
| TC_DATA | | Virtual image |
| TC_ROOT | | Virtual image |
| TC_VOLS | | Virtual image |
| TEMPLATES_DIR | | Virtual image |
| PROJECTS_DIR | | Virtual image |
| CORP_SERVER_CONFIG | | Virtual image |
| TEMPLATES | | Virtual image |

Appendix

B Course agenda

Application and data model administration course agenda

Daily schedule based on 8:30 am to 4:30 pm schedule.

| | | |
|--------------|-----------------|--|
| Day 1 | Introduction | |
| | Course overview | |
| | Lesson 1 | Introduction to administration |
| | Lesson 2 | Introduction to organization |
| | Lesson 3 | Introduction to the Business Modeler IDE |
| | Lesson 4 | Data model |
| | Lesson 5 | UML editor |
| Day 2 | Lesson 6 | Lists of values |
| | Lesson 7 | Datasets |
| | Lesson 8 | Options |
| | Lesson 9 | Constants |
| | Lesson 10 | Rules |
| | Lesson 11 | Data model files |
| Day 3 | Lesson 12 | Organization hierarchy |
| | Lesson 13 | Query Builder definitions |
| | Lesson 14 | Preference management |
| Day 4 | Lesson 15 | Report Builder definitions |
| | Lesson 16 | Access Manager |
| | Lesson 17 | Projects to control access |
| Day 5 | Lesson 18 | Workflow process modeling |
| | Course summary | |

Appendix

C Student profile

Teamcenter student profile

Name _____ Date _____

Employer _____

U.S. citizen? Yes / No

When is your planned departure time? _____ am/pm

Please answer the following questions as honestly as you can. We strive to provide training that meets your needs. If you have any additional comments, please write them on the back of this form.

1. Job title: _____

2. Current responsibilities: _____

3. How long have you held these responsibilities? Years _____

4. How long have you been working with PDM and CAD/CAM systems?
Years _____

5. What other PDM systems are you familiar with?

6. What other CAD/CAM systems are you familiar with?

7. Are you currently using Teamcenter? Yes / No Product _____
Version _____ Hours per week? _____

8. Are you currently using NX? Yes / No Version _____ Hours per week? _____

9. What are the primary uses of Teamcenter at your site? _____

10. What do you model in your NX part files (castings, assemblies, and so on)?

11. List other completed PDM or CAD/CAM courses and the provider including self-paced/computer-based training:

| Course | Provider |
|--------|----------|
| | |
| | |
| | |
| | |
| | |
| | |

12. Check the box that best describes your current skill level in the various Teamcenter and NX disciplines listed in the following table.

| | None | Novice | Intermediate | Advanced | Future |
|--------------------------------|------|--------|--------------|----------|--------|
| NX user | | | | | |
| Teamcenter Integrations for NX | | | | | |
| Teamcenter user | | | | | |
| Teamcenter admin | | | | | |
| Teamcenter customizer (BM) | | | | | |

Additional comments:

Appendix

D Course evaluation

Course Evaluation



You may instead complete this evaluation online at:

<http://training.ugs.com/eval>

The online evaluation will require you to enter a "session ID" for this class. This should be provided by your instructor and noted here: Session ID _____

Course Name _____ Course # _____
 Dates _____ thru _____

Please share your opinion in all of the following sections with a "check" in the appropriate box:

Instructor: _____ ☒

If there were 2 instructors, please evaluate the 2nd instructor with "X's"

Instructor: _____ ☒

| | STRONGLY DISAGREE | DISAGREE | SOMEWHAT DISAGREE | SOMEWHAT AGREE | AGREE | STRONGLY AGREE |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1. ...clearly explained the course objectives..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2. ...was knowledgeable about the subject..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 3. ...answered my questions appropriately..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4. ... encouraged questions in class..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5. ...was well spoken and a good communicator..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 6. ...was well prepared to deliver the course..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 7. ...made good use of the training time..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 8. ...conducted themselves professionally..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 9. ...used examples relevant to the course and audience..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 10. ...provided enough time to complete the exercises..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 11. ...used review and summary to emphasize important information..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 12. ...did all they could to help the class meet the course objectives..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Comments on overall impression of instructor(s):

Overall impression of instructor(s).....Poor ☐ ☐ ☐ ☐ ☐ ☐ Excellent

Suggestions for improvement of course delivery: _____

What you liked best about the course delivery: _____

Class Logistics:

| | | | | | | |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1. The training facilities were comfortable, clean, and provided a good learning environment..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2. The computer equipment was reliable..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 3. The software performed properly..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4. The overhead projection unit was clear and working properly..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5. The registration and confirmation process was efficient..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

SEE BACK

Course Evaluation

Course Name _____ Course # _____
 Dates _____ thru _____

Please share your opinion for all of the following sections with a "check" in the appropriate box:

Material:

- | | STRONGLY
DISAGREE | DISAGREE | SOMEWHAT
DISAGREE | SOMEWHAT
AGREE | AGREE | STRONGLY
AGREE |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1. The training material supported the course and lesson objectives..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2. The training material contained all topics needed to complete the projects..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 3. The training material provided clear and descriptive directions..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4. The training material was easy to read and understand..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5. The course flowed in a logical and meaningful manner..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 6. How appropriate was the length of the course relative to the material?..... <input type="checkbox"/> Too short <input type="checkbox"/> Too long <input type="checkbox"/> Just right | | | | | | |

Comments on Course and Material: _____

Overall impression of course.....Poor ☐ ☐ ☐ ☐ ☐ ☐ Excellent

Hotels: (We try to leverage this information to better accommodate our customers)

1. Name of the hotel _____ Best hotel I've stayed at... ☐ ☐ ☐ ☐ ☐ ☐
2. Was this hotel recommended during your registration process?.....☐ YES ☐ NO
3. Problem? (brief description) _____

Student:

- | | | | | | | |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1. I met the prerequisites for the class (I had the skills I needed)..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 2. My objectives were consistent with the course objectives..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 3. I will be able to use the skills I have learned on my job..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 4. My expectations for this course were met..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |
| 5. I am confident that with practice I will become proficient..... | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |

Student Name (optional): _____ Location/room _____

☐ Please "check" this box if you would like your comments featured in our training publications.

☐ Please "check" this box if you would like to receive more information on our other courses and services.

*Thank you for your business.
 We hope to continue to provide your training and personal development for the future!*

Rev-2/1/06-btr