

Teamcenter Application and Data Model Administration

**Student Guide
July 2009
MT25460 – Teamcenter 8**

PRELIMINARY
8/3/2009 21:41:11

**Publication Number
MT25460_S_0801**

Proprietary and restricted rights notice

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

© 2009 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Contents

Course overview	19
Course objectives	19
Key benefits	20
Prerequisites	20
Audience	20
Learning tracks	21
Training materials provided	21
Accessing Teamcenter online help	22
 Introduction to administration	 1-1
Introduction to Administering Teamcenter	1-2
Teamcenter functional requirements (example)	1-3
Teamcenter administrative applications	1-4
Teamcenter administrative applications	1-5
Business Modeler IDE administration tasks	1-7
Business Modeler IDE administration tasks	1-8
Teamcenter functional requirements (example)	1-9
Administrative interfaces	1-10
Rich client administration tasks	1-14
Teamcenter configuration process	1-15
Activities	1-16
Review questions	1-17
Teamcenter Preferences	1-18
Teamcenter environment	1-19
Teamcenter environment variables	1-20
Command line scripts	1-21
tcCustDev command script	1-22
Organization setup	1-25
Create DBA and test user	1-27
Teamcenter environment for utilities	1-28
The make_user utility	1-29
Teamcenter architecture overview	1-30
Two-tier architecture logical view	1-31
Four-tier architecture logical view	1-32
Two-tier architecture logical view	1-34
Four-tier architecture logical view	1-35
Teamcenter Folders	1-37
Teamcenter Root folder	1-38

Teamcenter Sample folder (ITK files)	1-40
Teamcenter Sample folder	1-41
Summary	1-42
Using Teamcenter Options and Utilities	2-1
Introduction to preferences	2-2
Installation and upgrades	2-3
Preference interface	2-4
Working with Options	2-5
Viewing and setting options	2-6
Working with Preferences	2-7
Viewing and setting preferences	2-8
Preference categorization	2-10
Preference scope	2-11
Preference precedence	2-13
Key Teamcenter preferences	2-15
Search by keyword	2-18
Search by organization	2-19
Review of Teamcenter Preferences	2-20
Viewing and setting preferences	2-21
Preferences Index and Search	2-22
Preference scope	2-23
Teamcenter customization preferences	2-24
Activities	2-25
Review questions	2-26
Generating preference reports	2-28
Create preference reports	2-29
Import and export preferences	2-30
Import preferences in to the database	2-31
Export preferences from the database to an XML file	2-32
preferences_manager utility	2-33
Activities	2-35
Review questions	2-36
Summary	2-39
The Business Modeler IDE fundamentals	3-1
Create a Business Modeler IDE template project	3-2
Business Modeler IDE extension files	3-3
Files in a template project	3-4
Understanding custom versus COTS templates	3-6
Introduction to the Business Modeler IDE	3-7
Basic concepts for using the Business Modeler IDE	3-8
Business Modeler IDE interface	3-9
Business Modeler IDE perspective	3-10
Business Modeler IDE views	3-11

Business Objects view	3-12
Classes view	3-13
Extensions view	3-14
Console view in the Business Modeler IDE	3-16
Navigator view in the Business Modeler IDE	3-17
Configure the Business Modeler IDE	3-18
Server connection profiles	3-19
Server connection profile created during the Business Modeler IDE installation	3-20
Add a server connection profile	3-21
Server connection profile properties	3-22
Deploy with a two-tier server connection profile	3-23
Relevant Business Modeler IDE installation information	3-24
Before you begin	3-25
Install the Business Modeler IDE as a stand-alone application	3-26
Install the Business Modeler IDE as a stand-alone application (continued)	3-27
Start the Business Modeler IDE	3-28
Start the IMR	3-29
Activities	3-30
Review questions	3-31
Activities	3-32
Review questions	3-33
Activity	3-34
Review questions	3-35
Summary	3-36
CreateDescriptor tab	3-37
Manage business objects for creation	3-38
Managing Templates	4-1
Overview of template installation	4-2
Introduction to templates	4-3
Templates overview	4-4
Install the initial template	4-5
Install a subsequent template	4-6
Hot deploy a template	4-8
Deploy extensions overview	4-9
Deploying templates	4-10
Deploy a template to a test server	4-11
Deploy operational data to a production server	4-12
Understanding custom versus COTS templates	4-14
Import a Business Modeler IDE template project	4-15
Import a template file	4-17
Activities	4-19
Review questions	4-20
Summary	4-21

Teamcenter data model	5-1
Data model concepts introduction	5-2
What are business objects	5-3
What are classes	5-4
POM interface to database	5-5
Introduction to attributes	5-6
Working with properties	5-7
Why create business objects?	5-8
Find business objects	5-9
Find classes	5-10
Teamcenter business objects	5-11
Data model	5-12
Basic item structure	5-13
Business objects and classes	5-14
Teamcenter POM schema	5-15
What is the data model	5-17
Business objects and properties	5-18
Introduction to the UML editor	5-19
What is a UML diagram	5-20
Outline view in the Business Modeler IDE	5-21
Managing the data model using the UML editor	5-22
Opening a business object in the UML editor	5-23
Open a class or business object in the UML editor	5-24
Create a new UML diagram	5-26
Create a new class or business object	5-27
Activities	5-28
Review questions	5-29
Class attributes reference	5-31
Tasks for using the data model	5-32
Classes, attributes, and compound properties	5-33
What are class attributes	5-34
Class attributes table	5-35
Working with compound properties	5-38
Activities	5-39
Review questions	5-40
Extensions overview	5-41
Generic Relationship Manager	5-42
Constants introduction	5-43
Introduction to constants	5-44
Constants framework	5-45
Constants precedence	5-46
Property constants reference	5-47
Working with property constants	5-49
Activities	5-50
Review questions	5-51
Constants	5-52

Summary	5-53
Summary	5-54
Main topic template	5-55
Subtopic template	5-56
My Modifier 1	5-57
My Modifier 2	5-58
Item business object configuration	6-1
Data model custom introduction	6-2
Configuring a new item business object and properties	6-3
Extending item business objects	6-4
Creating item business objects	6-5
Create an item business object	6-7
Activities	6-9
Review questions	6-10
Extensions overview	6-11
Basic tasks using the Business Modeler IDE	6-12
Setting active extension files	6-13
Defining commonly used business objects	6-14
Basic item structure	6-15
Data model extension example	6-16
Defining business objects	6-17
Add a persistent property	6-18
Create a persistent property	6-19
Property constants	6-20
Modifiable property constant	6-22
Deploy with a two-tier server connection profile	6-23
Configuring a compound property	6-24
Defining a compound property rule path	6-25
Add a compound property	6-26
Activities	6-29
Review questions	6-30
Configuring a property constant	6-31
Changing a property constant value	6-32
Create a property constant	6-33
Activities	6-36
Review questions	6-37
Summary	6-38
Configuring new business object rules	6-39
Configuring a new dataset business object	6-40
Extensions overview	6-41
Naming objects	6-42
Form business object configuration	7-1
Configuring a new form business object	7-2

Defining forms	7-3
Create a form business object	7-4
Storage class form	7-6
Add, change, or remove properties on a form business objects	7-7
Hide properties on a form business object	7-8
Activities	7-9
Review questions	7-10
Configuring a new class	7-11
Topics Held for Inserting Later	7-12
Adding new classes	7-13
Add a new class	7-14
Add or change attributes on classes	7-16
Activities	7-18
Review questions	7-19
Summary	7-21
Relation business object configuration	8-1
Summary	8-2
LOV (list of value) extensions	9-1
Introduction to lists of values (LOVs)	9-2
List of values (LOV) interface	9-3
Add a list of values (LOV)	9-5
Attach an LOV to a property	9-10
Add, remove, or clear a value to an LOV	9-12
LOV variations	9-13
Filter LOV	9-14
Cascading LOV	9-15
Create a filter LOV	9-16
Create a cascading LOV	9-18
Example cascading LOV	9-19
Create an interdependent LOV	9-20
Activities	9-22
Review questions	9-23
Summary	9-24
Dataset business object configuration	10-1
Introduction to dataset business objects	10-2
Introduction to tools	10-3
Add a tool	10-4
Create a dataset business object	10-6
Dataset and named references	10-9
Named references	10-10
Activities	10-11
Review questions	10-12

Summary	10-14
Compound properties	11-1
Summary	11-2
Option extensions	12-1
Introduction to options	12-2
Introduction to notes	12-3
Add a note type	12-4
Introduction to status	12-6
Add a status	12-7
Introduction to units of measure	12-9
Add a unit of measure	12-10
Introduction to change	12-12
Add a change	12-15
Activities	12-18
Review questions	12-19
Summary	12-21
Rule extensions	13-1
Introduction to rules	13-2
Working with business object display rules	13-4
Add a business object display rule	13-5
Working with GRM rules	13-7
About GRM rules	13-8
Add a GRM rule	13-10
Activities	13-12
Review questions	13-13
Working with naming rules	13-14
Create and attach a naming rule	13-15
Add a naming rule	13-16
Attach a naming rule to a property	13-19
Using literal variables in patterns	13-21
Using counters with naming rules	13-22
Working with ID context rules	13-23
Activities	13-24
Review questions	13-25
Working with deep copy rules	13-26
Deep copy example	13-27
Add a deep copy rule	13-28
Working with extensions	13-32
Predefined extensions	13-33
Add a predefined extension to a business object	13-35
Activities	13-37
Review questions	13-38

Summary	13-40
Data model files	14-1
Introduction to managing templates	14-2
Using an SCM system	14-3
Install a template to a production server	14-4
Package extensions into a template	14-5
Adding extensions to the template	14-7
Install a template using TEM	14-8
Activities	14-10
Review questions	14-11
Summary	14-12
Organization	15-1
Introduction to Organization	15-2
Organization interface	15-3
Organization hierarchy	15-4
Basic concepts for using Organization	15-5
Typical organization administration tasks	15-7
Defining persons	15-8
Defining users	15-9
Passwords	15-10
Teamcenter Security Services	15-11
Defining roles	15-12
Defining groups	15-13
Defining subgroups	15-14
Group hierarchies	15-15
Defining volumes	15-16
Defining authorized data access licenses	15-17
Review questions	15-18
Defining administrative privileges	15-19
infodba account	15-20
System administration accounts	15-21
Creating your virtual organization	15-22
Creating the organization structure	15-24
Creating a volume	15-25
Modifying volume properties	15-28
Controlling volume access	15-29
Creating a group	15-30
Creating a role	15-31
Creating a person	15-32
Creating a user	15-34
Group member settings	15-38
Creating an authorized data access license	15-39
Add a new role to a group using the Organization Role wizard ..	15-41

Add a new user to a group/role using the Organization User wizard	15-43
Add an existing user to a role/group using the Organization User wizard	15-46
Changing user status	15-48
Deactivate a user account	15-49
Activate a user account	15-51
Using Organization find	15-52
Managing group members	15-53
Remove a member from a group	15-54
Activate a group member	15-55
Deactivate a group member	15-56
Suppress the display of inactive group members in the Organization tree	15-57
Activities	15-58
Review questions	15-59
Utility account generation	15-60
The make_user utility examples	15-61
Activity	15-63
Introduction to Authorization	15-64
Authorization interface	15-65
Basic concepts of using Authorization	15-66
System-level authorization rules	15-67
Basic tasks using Authorization	15-68
Import and export organization	15-69
Export organization	15-70
Import organization	15-71
Activities	15-72
Review questions	15-73
Summary	15-74
More Teamcenter Options and Utilities	16-1
import_file utility	16-17
Query Builder definitions	17-1
Introduction to Query Builder	17-2
Query Builder interface	17-3
Basic concepts for using Query Builder	17-4
Creating and managing queries	17-5
Create queries	17-6
Using class attribute selections	17-8
Add class attributes to search criteria	17-11
Search Criteria pane	17-12
Using search criteria clauses	17-14
Create a new query based on an existing definition	17-15

Activities	17-16
Review questions	17-17
Custom item type query definitions	17-21
Create a referenced-by query	17-23
Limit access to query definitions	17-26
Access control list	17-27
Importing and exporting query definitions	17-28
Import query definitions	17-29
Export query definitions	17-30
Activities	17-31
Review questions	17-32
Summary	17-35
Report Builder definitions	18-1
Introduction to Report Builder	18-2
Before you begin	18-3
Report Builder interface	18-4
Basic concepts for using Report Builder	18-5
Report Builder definition types	18-6
Report definition structure	18-7
Standard Teamcenter report definitions	18-8
PLM XML report data	18-9
Teamcenter data model quick review	18-10
Creating transfer mode objects	18-11
Creating closure rules	18-12
Defining property sets	18-13
Importing and exporting report definitions	18-14
Activities	18-15
Review questions	18-16
Summary	18-20
Using item revision configuration	19-1
Understanding revision rules	19-2
Elements of a revision configured product structure	19-3
Configuring privileged and unprivileged users	19-5
Creating a revision rule	19-7
Create a rule entry	19-8
Modify a rule entry	19-9
Delete a rule entry	19-10
Edit the entries within a rule	19-11
Modify the current revision rule	19-12
Defining revision rule entries	19-13
Defining a Working entry	19-14
Defining a Status entry	19-17
Defining a Latest entry	19-20

Defining a date entry	19-22
Defining a unit number entry	19-23
Defining an end item entry	19-24
Defining grouped entries	19-25
Applying a revision rule to a Structure Manager window	19-32
Set a revision rule	19-33
Set date/unit/end item	19-34
Set an override folder	19-35
Viewing revision rule information	19-36
Displaying and editing revision effectivity	19-38
Display revision effectivity data	19-39
Edit revision effectivity data	19-40
Managing nested effectivity	19-43
Configuring nested effectivity	19-44
Creating a configuration item	19-45
Creating a revision rule for nested effectivity	19-46
Create effectivity mapping on a configuration item	19-49
Managing product generations	19-51
Using revision rules from My Teamcenter	19-53
Managing occurrence to part relationships	19-54
Show occurrences of superseded item revision	19-55
Update precise occurrences of superseded item revision	19-56
Capturing configurations	19-57
Using snapshots	19-58
Using baselines	19-62
Using intermediate data captures	19-69
Access Manager	20-1
Introduction to Access Manager	20-2
Access Manager interface	20-3
Basic concepts for using Access Manager	20-4
Basic tasks using Access Manager	20-5
Protecting Teamcenter data	20-6
Rules-based protection	20-7
Object-based protection	20-8
Access control lists	20-9
Life cycle of data	20-10
Access Manager rule tree	20-11
How rules work	20-12
Rule syntax	20-13
Evaluating the rule tree for the effective ACL	20-14
Example rule tree evaluation by order of precedence	20-15
Activity	20-16
Example of compiling an effective ACL	20-17
Access privileges	20-19
Categories of accessors	20-22

Complex rule tree example	20-23
Activity	20-26
Review questions	20-27
Understanding the rule creation process	20-28
Add an Access Manager rule	20-29
Create an access control list (ACL)	20-30
Modify an Access Manager rule	20-31
Import and export the Access Manager rule tree	20-32
Import and export guidelines	20-33
Activities	20-34
Review questions	20-35
Best practices	20-36
Cautionary statements	20-38
Verifying the effect of access rules	20-39
View the rules from which privileges are derived	20-40
Activities	20-41
Configuring group security	20-42
Configuring security to prevent suppliers from viewing internal data	20-43
Configuring security for data owned by a supplier (external data)	20-44
Review questions	20-45
Summary	20-46
Projects to control access	21-1
Introduction to Project	21-2
Project administration window	21-3
Project administration buttons	21-4
Project administration tabs	21-5
Project quick links	21-6
Basic concepts about Project	21-7
Project administrators and team members	21-8
Using projects	21-10
Applying project security (Access Manager) rules	21-11
Project security rule tree	21-12
Modifying existing projects	21-13
Assigning data to projects	21-14
Automatically assigning objects to projects	21-15
autoAssignToProject availability	21-17
Configure automatic project assignment	21-18
Activities	21-19
Review questions	21-20
Summary	21-21

Teamcenter security	22-1
Introduction to Teamcenter security administration	22-2
Teamcenter security applications	22-3
Basic concepts	22-4
Teamcenter object model hierarchy	22-5
Authentication	22-6
Authorization	22-7
Rules-based protection	22-8
Object-based protection	22-9
Authorized data access	22-10
Controlling access to working data	22-11
Controlling access to in-process data	22-12
Activities	22-13
Review questions	22-14
Summary	22-16
 Workflow process modeling	 23-1
Introduction to Workflow Designer	23-2
Workflow process terminology	23-3
Basic concepts for using Workflow Designer	23-4
Workflow process template	23-4
Workflow task template	23-4
Basic tasks using Workflow Designer	23-5
Working with Workflow Designer	23-6
Workflow Designer interface	23-7
Workflow Tasks	23-8
Enterprise Process Modeling	23-9
Defining a process model	23-11
Sharing process templates	23-12
Export a process template	23-13
Import a process template	23-14
Activity	23-15
Review questions	23-16
Releasing data with a process	23-18
Using quick-release process templates	23-19
Create a quick-release process template	23-20
Batch release utility	23-22
Activities	23-24
Review questions	23-25
Creating a baseline process	23-27
Optional activities	23-28
Review questions	23-29
Change Management process model	23-30
Change management process	23-31
Introduction	23-34

Basic concepts for using Change Manager	23-35
Change objects	23-36
Workflow handlers designed for the Change Management process	23-38
Review questions	23-40
Summary	23-42
Workflow process templates	24-1
Creating Workflow process templates	24-2
Create a generic process template	24-3
Workflow task templates	24-4
Do task templates	24-7
Review task template	24-8
Acknowledge task template	24-10
Resource pool	24-11
Add Status task template	24-12
Modifying task behavior	24-13
Workflow accessors and privileges	24-14
Adding task handlers	24-15
Examples of useful handlers	24-17
Activities	24-19
Review questions	24-20
Adding secure tasks	24-24
Activity	24-25
Review questions	24-26
Failure path	24-27
Create failure paths	24-28
Activities	24-29
Adding Condition tasks	24-30
Set Condition task flow paths	24-32
Optional activities	24-34
Review questions	24-35
Troubleshooting your workflow	24-37
Perform actions against a workflow process	24-38
Cancel a workflow process	24-39
Resume a suspended task	24-40
Deploying process templates	24-41
Activity	24-43
Review question	24-44
Summary	24-45
Course summary	25-1
Appendix: Utilities	A-1
Utility import_file	A-7
Utility import_file	A-8

Utility xxx	A-9
Appendix: Data model reference	B-1
Subtopic	B-2
Subtopic	B-3
Index	Index-1

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 Teamcenter rich client interfaces.
- To configure the Business Modeler IDE and to execute the basic Business Modeler IDE process.
- To create and work with *business objects* and *classes*.
- To modify *properties* and *attributes*.
- To edit the data model graphically.
- To create and attach a *list of values (LOV)* to a property.
- To create and configure *options*.
- To create *rules* for *business objects*.
- To work with data model files and *template* projects.
- To create an *organization*.
- To configure the working environment and manage *preferences*.
- To import and export data with *utilities*.

- To create *saved queries* and *report definitions*.
- To configure *access permissions* and *projects* to control access.
- To create and manage *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.
- Importing legacy data.
- 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

Instructor Note:

We expect the students to have an understanding of how to use Teamcenter.

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

Instructor Note:

Primary audience for this class are the Application Administrators.

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 back of the <i>Student Guide</i>.</p> <p>Student profile is provided in the back of the <i>Student Guide</i>.</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>

Instructor Note:

The presentation is a subset of the *Student Guide*.

Warning

Short cut keys will not work until the presentation is brought into focus. Click in the presentation header area to bring the presentation into focus.

Common terms:

Term	Description
Window	Refers to the entire application window.
Pane	Refers to a section of the application window.
Dialog box	Refers to a separate window that displays choices or information.
Menu	Refers to a menu with commands or the left navigator menu.
Button	Refers to a box with text or icon that you click to perform an action.

Accessing Teamcenter online help

The *Teamcenter Help Library* covers functionality from end-user tasks to customization instructions.

To access the *Teamcenter Help Library*:

- In the rich client, choose **Help® Help® Help Library** or press the F2 key.
- In the thin client, choose **Help® Web Collection** to access the thin client help, or choose **Help® General Collection** to access the full library.

To access help for the current application:

- In the rich client, choose **Help® Help® Current Application** or press the F1 key.

Note

You cannot access application-specific help in the thin client.

Instructor Note:

Demonstrate how to access help information. Review how help topics are organized.

Point out how to find the *Glossary of Terms*, since we do not repeat this information in the student guide.

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](#)*
- *[Application Administration Guide](#)*
- *[Business Modeler IDE Guide](#)*

Introduction to Administering Teamcenter

Teamcenter administration involves the following two tasks:

- Configuring the rich client and thin client appearance and behavior.

Using the Teamcenter administrative application interfaces, you define your organization in Teamcenter, establish access controls for your organization information, create queries and reports, build your workflow processes, set up projects, and so on.

- Configuring the Teamcenter data model and business rules.

Using the Business Modeler IDE application, you define Teamcenter business objects, properties, and conditions based on your set of functional requirements. The conditions you define can include naming constraints, allowable value lists for properties, data validation checks, product configuration and revision rules, and so on.

Instructor Note:

Topic instructor notes.

Teamcenter functional requirements (example)

Your company requirements

1. The ...
2. The ...

Instructor Note:

Topic instructor notes.

Teamcenter administrative applications

















Using the Teamcenter administrative applications you define queries and reports, establish workflow processes, and implement projects and access controls. This will configure the appearance and behavior of the Teamcenter application interfaces for the end users.

This course is intended for Teamcenter administrators responsible for:

- Configuring the rich client and thin client interfaces.
- Modeling and maintaining your company's organization in Teamcenter.
- Defining queries and reports.
- Configuring security and data access rules.
- Designing and implementing workflow processes.
- Configuring access to applications and commands within applications.

Teamcenter administrative applications

The following applications are used to perform administrative tasks in the Teamcenter rich client:

Application	Description
 Access Manager	Manages permissions for data.
 Audit Manager	Tracks actions that take place on specified objects in the database.
 Authorization	Manages user access to administrative applications and utilities.
 Classification Administration	Manages the classification of parts within a hierarchy.
 Command Suppression	Manages the display of menu and toolbar commands in Teamcenter.
 My Teamcenter	Manages options and preferences.
 Organization	Manages your company's virtual organization, storage volumes, sites, calendars, and authorized data access licences.
 PLM XML Export Import Administration	Manages transfer mode objects that help users import and export Teamcenter objects and system data.
 Project	Manages projects and project members as a way to control access to data.
 Query Builder	Manages customized search criteria definitions for objects.
 Report Builder	Manages summary, item, and custom report definitions for end users to execute reports.
 Setup Wizard	Manages your company's virtual organization using an input file.
 Subscription Monitor	Manages and tracks user defined subscriptions running in Teamcenter.
 Validation Manager	Manages item revision compliance before they are released.
 Volume Management	Manages volumes and hierarchical storage policies.
 Workflow Designer	Manages process tasks to simulate company workflow.

Instructor Note:

Applications not covered in this course are:

- Audit Manager
- Authorization
- Classification Administration
- Command Suppression
- Subscription Monitor
- Setup Wizard
- Validation Manager
- Volume Management

Business Modeler IDE administration tasks

The Teamcenter administrator performs the following administration tasks using the Business Modeler IDE:

- Configure the data model with new business objects, classes, and properties.
- Extend the POM schema directly for new data storage attributes.
- 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.
- Use *templates* to deploy data model updates to the client.

Business Modeler IDE administration tasks

The Business Modeler IDE is intended for business analysts who are responsible for tailoring Teamcenter for use at their companies. Users of the IDE must have a thorough understanding of how to perform end-user tasks with Teamcenter and have some knowledge of the data model items used in Teamcenter.

Use the IDE to create:

Data model objects	Description
Business objects	The fundamental objects used to model business data. Create business objects to represent product parts, documents, change processes, and so on. Business objects were formerly known as <i>types</i> in Teamcenter Engineering.
Classes	The persistent storage containers of business objects. Each business object has one corresponding class where its property values are stored. Classes are also known as the <i>persistent object model (POM)</i> .
Properties	Item characteristics, such as name, number, description, and so on. Properties are attached to business objects. All children of a business object inherit their parents' properties.
Code generation objects	Objects used for software development, and include data types, libraries, releases, and services. These are used when you want to write C++ code.
Constants	Configuration points within the data model. They provide consistent definitions that can be used throughout the system.
Document management objects	Objects that define how documents are handled in Teamcenter.
Lists of values (LOVs)	Pick lists of values. They are commonly accessed by Teamcenter users from a menu at the end of a data entry box.
Options	Miscellaneous objects such as change objects, units of measure, notes, and so on.
Rules	Directives that govern how objects are handled, including how they are named, what actions can be undertaken on them, and so on. Creating rules is also known as business modeling.

Teamcenter functional requirements (example)

REVIEW NOTE

Issue: DK: Consider creating this divided list in an activity.

Your company requirements for theBusiness Modeler IDE application

1. The ...
2. The ...

Your company requirements for Teamcenter administrative applications:

1. The ...
2. The ...

Instructor Note:

Topic instructor notes.

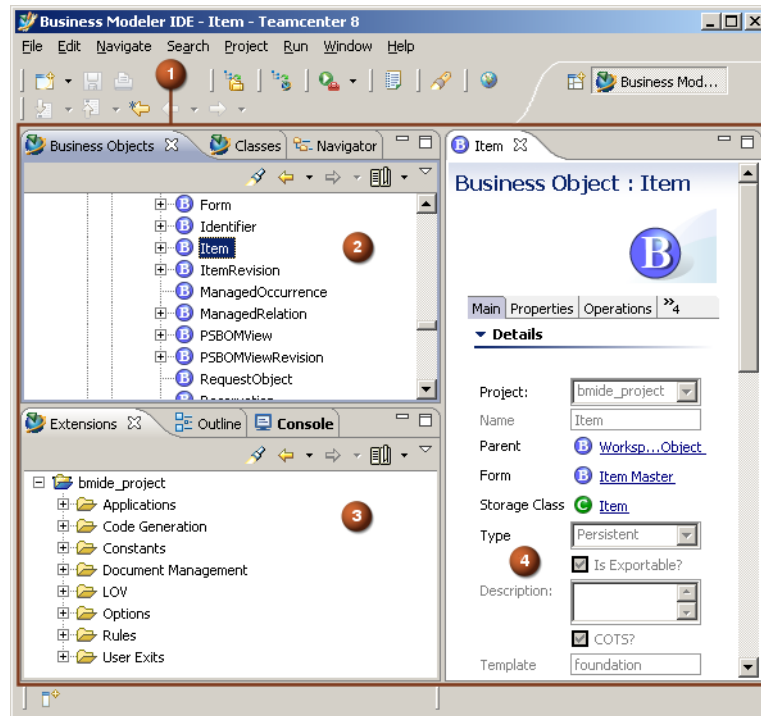
Administrative interfaces

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.

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.

Business Modeler IDE interface

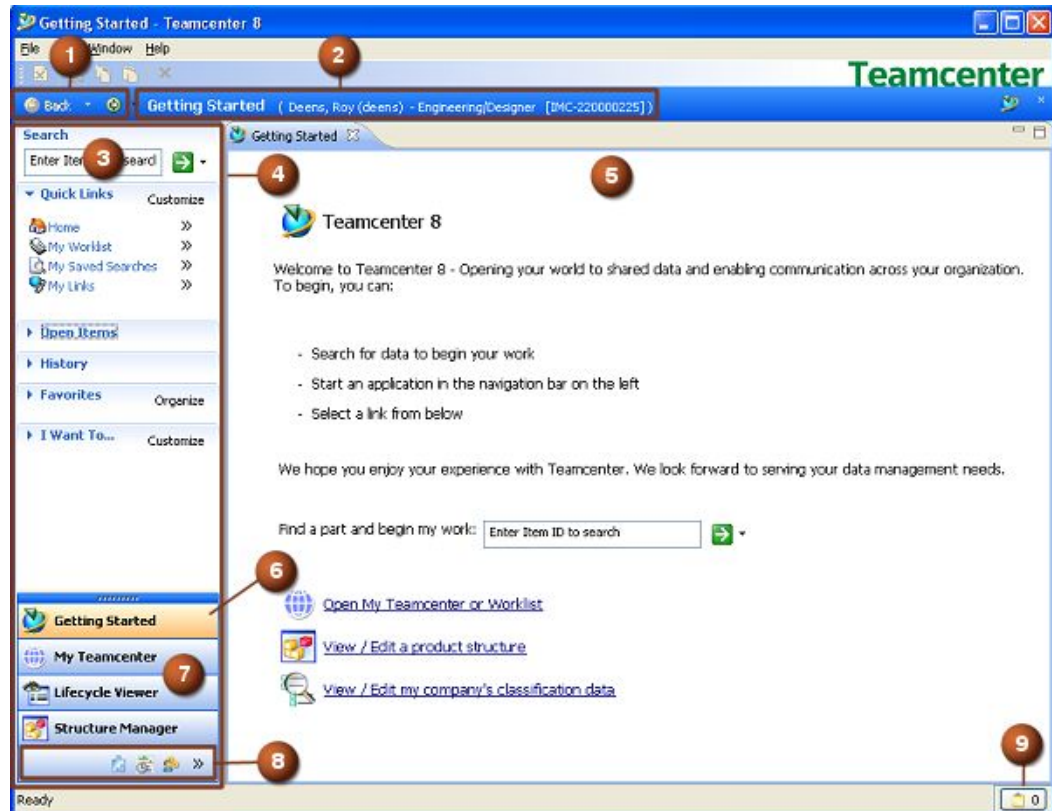


The Business Modeler IDE utilizes the Eclipse user interface, which is composed of perspectives, views, and editors. A *perspective* is an arrangement of views. A *view* is a tabbed window within the UI that provides a view of data. An *editor* is a window that allows you to edit source files. The user can rearrange the user interface in any configuration by dragging and dropping views and editors.

- | | | |
|---|---|---|
| 1 | Business Modeler IDE perspective | Allows you to work with business objects. |
| 2 | Business Objects view | Used for working with business objects, the fundamental objects that model business data. Business objects were formerly known as <i>types</i> in Teamcenter Engineering. |
| 3 | Extensions view | Displays extensions to the data model. |
| 4 | Business Object editor | Enables you to work on different characteristics of the selected business object. |

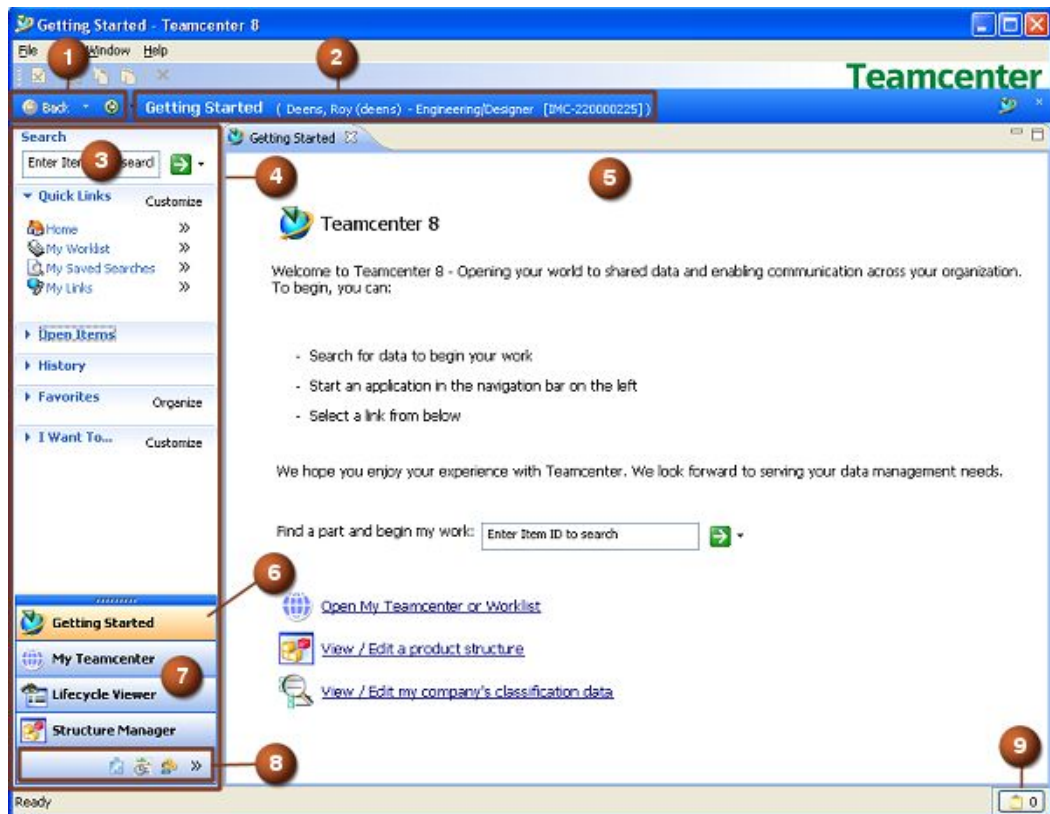
Rich client interface

The rich client interface has a standard menu bar and toolbar with options that vary depending on the currently active application perspective. You can place the cursor over a rich client toolbar button to display a tool tip description.



- | | | |
|---|-------------------------|--|
| 1 | Back and Forward | The Back and Forward buttons allow you to move between loaded Teamcenter applications. The small arrows next to the buttons let you select from the list of currently loaded applications. |
| 2 | Application banner | The application banner shows the name of the active application and lists the current user and role. You can double-click the user and role to display the User Settings dialog box in which you can change your current role if multiple roles are available to your user. |
| 3 | Search box | The Search box provides predefined quick searches using dataset, item ID, item name, keyword search, and advanced search features. |














Rich client interface (continued)



- | | | |
|---|---|--|
| 4 | Navigation pane | The navigation pane provides quick access to the data you use most. |
| 5 | Application pane | The application pane displays the application perspectives that are open in your Teamcenter session. |
| 6 | Getting Started application button | Provides access to the Getting Started application. |
| 7 | Primary applications | Primary application buttons provide access to your most frequently used Teamcenter application perspectives. |
| 8 | Secondary applications | Secondary application buttons provide access to Teamcenter application perspectives you use infrequently. |
| 9 | Clipboard | The clipboard contains references to objects that have been cut or copied from your workspace. |

Rich client administration tasks

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

	Application	Description
	Organization	Configure persons, user, groups, and roles.
	Query Builder	Define queries for end users to find information.
	My Teamcenter	Define preferences to configure the environment.
	Report Builder	Define report definitions for end users to execute reports.
	Access Manager	Define permissions for data.
	Project	Define access permissions to data assigned to projects.
	Workflow Designer	Define process tasks to simulate company workflows.
	Command Suppression	Configure end user menus.
	Subscription Monitor	Track subscriptions running in Teamcenter.
	Classification Administration	Create and manage the classification hierarchy and attribute dictionary.
	PLM XML Export Import Administration	Create transfer mode objects that help users import and export Teamcenter objects and system data.
	Authorization	Control user access to administrative application and utilities.
	Setup Wizard	Create your company's virtual organization using an input file.

Instructor Note:

Tasks not covered in this course are:

- Command Suppression
- Subscription Monitor
- Classification Administration
- Authorization
- Setup Wizard

Teamcenter configuration process

Your company requirements

Iterative processes with the following to implement your requirements:

- Business Modeler IDE administration tasks
- Teamcenter administrative applications

Instructor Note:

Topic instructor notes.

Activities

In the *Introduction to Teamcenter* section, do the following activities:

- Log on to Teamcenter.
- Getting started with Teamcenter administration.
- Create two users.

Review questions

1. Which are true ... ?

Select all that apply.

- A commonly used ...

2. The

- True
- False

3. How do you ... ?

Select all that apply.

- **Show→Children**

Instructor Note:

Activity instructor notes.

Answers to review questions

1. A commonly used

2. False

3. **Show→Children**

Teamcenter Preferences

Primary

REVIEW NOTE

Issue: DK: simply introduce preferences here.

Secondary

Instructor Note:

Topic instructor notes.

This is just an introduction to Preferences and Options since they will be explained in detail separately.

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 8
MP1→Teamcenter 8 MP1→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.

Teamcenter environment variables

Environment variables are also important to execute programs when customizing Teamcenter. The environment variables are defined in the **tc_profilevars.bat** file in the *TC_DATA* folder. The **tc_profilevars.bat** file is shipped with a set of predefined defaults and is further configured automatically during the Teamcenter install.

To run most Teamcenter utility programs you must first propagate the required variables. To set Teamcenter environment in a Windows operating system, choose the following:

Start → All Programs → UGS Teamcenter 2007 → Teamcenter 2007 → * Command Prompt**

Note

The *** represents the node ID and depends on the installation. It is recommended that you create a shortcut for this on your desktop if one has not already been added.

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.

tcCustDev command script

Instead of repeatedly using the following,

**Start → All Programs → UGS Teamcenter 2007 → Teamcenter 2007 →
*** Command Prompt**

experienced programmers will use a script that will create a window with the necessary Teamcenter environment variables. This Command Prompt window can be used to run most Teamcenter utility programs you must first propagate the required variables. For programmers, this window can be used to compile and link your programs with the Teamcenter utilities for compiling and linking.

How to create **tcCustDev.bat** script to set environment variables.

You will need to know where *TC_ROOT*, *TC_DATA* and **vsvars32.bat** are located. Use the class data sheet to find the environment for the classroom.

Create **tcCustDev.bat** with the following:

```
cd /d d:
set TC_ROOT= D:\Tc200713\Local
set TC_DATA= D:\Tc200713\tcdata
set TC_INCLUDE= D:\Tc200713\Root\include
set TC_LIBRARY= D:\Tc200713\Root\lib

call %TC_DATA%\tc_profilevars
call "C:\Program Files\Microsoft Visual Studio 8\Common7\Tools\vsvars32.bat"

set MSDEV_HOME=%VCINSTALLDIR%
set MSVCDir=%VCINSTALLDIR%

cmd
```

list_users

Creates a list of users currently logged on to Teamcenter and the node they are using. This information is useful if database maintenance is necessary and all users currently logged on must be notified.

SYNTAX

list_users **-u**=*user-id* **-p**=*password* **-g**=*group*

ARGUMENTS

-u

Specifies the user ID.

This is generally **infodba** or another user with administration privileges. If this argument is used without a value, the operating system user name is used.

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-id* value to be the password.

-g

Specifies the group associated with the user.

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

clearlocks

SYNTAX

clearlocks [-verbose] [-assert_all_dead -u=user -p=password
-g=group] [-h]

Clears dead process locks from the database. Dead process locks typically occur when a Teamcenter session terminates abnormally. Process locks are set on an object when it is being modified or deleted. If a Teamcenter session does not terminate gracefully (by logging out), these locks can remain in place. There are occasions when such dead process locks must be explicitly removed from the database, and the **clearlocks** utility is used for this purpose.

ARGUMENTS

- **-verbose**

Displays a summary of processes and states (dead, alive, and unknown). Dead processes are cleared and live processes are *not* cleared. Unknown processes are the other processes shown in the display output. The following is an example of a line message (report) produced by **clearlocks -verbose**:

```
Processes: 7, Alive: 1, Dead: 6, Remote: 0, Other: 0
```

Note

Running **clearlocks -verbose** again will show the Dead processes cleared as follows.

```
Processes: 1, Alive: 1, Dead: 0, Remote: 0, Other: 0
```

- **-assert_all_dead -u= -p= -g=**

Asserts that all processes in the database are dead and clears all process locks. This option performs a complete cleanup of the database lock tables and reports on the sessions that were asserted to be dead. If any Multi-Site Collaboration transfer locks are alive and in use, schedule the use of **-assert_all_dead** use at a different time.

Note

To use this argument, you must enter the administrator's user name, password, and group.

- **-h**

Displays help for this utility.

Note

The **clearlocks** utility can be run with active Teamcenter sessions, provided that the **-assert_all_dead** arguments are not used.

Organization setup

The Organization application enables you to maintain your company's organization within Teamcenter.

When you install Teamcenter, the **infodba** system administration account is created. This user has special privileges in Teamcenter and therefore is not recommended for verifying configuration changes to Teamcenter.

For this course the following users have been created for testing:

User	Description
jgordon	DBA user with system administrative privileges.
twhite	Regular user with end user privileges.

Note

More information on the Organization application is covered in the Organization lesson.

make_user

currently logged on to Teamcenter users and ...

SYNTAX

make_user **-u**=*user-id* **-p**=*password* **-g**=*group*

ARGUMENTS

-u

Specifies the user ID.

This is generally **infodba** or another user with administration privileges. If this argument is used without a value, the operating system user name is used.

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-id* value to be the password.

-g

Specifies the group associated with the user.

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

Create DBA and test user

For your Business Modeler IDE development and testing, you should create two users accounts in Teamcenter.

DBA user For the server connection when you deploy your configurations from the Business Modeler IDE to Teamcenter.

Note

The Business Modeler IDE configuration activities do not require a user account, but to deploy the Business Modeler IDE configuration a DBA account is needed in Teamcenter.

Test user For testing the Business Modeler IDE deployments in Teamcenter.

Create DBA and test user

You can use the **make_user** utility to create new users, groups, persons, roles, and volumes.

The **make_user** utility allows you to:

- Create your organization from a command line or script.
- Modify properties of existing user, group, and role objects.

The **make_user** utility supports batch mode processing using an input file. You can run the **make_user** utility script multiple times.

Warning

The **make_user** utility cannot be used to delete the organization objects.

Use the **make_user** utility script to:

- Load your entire organization with one command.
- Back up your organization.
- Populate your test and production environments.

Teamcenter environment for utilities

To run most Teamcenter utility programs you must first propagate the required environment variables to a command window. To set Teamcenter environment variables in a Windows operating system, choose the following:

Start → All Programs → Teamcenter 8 → * Command Prompt**

Note

The *** represents the node ID and depends on the installation.

The environment variables are defined in the **tc_profilevars.bat** file in the *TC_DATA* folder. The **tc_profilevars.bat** file is shipped with a set of predefined defaults and is further configured automatically during the Teamcenter install.

The `make_user` utility

The **-file** argument specifies that the input file is read to create users or to modify existing users, groups and roles after other arguments are processed.

Example

```
make_user -u=smith-p=smith -g=dba -file=org.txt
```

Each record in the file contains the following information:

```
person|user|password|group|role
```

Each field is delimited by the (|) character.

These users will be created for configuring and testing the Business Modeler IDE.

```
Gordon, Jack|jgordon|jgordon|dba|DBA
```

```
White, Tom|twhite|twhite|Engineering|Designer
```

Teamcenter architecture overview

Two architectures are available with Teamcenter:

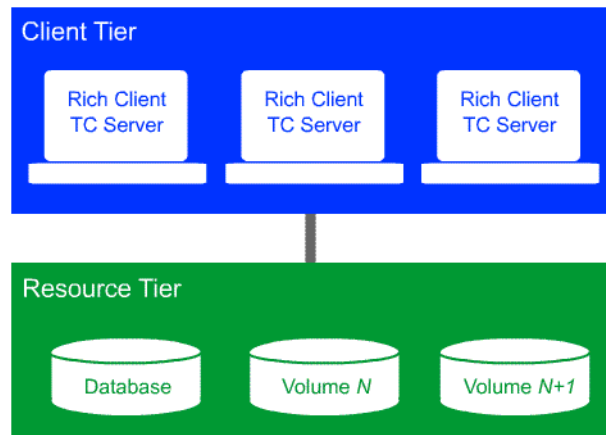
- **Two tier**
- **Four tier**

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

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

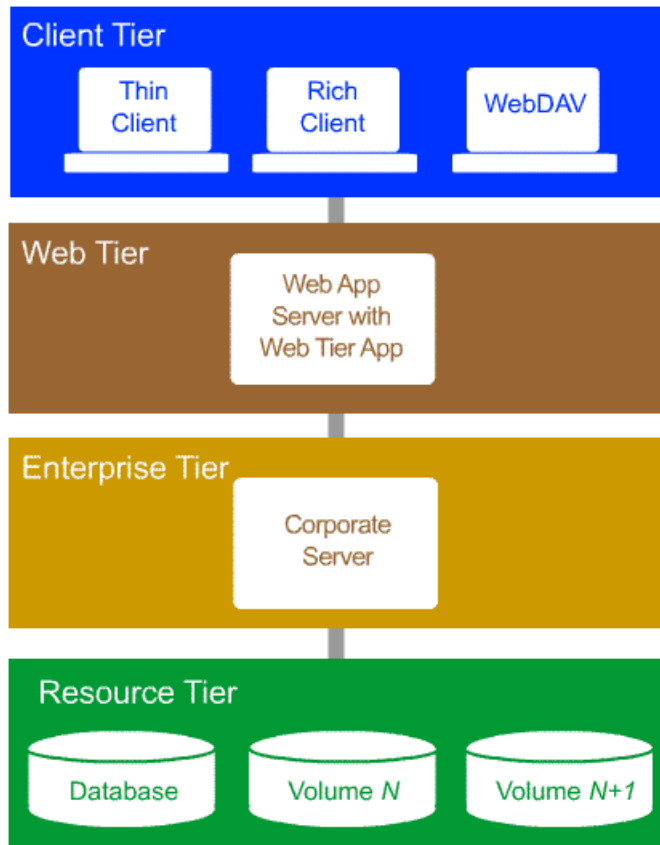
Prepare the Teamcenter system environment for command line interface tools.

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's 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.

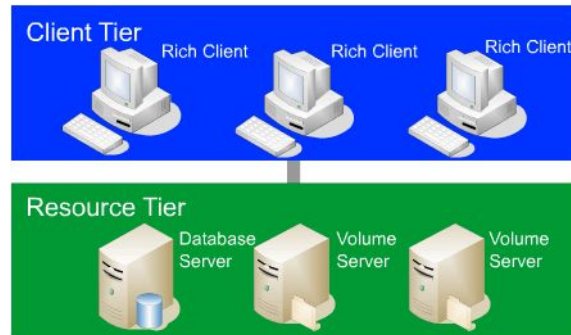
The enterprise tier sits on the Teamcenter corporate server. It is composed of:

- 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:

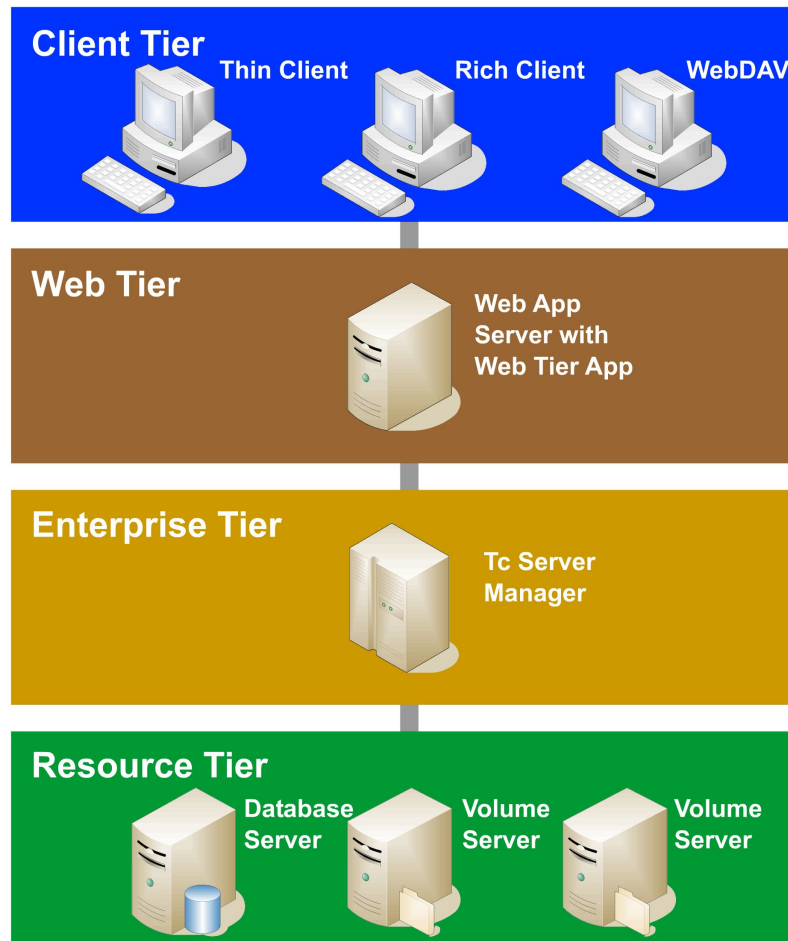
- Database server and database
- Volumes
- File servers

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. The *resource tier* 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's 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 is composed of:

- A pool of server processes managed by a server manager (four-tier architecture only).
- Transient volumes.
- The *resource tier* stores persistent metadata and files managed by Teamcenter.

The resource tier contains:

- Database server and database.
- Standard volumes.
- File servers for shared configuration and binary executables.

Teamcenter Folders

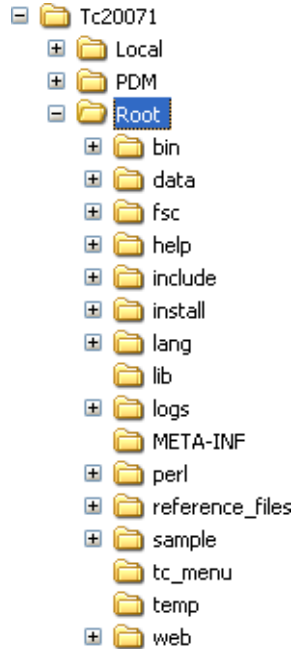
Before beginning any custom programming, you should understand the Teamcenter folder structure and environment variables. These contain folders and files that will be important for the custom ITK programs.

Each of the following will be discussed.

- **TC_ROOT** contains the base installation files.
- **TC_DATA** contains **tc_profilevars.bat** to set environment variables.
- **TC_ROOT\sample** contains sample files (many for programming).

Teamcenter Root folder

The following is an example of some of the folders under the Teamcenter root folder structure **TC_ROOT**.



bin	Teamcenter executables
data	template data used for creating <i>TC_DATA</i>
include	header files used for Teamcenter Integration Toolkit (ITK) programming
install	installation utilities
lang	localization files for different languages
lib	all archive and shared libraries
sample	sample source code and scripts for customizing Teamcenter
web	files for Teamcenter Web

Instructor Note:

bin – This week we talk some about utilities. This is where utilities are located and other .exe's are located.

data – Whenever we take a brand new Teamcenter database, we'll eventually work toward having a production, training, development and even test databases. If you want some standards as what you want in these you can stick that in a the data folder and whenever you create a new database you'll push that into it.

lib – This is where DLLs are located. This week we update one DLL.

help – actually moved in version 7. It sets in the web based help.

Teamcenter data – every time you create a database you'll get an Teamcenter data folder that contains information like what the database looks like, the server, customizations, etc.

include – all the header files for the api's

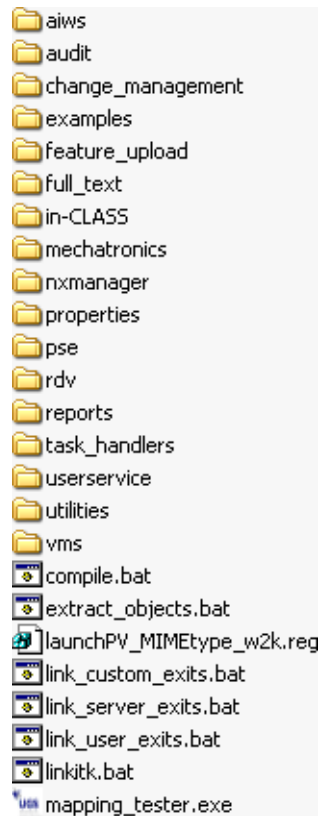
lang – localization files reside here

sample – best friend for a programmer

web – web interface all the source files. If you make changes to the web interface you'll do it here.

Teamcenter Sample folder (ITK files)

Teamcenter sample folder is located at **TC_ROOT\sample**. The sample folder contains sample files to compile and link ITK customization programs.



The following files are noted to compile and link your customization programs.

compile used to compile Teamcenter server programs

linkitk used to link Teamcenter batch programs

link_user_exits used to link Teamcenter user exit programs (and *replace* libuser_exits inside **TC_ROOT\bin**).

link_custom_exits used to link customer user exit programs for Teamcenter (that are *added* to **TC_ROOT\bin**).

link_server_exits used to link Teamcenter server exit programs (which handshake with Rich Client Java programs).

Teamcenter Sample folder

The following folders are noted since they are more useful for typical implementations.

examples	contains many Teamcenter server programs samples
utilities	contains the code for a few Teamcenter utilities
pse	contains server programs for PSE operations
properties	contains the code for Teamcenter server user exit programs
task_handlers	contains many programs and functions useful for Teamcenter Workflow

Summary

The following topics were taught in this lesson:

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

Lesson

2 *Using Teamcenter Options and Utilities*

Purpose

The purpose of this lesson is to manage Teamcenter options and preferences.

Objectives

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

- View and locate options.
- View and locate preferences.
- Modify and create preferences.
- Identify key Teamcenter preferences.
- Import and export preferences.

Help topics

Additional information for this lesson can be found in:

- [*Getting Started with Teamcenter*](#) (see *Managing options and preferences*)
- [*Preferences and Environment Variables Reference*](#)

Instructor Note:

Lesson instructor notes.

Introduction to preferences

Preferences are variables stored in the Teamcenter database. They are read during Teamcenter application usage. Each application may have associated preferences.

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.

Preferences allow users to configure many aspects of a session, such as:

- Column display settings
- Item default paste relation
- Dataset default tools

Preferences allow administrators to configure many aspects of a session, such as:

- Password requirements
- Dataset version purge limit
- Application logging

Note

Only a administrator can view and edit all preferences in Teamcenter.

The **tc_preferences.xml** file is read during system startup for loading the site preferences.

Preferences can also be manually modified in a copy of the **tc_preferences.xml** file. After modifying, use the **preferences_manager** utility to import the file.

Installation and upgrades

Teamcenter ships all preference data in the **tc_preferences.xml** file, stored in the *TC_DATA* directory. Teamcenter reads most of the preference data from this directory. Any custom *site* preferences created at your site are stored in the Teamcenter database.

Note

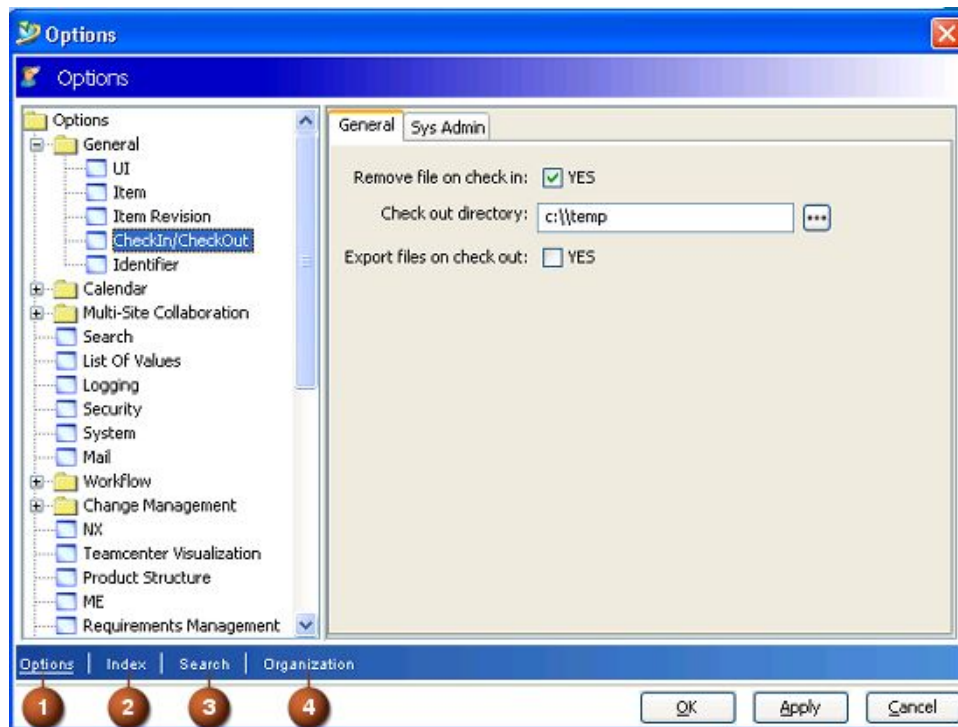
Previous to Teamcenter 2007, all preferences were stored in the Teamcenter database. Out-of-the-box preferences are now stored in the *TC_DATA* directory. Only custom site preferences are stored in the Teamcenter database.

You can also use the **preferences_manager** utility during upgrades to convert legacy preference files to XML format and to import and export preferences to and from the database. This utility parses the legacy site preference file to XML format.

Preference interface

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. For information about this dialog box, see *Getting Started with Teamcenter*.

Preferences can also be manually modified in a copy of the **tc_preferences.xml** file. After modifying, use the **preferences_manager** utility to import the file.



- 1 **Options link** Displays the **Options** pane where you work with a limited set of preferences in the form of options.
- 2 **Index link** Displays the **Preferences** pane where you search for preferences based on a preference name.
- 3 **Search link** Displays the **Search Options** pane where you search for preferences based on keywords.
- 4 **Organization link** Displays the **Organization** pane where you view the preferences applicable to group, role, and user.

Working with Options

You can work with a limited set of preferences. Use the **Options** pane of the **Options** dialog box to configure preferences.

Option details display when an option is selected from the **Options** tree.

Option details **optionally** have tabs of data to separate option configurations.

Viewing and setting options

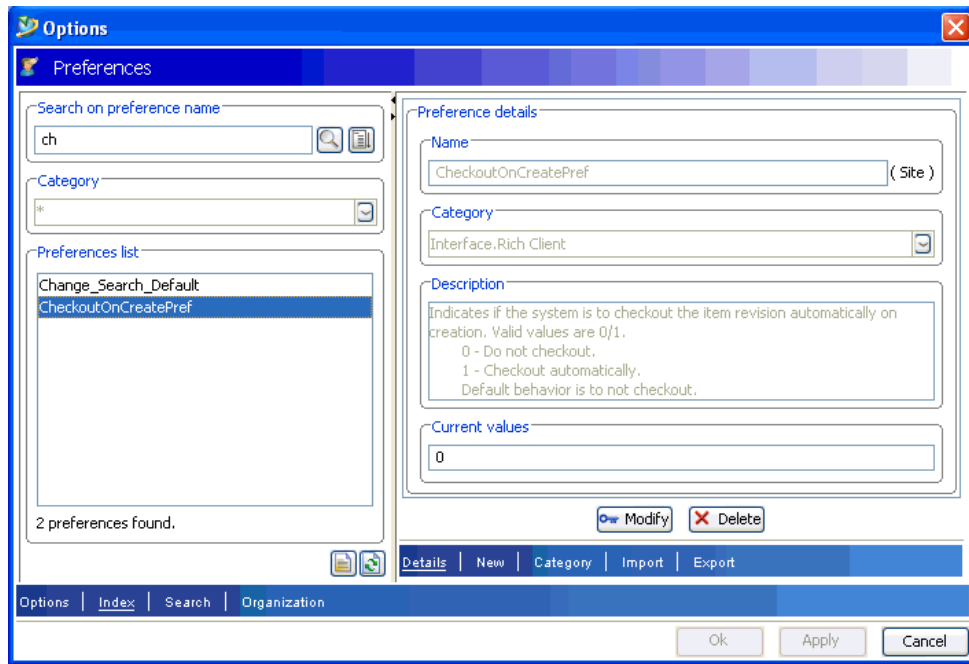
Options are presented by category and display a limited set of preferences. Options display list of values, check boxes, and radio boxes for preference configuration. Some option details have tabbed panes to display additional configurations selections.

When a user selects an option from the **Options** tree, only the user's group and role option configurations are available. Other option configurations and tab panes are unavailable. Only a Teamcenter administrator can view and edit all options in the system.


After making option configuration changes, click **Ok** or **Apply** to save the changes.

Working with Preferences

The **Index** link displays the **Preferences** pane of the **Options** dialog box. The **Preferences** pane enables you to work directly with the entire preference set.



From the **Preferences** pane, you can

- Search for preferences in the database.
- Modify preference values.
- Delete preferences.
- Add preferences to the database.
- Create new categories for organizing preferences.
- Import and export preferences.
- Report  customized preferences.

Viewing and setting preferences

Each user can view and modify certain preferences, depending on their role and group membership.

For example, a user logged on as **Engineering/Designer** could modify all **Engineering** group preferences and all **Designer** role preferences. The same user could not modify other role preferences, such as **Reviewer**, **Manager**, or **Tester**. Users belonging to multiple groups and roles must change their group or role setting to modify preferences affecting other groups or roles.

REVIEW NOTE

Issue: LR: This topic was not on the shared list, but I really like it and I want to use it in the course material. I didn't change the content, just made the topic more presentation friendly. I broke this down into bullets and ordered by user, group administrator and admin. Hopefully it will still work for tdocs.

- When a user browses or searches for preferences, only those group and role preferences available to the user are visible.
- When a user creates a new preference, the currently effective group and role determines the scope of the new preference.
- A group administrator can create or modify group preferences that affect all members of the group.
- A Teamcenter user logged on with the necessary privileges can add or modify site preferences that affect all users, groups, and roles.

REVIEW NOTE

Issue: LR: Can I replace “A Teamcenter user logged on with the necessary privileges” with “An administrator”? Isn't that what you are saying here?

An administrator can generate a report that lists all preferences that are customized at the site or for a selected group, role, or user.

REVIEW NOTE

Issue: LR: Should we link to the reporting topic here?

Only a Teamcenter administrator can view and edit all preferences in the system.

REVIEW NOTE

Issue: LR: Do we need to specify “Teamcenter” here? We don’t always do it on this page so it might confuse the user that there is a difference between a “Teamcenter administrator” and “administrator”.

Preference categorization

Preferences are categorized according to the technical area of Teamcenter they affect. 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.

```
ProductStructure.PSE.Behavior
```

Use preference categories to:

- Limit preference search to specific categories. This makes the search faster.
- Add new preferences to the appropriate category. This helps with future searches.
- Create new categories or subcategories that are appropriate for your business. Only administrators can do this.

Preference scope

When working with preferences, you must take each preference's scope into consideration. 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 scope of a preference determines:

- User access to the preference. A user's current group/role logon determines which existing preferences he can modify, and the scope assigned to any newly created preference.
- **Precedence** of its behavior when there are multiple incidences of the same preference. For example, when one instance of the preference for displaying the NX button in the Teamcenter toolbar is set by an administrator as a site preference, and another instance of the same preference is set by an individual user as a user preference, the preference's scope determines which instance is given precedence.

Preference scope is displayed in the **Preference details** box of the **Options** dialog box. When an existing preference is selected, its scope displays to the right of the **Name** box.



Preference scope is set when a new preference is created or when a preference is imported.

REVIEW NOTE

Issue: LR: I couldn't find any descriptions for these in either the training or docs so I made some up. I think this information is imported to say and not just assume the audience will figure it out. I also created definitions for Overlay and All that I'm just guessing about; need to get these items verified by development. Also, I still can't see how these scopes can be set.

The following scopes are available for preferences:

- **User** scope applies the preference to the user who created the preference.
- **Role** scope applies the preference to the user whose current role matches the role for the preference.
- **Group** scope applies the preference to the user whose current group matches the group for the preference.

- **Overlay** scope applies the preference to all users in the site, but takes precedence over the site scope. Use this scope when you want to override a site preference, but keep its original setting in tack. This scope can be set by adding a preference to the overlay file ???.
- **Site** scope applies the preference to the user whose working in the site that matches the site for the preference.
- **All** scope applies the preference to all users in Teamcenter. This scope can be set ???.

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 logon. 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 is shown in order of the precedence hierarchy used to resolve setting conflicts in multiple locations, with 1 being the highest. For example, a user preference take precedence over the same preference defined for a higher scope.

1. **User** preference precedence order: **User, Role, Group, Overlay, Site**.
2. **Role** preference precedence order: **Role, Group, Overlay, Site**.
If the user changes role settings during a session, the appropriate role preferences are loaded.
3. **Group** preference precedence order: **Group, Overlay, Site**.
If the user changes group settings during a session, the appropriate group preferences are loaded.
4. **Site** preference precedence order: **Overlay, Site**.
5. **All** preference precedence order: environment variables, **User, Role, Group, Overlay, Site**.

Keeping the rules of *user access* and *preference precedence* in mind, consider the following example:

A site administrator uses the search feature in the **Options** dialog box to search for the **TC_inbox_interval** preference and to set it at **20**, causing the system to check inboxes for new tasks every 20 minutes. A user with very few tasks creates another instance of this preference and sets it at **0**, causing the system to check his inbox for new tasks only once per session, at logon.

The site administrator created the preference while logged on with a group/role of **dba/dba**. The user created the preference while logged on with a group/role of **accounting/auditor**.

When the auditor logs on, the system automatically checks his user preferences. Finding an instance of the **TC_inbox_interval** preference, the system performs the behavior directed by the preference's value, and checks the auditor's inbox for new tasks only at logon. The system then continues to check the other scopes in precedence order. When it reaches the overlay preferences, it once again finds an instance of the **TC_inbox_interval** preference. Because a user preference setting takes precedence over an

overlay setting, the system continues to perform the behavior directed by the user's setting.

Key Teamcenter preferences

Category	Preference	Default value	Description
General	QRY_dataset_display_option	none	Determines whether the latest version (1) or all versions (2) of a dataset object are displayed when query results are returned. Note This preference must be defined in Teamcenter.
Interface.Rich Client	AE_dataset_default_keep_limit	3	Determines the number of old dataset versions to store in the database. When this number is exceeded, the oldest dataset version is deleted. REVIEW NOTE Issue: LR: <i>if it is not referenced</i> was added to this description. This does not correspond to the documentation. Need a use case to provide to tdocs if this behavior is valid.
	QRYColumnsShownPref	Valid Teamcenter query names	Determines the favorite queries available for selection from the list of System Defined Searches .

Category	Preference	Default value	Description
	TC_auto_login	TRUE	Enables (TRUE) or suppresses (FALSE) the automatic logon feature for the entire site. Auto logon uses operating system user names and passwords to log on to Teamcenter.

Note

The automatic logon feature is available only in a two-tier Teamcenter environment.

Product
Structure.
PSE.Display

Incremental_Change_
Management

false

Enables (true) or disables (false) incremental change functionality. An incremental change groups multiple structure changes to a given component.

Product
Structure.
PSE.Behavior

PSE_default_view_
type

view

Sets the default BOM view used when opening Structure Manager or pasting in an assembly when more than one BOM view is used at the site.

REVIEW NOTE

Issue: LR: this does not match up with tdocs. I will issue a PR to get this clarified.

Category	Preference	Default value	Description
	TC_config_rule_name	Latest Working	Sets default revision rule that is used when opening and printing BOM views or BOM view revisions. REVIEW NOTE Issue: LR: this does not match up with tdocs. I will issue a PR to get this clarified.
Classification	ICS_classifiable_types	Item ItemRevision	Specifies the list of Teamcenter types that can be classified. The two default values, Item and ItemRevision , are considered a set. Therefore, if the item is classified the item revision cannot be classified, but it inherits the classification record from the item. This also works in the reverse; if the item revision is classified, the item cannot be classified, but inherits the classification record.

A complete list of preferences can be found in the *Preferences and Environment Variables Reference*.

Search by keyword

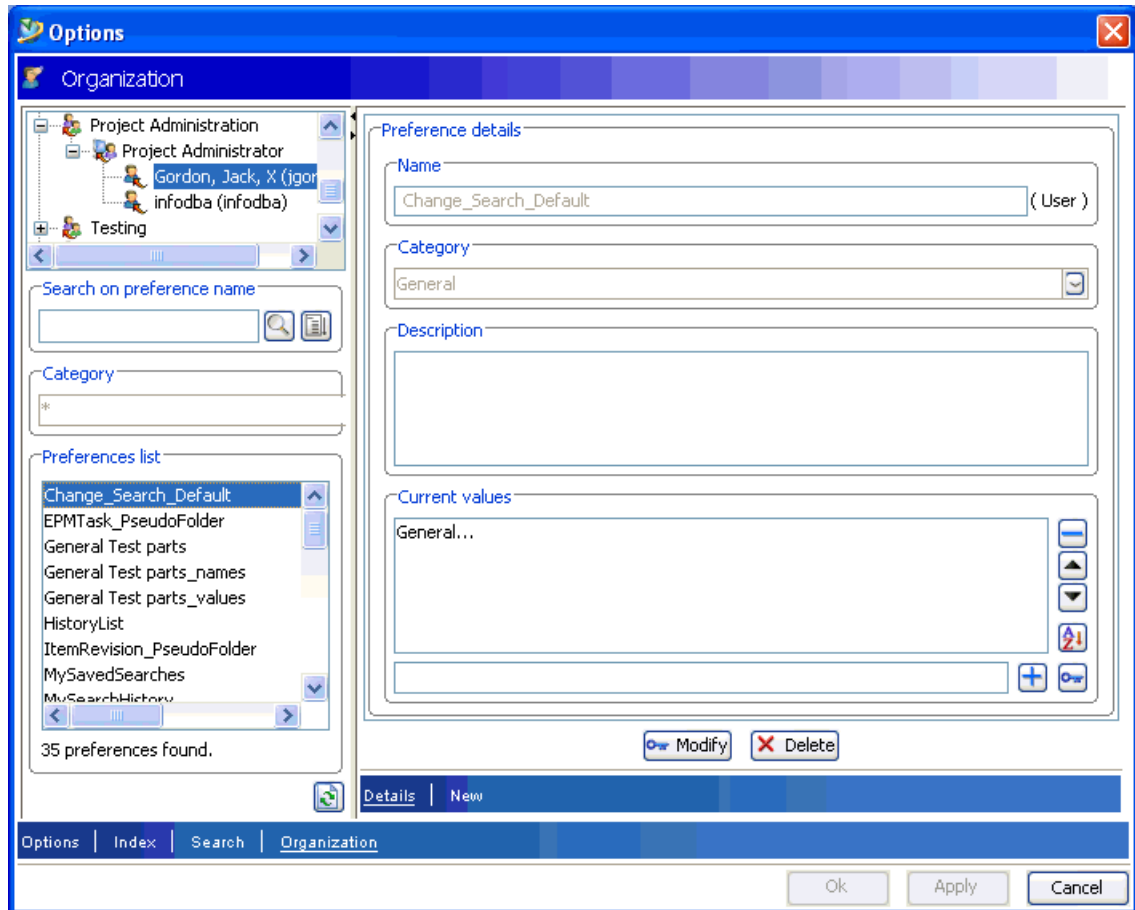
The **Search** link displays the **Search Options** pane of the **Options** dialog box. The **Search Options** pane enables you to work directly with the entire preference set. The **Search Options** pane operates similar to the **Preferences** pane, but with additional search features.

From the **Search Options** pane, you can

- Search for preferences based on keywords.
- Select search in options of keys, description, and values.
- Select match options for case sensitivity and entire word.
- Specify a wildcard character.

Search by organization

The **Organization** link displays the **Organization** pane of the **Options** dialog box. The **Organization** pane enables you to view preferences based on groups, roles and users in the organization.

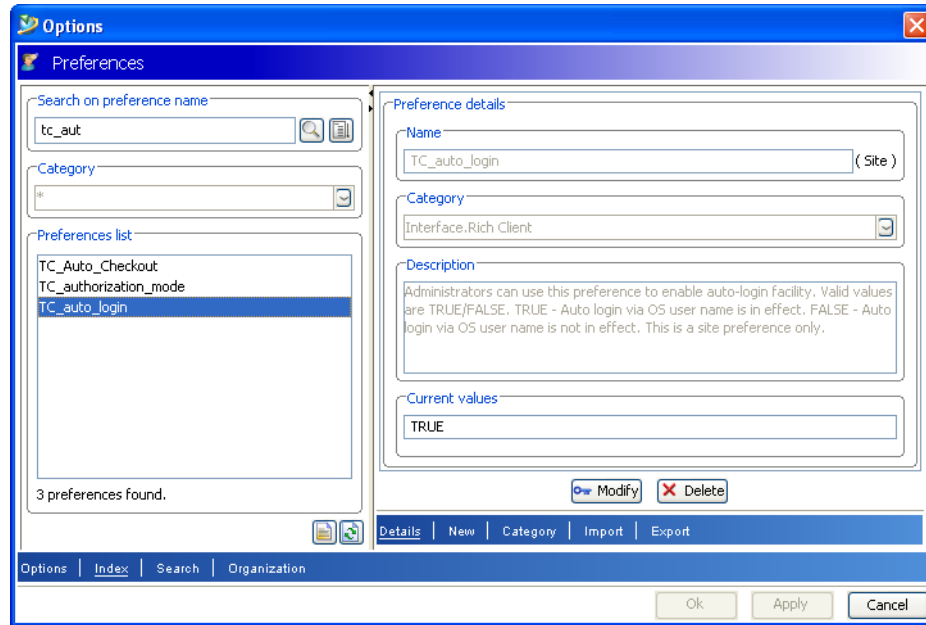


From the **Organization** pane:

- Users can see the preference applicable to their site, group, role 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 users.

Review of Teamcenter Preferences

Preferences are special variables stored in the Teamcenter database. They are read during Teamcenter application usage. Each application may have associated preferences.



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.

The **Preferences** pane enables you to work folder with the entire preference set.

From the **Preferences** pane, you can

- Search for preferences in the database.
- Modify preference values.
- Delete preferences.
- Add preferences to the database.

Viewing and setting preferences

The **Preference Details** are used to view, modify, or create a new preference. In this example, the scope is set to **Site**, and the preference has **Multiple Values**.


The screenshot shows the 'Preference details' dialog box. It has four main sections:

- Name:** A text field containing 'QRYColumnsShownPref' and a dropdown menu set to '(Site)'.
- Category:** A dropdown menu showing 'Interface.Rich Client'.
- Description:** A text area containing the text: 'Specifies the list of default (Favorite) saved queries used for both Teamcenter Rich/Thin clients. The values of this preference should be the localized query names.'
- Current values:** A list box containing the following items: 'General...', 'Item...', 'Item Revision...', 'Remote...', and 'Checked-Out Dataset...'. To the right of the list box are several control buttons: a minus sign, up and down arrows, a plus sign, and a button with a red 'X'.

Each user can view and modify certain preferences, depending on their role and group membership.

Only an administrator can view and edit all preferences in Teamcenter.

Note

For a preference with **Multiple Values** set to **True**, in the box below the **Current values** pane, type values and click  (or press **Enter**) to add each value to the **Current values** list

Preferences Index and Search

The **Index** link displays the **Preferences** pane of the **Options** dialog box. With **Index** you can enter the beginning characters to locate the preference.

The **Search** link displays the **Search Options** pane of the **Options** dialog box. The **Search Options** pane enables you to work directly with the entire preference set.

The **Search Options** pane operates similar to the **Preferences** pane, but with additional search features.

From the **Search Options** pane, you can

- Search for preferences based on keywords.
- Select search in options of keys, description, and values.
- Select match options for case sensitivity and entire word.
- Specify a wildcard character.

Preference scope

Preference scope is displayed in the **Preference details** box of the **Options** dialog box. When an existing preference is selected, its scope displays to the right of the **Name** box.



Preference scope *is set* when a new preference is created or when a preference is imported.

Preference scope determines the order in which the system searches for the preference and its values.

Possible scope values include: **User**, **Role**, **Group**, **Overlay**, **Site**, and **All**.

The following list is shown in order of the precedence hierarchy used to resolve setting conflicts in multiple locations, with **1** being the highest. For example, a user preference take precedence over the same preference defined for a higher scope.

1. **User** preference precedence order: **User**, **Role**, **Group**, **Overlay**, **Site**.

2. **Role** preference precedence order: **Role**, **Group**, **Overlay**, **Site**.

If the user changes role settings during a session, the appropriate role preferences are loaded.

3. **Group** preference precedence order: **Group**, **Overlay**, **Site**.

If the user changes group settings during a session, the appropriate group preferences are loaded.

4. **Site** preference precedence order: **Overlay**, **Site**.

5. **All** preference precedence order: environment variables, **User**, **Role**, **Group**, **Overlay**, **Site**.

Teamcenter customization preferences

At times during this class you will set a Teamcenter preference that is used with the specific area of customization.

The Teamcenter preference that will be viewed now is **Mail_OSMail_activated**, which is set to **true** by default.

Mail_OSMail_activated

DESCRIPTION

Determines whether operating system email can be sent from Teamcenter. When enabled, operating system email is additionally sent when a xxx mail envelope is sent.

VALID VALUES

true

Operating system email enabled (sent by Teamcenter).

false

Operating system email disabled (by Teamcenter).

DEFAULT VALUES

true

SCOPE

Site preference.

Activities

In the *Preference management* section, do the following activities:

- Modify a site preference.
- Create a user preference.
- Create a group preference.

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.

- 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.

- **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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Changes that have been made to site preferences at your site

Custom preferences

Help answer: You can generate reports that list the changes that have been 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.

2. To log on to the system as a user of the group

Help answer: The **Group** preference report lists all preferences stored for your current logged in group.

3. **Site, Group, Role, and User** preference reports.

Help answer: The **Site** lists the custom preferences that have been created at the site. The **Group, Role, and User** preference reports list all preferences stored for the current logged in group, role, and user ID.

4. True

Help answer: You can export the preference report in XML file format.

Generating preference reports

You can generate reports that list the changes that have been 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. Use this report to determine which 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 **Index** 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.

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.

Use the **Import** link on the **Options** dialog box to import preferences and their values to the database. You can specify handling options to control the import. Use the **Export** link on the **Options** dialog box to 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.

Import preferences in to the database

1. Choose **Edit→Options**.

Teamcenter displays the **Options** dialog box.

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.

Teamcenter displays the **Import Preferences** 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. (Optional) Choose the category of preferences to be imported. If you choose a category, all preferences in the category are imported and all preferences in all other categories are ignored. If you do not choose a category, all preferences in all categories are imported.

7. 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 category 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.

8. 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**

9. Click **Import**.

The preferences and their values are imported in to the database according to the specified handling options.

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. (Optional) Choose the category of preferences to be exported. If you choose a category, all preferences in the category are exported and all preferences in all other categories are ignored. If you do not choose a category, all preferences in all categories are exported.
7. (Optional) Select the **Open on Export** box to open the XML file when the preference export operation is complete.
8. Click **Export**.

Teamcenter exports the selected preferences to the specified XML file.

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 2-1. 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 options that can be used with -mode=generatexml option.
<code>preferences_manager -u=user-id -p=password -g=dba -mode=generatexml -context=Teamcenter -file=legacy-preference-file -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 -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.
<code>preferences_manager -u=infodba -p=password -g=dba -mode=import -preference=TestPreference -scope=SITE -values=Val1,Val2,Val3 -action=OVERRIDE</code>	Imports a preference and override the values in the database with the values specified for the preference on the command line.

Table 2-1. Preferences utility examples

Command line	Results
<pre>preferences_manager -u=user-id -p=password -g=group-name -mode=export -scope=GROUP -out_file=c:\temp\group_pref.xml</pre>	Exports all group preferences in the database for the user's group when the utility is run for that specific group.
<pre>preferences_manager -u=user-id -p=password -g=group-name -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=user-id -p=password -g=dba -mode=migrate -dir=some-directory</pre>	<p>Migrates the legacy site, user, role, and group preferences files. If the required -dir directory contains the legacy site preference file (.iman_env), all preferences within this file are migrated to the database.</p> <p>If the TC_GROUP_PFILE, TC_ROLE_PFILE and TC_USER_PFILE preferences are set, then the legacy files for group, role, and user are obtained up from these directories.</p> <p>If no preferences are specified and the directory supplied contains the gpfiles, rpfiles, and upfiles subdirectories, then these subdirectories are used for migration.</p>

For information, see *Configuration utilities* in the *Utilities Reference*.

Activities

In the *Preference management* section, do the following activities:

- Create a modified site preference report.
- Create a modified group preference report.
- Create custom categories, subcategories, and preferences.
- Export custom preferences.
- Use the `preferences_manager` utility to import new categorized preferences.

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.

- **Edit® User Setting**
- **View® Organization**
- **Edit® Options**

4. What does the **AE_dataset_default_keep_limit** preference do?

Select one.

- 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.

- **Site, Group, Role, User**
- **Group, Role, User, Site**
- **User, Group, Role, Site**
- **User, Role, Group, Site**

7. You only can import and export preferences using the **preference_manager** utility.

- True
- False

8. What is a the **Selective** import mode?

Select one.

- 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

9. 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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Special environment variables stored in the Teamcenter database

Help answer: Preferences are special environment variables stored in the Teamcenter database and read whenever a preference value is set or modified.

XML strings in specific tables in the database

Help answer: Preferences are stored as XML strings in specific tables in the database, the scope determines in which table a preference is stored.

2. True

Help answer: Preferences allow you to configure many aspects of a session such as user logon names and the columns displayed by default in a properties table. Basic system behavior such as business rules and security access are also determined by preferences.

3. **Edit® Options**

Help answer: Choose **Edit® Options** to open the **Options** dialog box that contains the **Options** pane, the **Index** pane, the **Search** pane and the **Organization** pane to search, view, modify, create, and import/export Teamcenter preferences.

4. Sets the number of old dataset versions to store in database

Help answer: 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.

5. True

Help answer: Preferences are categorized according to the technical area of Teamcenter they affect and may contain subcategories to allow easier location and identification of a specific preference.

6. **User, Role, Group, Site**

Help answer: When scope is **User**, the system searches through preference data in the following order: user, role, group, site.

7. False

Help answer: Preferences can be imported and exported in two ways, using the **Import** and **Export** links, or using the **preferences_manager** utility.

8. A mode that compares the values of the preferences in the XML file with the preference values in the database for the specified scope

Help answer: The **selective** import mode 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.

9. To import and export preferences to and from the database

Help answer: The **preference_manager** utility can be used to import and export preferences to and from the database and to migrate the legacy **upfiles**, **rpfiles** and **gpfiles** preference files to the database.

Summary

The following topics were taught in this lesson:

- View and locate preferences.
- Modify and create preferences.
- Create categories for organizing your preferences.
- Preference scope and precedence.
- Generate preference reports.
- Import and export preferences.

Instructor Note:

Summary instructor notes.

Lesson

3 *The Business Modeler IDE fundamentals*

Purpose

The purpose of this lesson is to introduce using 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 and perform the basic Business Modeler IDE process.
- Create a custom Item business object and deploy it to the server.

Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#)

Instructor Note:

Lesson instructor notes.

Create a Business Modeler IDE template project

1. To create a new project, choose **File→New→Project**.
2. Expand **Business Model IDE** to select the **New Business Model IDE Template Project** as the wizard to start.
3. Follow the wizard through the creation process where you define:
 - The **Project name**.
 - The **Location** for your project files.
 - The **Teamcenter templates folder** location where the dependent templates reside.
 - The **Template name** and **Template display name**, which default to names derived from the **Project name**.
4. Choose the dependent **Templates**. At a minimum, choose the **Foundation** template.

Note

The selected **Templates** should match the **Templates** in your Teamcenter installation.

The new project appears in your perspective.

Note

Open the perspective where the project will be created.

For **Project name**, do not use spaces. For example, **CCC_DEV**.

Use the default location or specify a location for your project files. The destination directory must be named the same as the project.

Instructor Note:

You should demo creating a project. Show them the project directory that is created and the location of the template files.

The detail procedure steps for creating a project can be found in the BMIDE guide.

Business Modeler IDE extension files

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

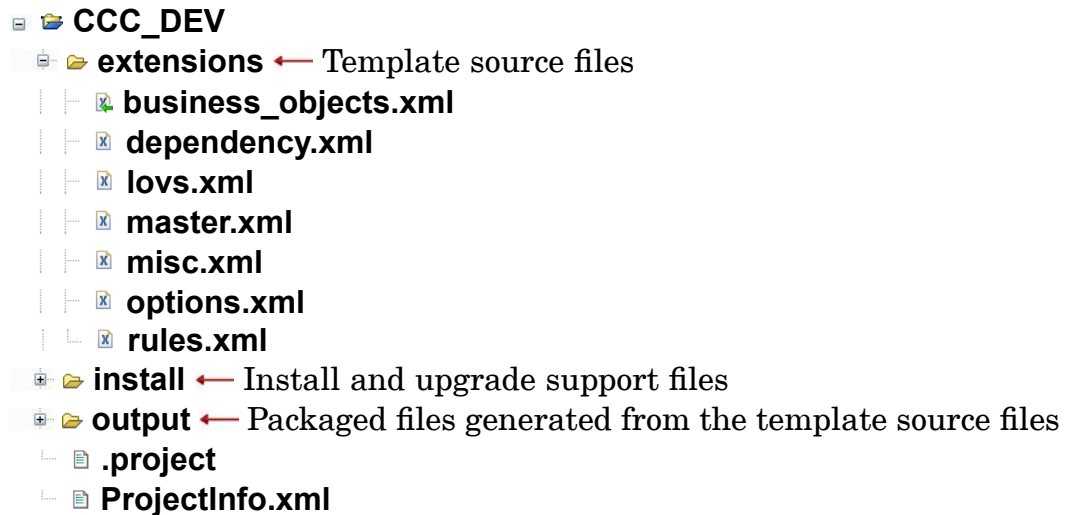
Additional extension files can be added, but you would typically organize your extensions in one of the following areas:

- Business objects
- LOVs
- Miscellaneous
- Options
- Rules



Files in a template project

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



Extensions directory

The **extensions** directory contains three types of files

- The template source files are XML files that contain the template extensions to the data model and business rules. Multiple files are available or 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

The **install** directory contains all the files that support install and upgrade of the template within TEM.

- The feature file presents your template as a choice in TEM for install and upgrade.
- Bundle files provide localizable strings for your TEM feature file.
- The install script contains a set of ordered commands that 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 **output** directory is a directory where files are generated from the template source code.

- 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. Without this file, the Business Modeler IDE cannot 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.
- The **ProjectInfo.xml** file contains additional project information, such as settings used for building and compiling. To see the project information that is placed in this file, right-click the project in the **Navigator** view and choose **Properties**, and then choose **Teamcenter** in the left pane of the **Properties** dialog box.

Instructor Note:

Topic instructor notes.

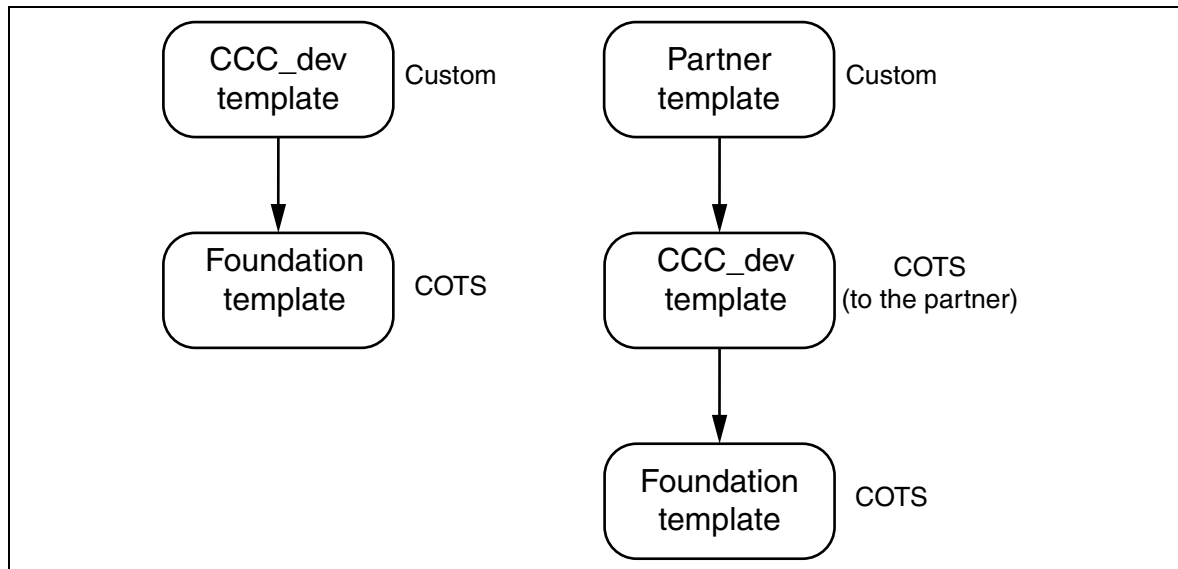
Note

Only the list of files will show in presentation mode.

The details of each file will only show in the student guide.

Understanding custom versus COTS templates

Custom data model objects are those objects you create and store in a custom template. *COTS* (commercial-off-the-shelf) data model objects are the objects that your custom data model objects are dependent on. Therefore, custom templates are dependent on COTS templates. You can change or delete only the custom objects in the data model.



Custom versus COTS template example

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.

Note

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 the Business Modeler IDE

The Business Modeler IDE (Integrated Development Environment) is a tool for configuring and extending the data model of your Teamcenter installation. The data model objects define the objects and rules used in Teamcenter.

Administrators and business analysts use the IDE to:

- Create new data model objects such as:
 - Business objects (for example, items and datasets)
 - Properties
 - Lists of values (LOVs)
 - Rules
- Monitor logging using the Digital Dashboard
- Migrate data using the Mapping Designer

The Business Modeler IDE is built on top of the Eclipse platform. Eclipse is a generic platform for tool development that is extended using its plug-in and extension point technology. For more information about Eclipse, go to the following Web site:

<http://www.eclipse.org>

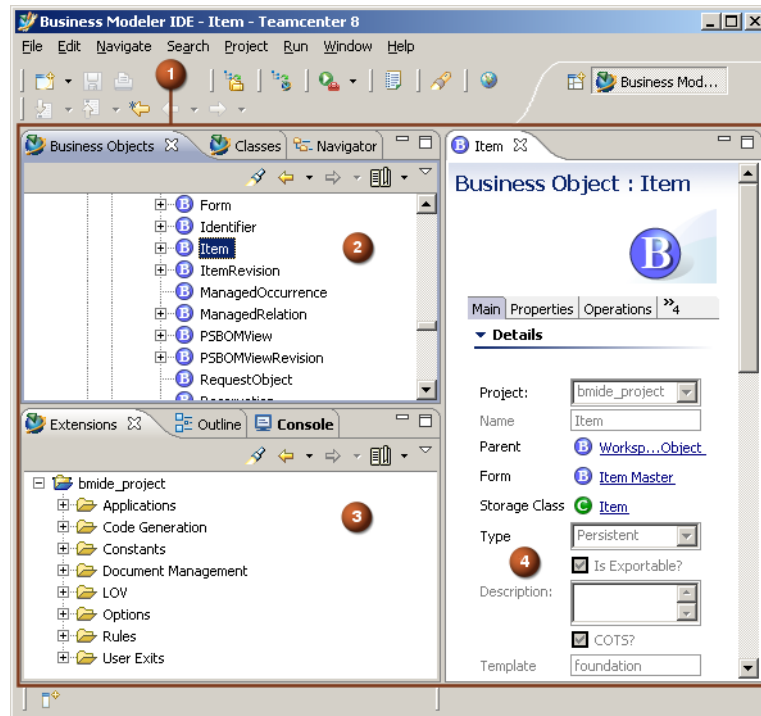
Basic concepts for using the Business Modeler IDE

The Business Modeler IDE is a tool for adding your own data model objects on top of the default Teamcenter data model objects. The Business Modeler IDE accomplishes this by separating your data model into its own set of files that are kept apart from the standard data model, known as the COTS (commercial off-the-shelf) data model.

Data model objects are collected into templates that contain 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 data model, your data model is rolled up into its own template.

As you develop data model, you can deploy it onto a test server to verify that it behaves the way you want it to. After you are finished testing, you can package the data model into a template that can be installed to a production server using Teamcenter Environment Manager (TEM).

Business Modeler IDE interface



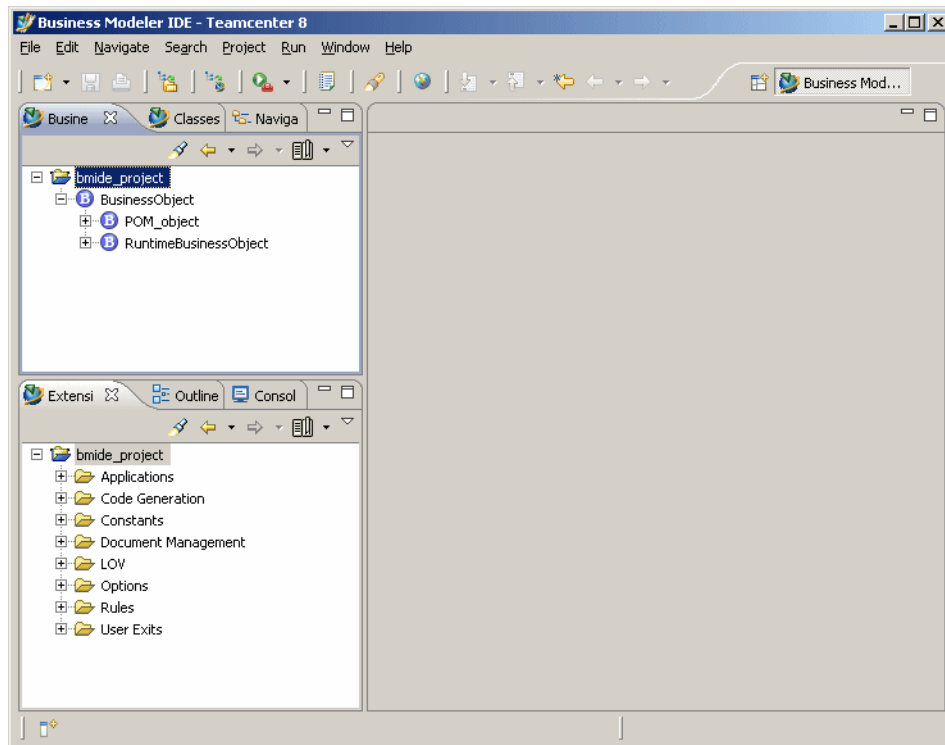
The Business Modeler IDE utilizes the Eclipse user interface, which is composed of perspectives, views, and editors. A *perspective* is an arrangement of views. A *view* is a tabbed window within the UI that provides a view of data. An *editor* is a window that allows you to edit source files. The user can rearrange the user interface in any configuration by dragging and dropping views and editors.

- | | | |
|---|---|---|
| 1 | Business Modeler IDE perspective | Allows you to work with business objects. |
| 2 | Business Objects view | Used for working with business objects, the fundamental objects that model business data. Business objects were formerly known as <i>types</i> in Teamcenter Engineering. |
| 3 | Extensions view | Displays extensions to the data model. |
| 4 | Business Object editor | Enables you to work on different characteristics of the selected business object. |

Business Modeler IDE perspective

The **Business Modeler IDE** perspective allows you to work with many aspects of the Teamcenter data model and functional behavior.

To access the **Business Modeler IDE** perspective, choose **Window→Open Perspective→Other→Business Modeler IDE**.



Business Modeler IDE perspective

Instructor Note:

This is a good time to bring up the Business Modeler IDE and show them the user interface.

Business Modeler IDE views

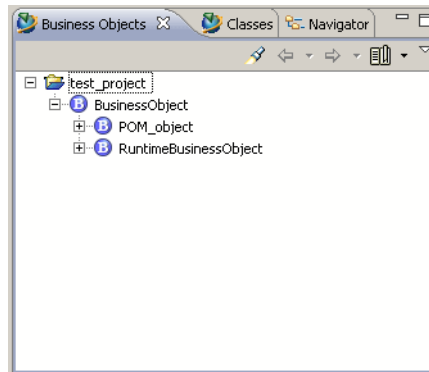
Views are the tabbed panes that appear in the user interface. Most of these views are found in the Business Modeler IDE user interface perspectives; however, some can only be accessed from the **Show View** menu. To access all the Business Modeler IDE views, choose **Window→Show View→Other→Business Modeler IDE**.

Instructor Note:

Topic instructor notes.

Business Objects view

Business objects are the fundamental objects used to model business data. Create business objects to represent product parts, documents, change processes, and so on.



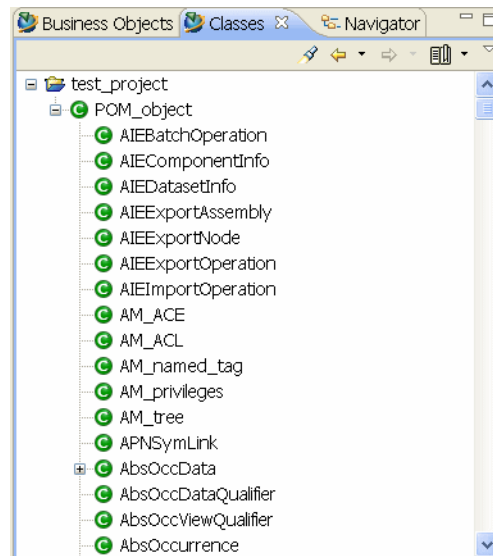
Business Objects view

The **Business Objects** view displays the hierarchy of business objects on your server. Business objects are shown in a tree where the top-level **POM_object** business object is at the top and its children are directly below. The **RuntimeBusinessObject** folder contains business objects that have no parents and are used in run-time processes in Teamcenter.

You can use preferences to set the **WorkspaceObject** business object as the top-level object or to hide the **RuntimeBusinessObject** folder.

Classes view

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)*.

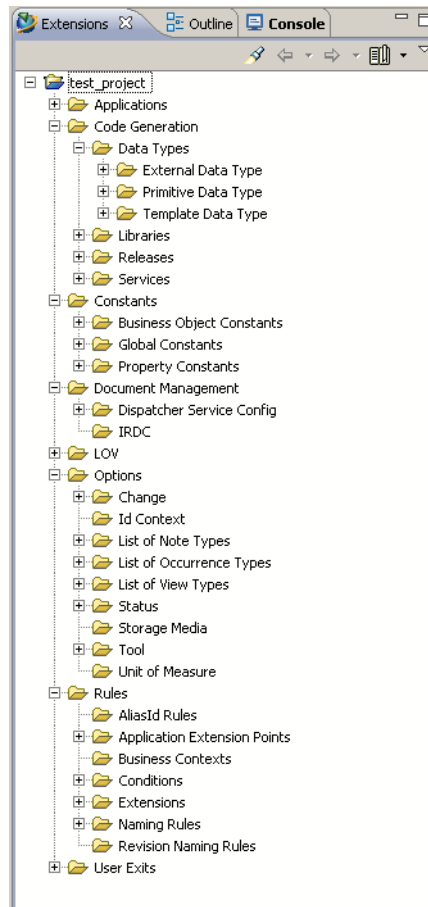


Classes view

The **Classes** view displays the classes on your server. Classes are shown in a tree where the top-level **POM_object** class is at the top and its children are directly below.

Extensions view

The **Extensions** view displays extensions to the data model.



Extensions view

The **Extensions** view contains the following extensions:

- | | |
|------------------------|--|
| Applications | Applications are used on classes to indicate what rich client application can add or edit attributes on that class. |
| Code Generation | This folder contains objects used for software development, such as data types, libraries, and releases. |
| Constants | Constants are reusable values. They provide consistent definitions that can be used throughout the system. |
| LOV | Lists of values (LOVs) are pick lists of values. They are commonly accessed by end users from a menu at the end of a text box. |
| Options | Options are miscellaneous objects such as change requests, units of measure, notes, and so on. |

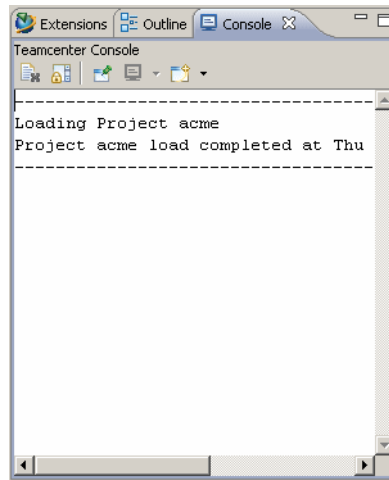
- Rules** Rules define the business rules that govern how objects are handled, including how they are named, what actions can be undertaken on them, and so on. Creating rules is also known as business modeling.
- User exits** User exits are used for adding base action extension rules. User exits are places in the server where you can add additional behavior by attaching an extension to the user exit. To work with user exits, right-click a user exit and choose **Open Extension Rule**.

Instructor Note:

Topic instructor notes.

Console view in the Business Modeler IDE

The **Console** view is a default Eclipse view used to display output from loading and building your projects. Watch this window for any problems or errors with the build, database update, or server startup.

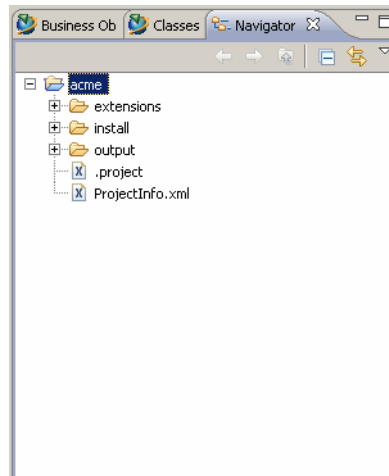


Console view

Typically, an error may occur as a result of incorrectly merging a source file with a source control management (SCM). Each error displays the XML file name, the line number, and error message. All errors should be fixed before continuing to use the Business Modeler IDE.

Navigator view in the Business Modeler IDE

The **Navigator** view is a default Eclipse view used for browsing file system objects. The **Navigator** view always shows the projects at the top-level and any folders and files under them.



Navigator view

For example, if you have a project, expand the project in the **Navigator** view and note the files it contains. You can open files in an editor by double-clicking them, or right-click them and choose **Open**.

Projects typically contain a project file, and source files that contain your extensions to the data model. They can also contain **.tmd** files generated by the UML editor.

Configure the Business Modeler IDE

There are three steps required to configure the Business Modeler IDE prior to using the IDE to extend the data model.

1. 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 that manages your Teamcenter data model extensions in a template. The project contains folders and files that are used to organize your template XML files and to package your template for deployment.

2. Set Business Modeler IDE preferences.

You set Business Modeler IDE preferences from the **Window→Preferences** menu. Most of these preferences are optional. But you must set the **Server Connection Profiles** preference to deploy extensions to a server and to query the server for data.

Note

You can also set preferences for the Business Modeler IDE in the Teamcenter rich client. 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.

3. Add a server connection profile.

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

If you selected 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 named **TcData** is already added.

Server connection profiles

Primary

Secondary

Instructor Note:

Topic instructor notes.

Server connection profile created during the Business Modeler IDE installation

During the Business Modeler IDE installation, the **Configure Teamcenter Servers** feature provides options to automatically create a server connection profile for deployment of data model changes to a server. When you select one of these options, a server connection profile is added in the Business Modeler IDE.

- **2 Tier Teamcenter Server Configuration**

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

- **4 Tier Teamcenter Server Configuration**

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

Caution

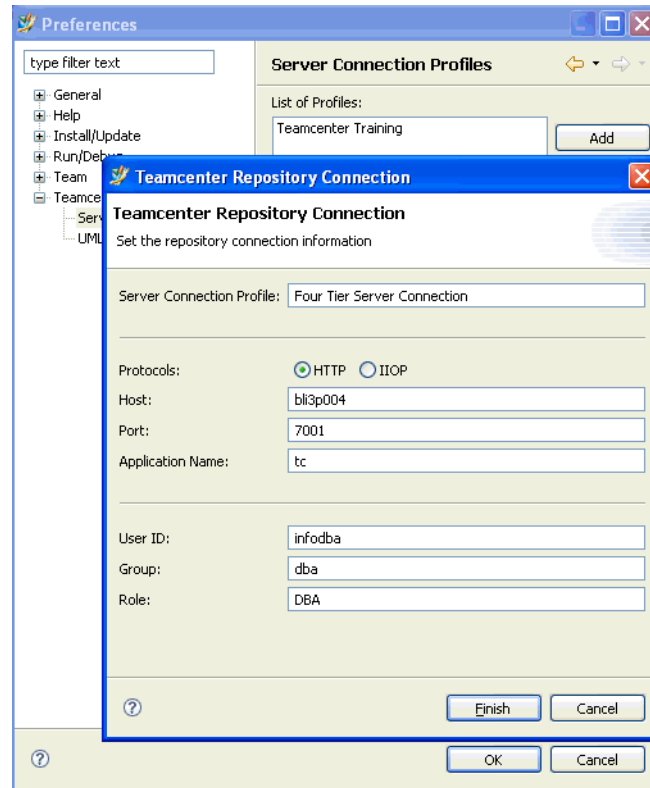
In most instances, only use these options to connect to a test server rather than a production server. However, you can also deploy operational data such as LOVs and rules to production servers.

Instructor Note:

Topic instructor notes.

Add a server connection profile

Server profiles define the Teamcenter servers to connect to.



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. Enter the connection properties.
5. Click **Finish**.
6. Test the connection to the server.

Server connection profile properties

Server connection profiles require different properties based on either a two-tier or four-tier Teamcenter environment.

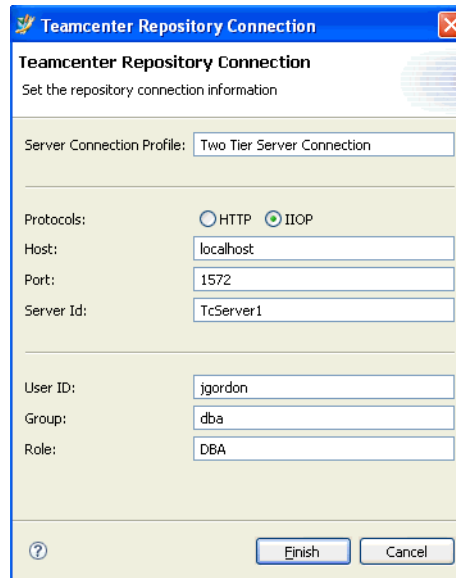
Property	Description	Two-tier	Four-tier
Server Connection Profile	Name for the profile. You can type the name of the server, for example.	✓	✓
Protocols: HTTP	Web communication protocol.		✓
Protocols: IIOP	Network communication protocol.	✓	
Host	The host name assigned to the Teamcenter server. For two-tier, this is typically set to localhost .	✓	✓
Port	The port assigned to the Teamcenter server.	✓	✓
Application Name	The name of the application to connect to. The default application name for four-tier is tc .		✓
Server ID	The ID of the server on the network, for example, TcServer1 .	✓	
User ID	The ID of the authorized user on the Teamcenter server. This user must be a member of the dba group and the DBA role in order to successfully deploy extensions.	✓	✓
Group	The group the user is assigned to (for example, type dba for the database administration group). This is optional.	✓	✓
Role	The role the user is assigned (for example, type DBA for the database administrator role). This is optional.	✓	✓

It is important to test the connection to the server. Simply attempt to load the organization from the Teamcenter server from the **Display Rules**.

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

Deploy with a two-tier server connection profile

To deploy to the Teamcenter server, right-click a project and choose **Deploy Template**. In the two-tier server connection profile in the Business Modeler IDE, choose IIOP as the protocol.



Tip

The IIOP connection information is obtained from the following file:

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

RELEASE represents the current product release number and equates to something like 8000.0.0.

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

Relevant Business Modeler IDE installation information

Primary

Secondary

Instructor Note:

Topic instructor notes.

Before you begin

Prerequisites

The following are required on the machine that runs the Business Modeler IDE:

Java™ Runtime Environment JRE 1.6 (6.0)

For the installation of the stand-alone version of the Business Modeler IDE.

Java SDK 1.6 (6.0)

For the installation of the Business Modeler IDE into an existing Eclipse environment.

One GB of RAM dedicated to the Business Modeler IDE

You can allocate memory to the Business Modeler IDE in the **BusinessModelerIDE.ini** file and in a **BMIDE_SCRIPT_ARGS** environment variable.

Eclipse 3.3

This is required only if you install the Business Modeler IDE into an existing Eclipse installation.

Administrator rights

Users of the Business Modeler IDE must be members of the database administrators (**dba**) group on the Teamcenter test server. Use the Organization application in the Teamcenter rich client to add a user to the **dba** group.

Enable the Business Modeler IDE

To enable the Business Modeler IDE, you must install it either as a stand-alone application, or place the plug-ins into an existing Eclipse environment.

Start the Business Modeler IDE

To start the stand-alone Business Modeler IDE application, run **bmide.bat** (Windows) or **bmide.sh** (UNIX). To start the Business Modeler IDE when it is distributed in an Eclipse environment, run **eclipse.exe**.

Configure the Business Modeler IDE

Before you can use the Business Modeler IDE, you must configure it by creating a project, setting preferences, and creating a server profile.

Install the Business Modeler IDE as a stand-alone application

1. Start Teamcenter Environment Manager (TEM).
2. Proceed to the **Solutions** panel, select **Business Modeler IDE**, then click **Next**.
3. From the **Select Features** window, select these **Business Modeler IDE** features.
 - ☒ **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.
 - ☒ **Client**

Installs the IDE as a client on your machine.
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**

Select this option if you want to connect to a server in a two-tier environment over a network using IIOP protocol.
 - ☒ **4 Tier Teamcenter Server Configuration**

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

Caution

In most instances, only use these options to connect to a test server rather than a production server. However, you can also deploy operational data such as LOVs and rules to production servers.

Install the Business Modeler IDE as a stand-alone application (continued)

5. In the **Installation Directory**, enter the location where you want to install the Business Modeler IDE. The Business Modeler IDE files are installed to a **bmide** subdirectory.
6. From the **Business Modeler IDE Templates** panel, select the templates to install. *Templates* contain the data model for Teamcenter solutions.

☒ **Teamcenter Foundation**

Select the same templates that were installed on the server so that you can see the same data model definitions in the Business Modeler IDE that were installed on the server. At a minimum, select the **Teamcenter Foundation** check box. The Foundation template contains the data model used for core Teamcenter functions. All customer templates must extend the Foundation template.

7. Verify the installed files in the *install-location* directory.

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.
- Start from an Eclipse environment.

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

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

2. Click one of the buttons in the **Welcome** window to learn more about the Business Modeler IDE:

- The **Overview** button  provides links to online help topics.
- The **Tutorials** button  provides links to tutorials.

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

Note

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

4. Choose **Window**→**Open Perspective**→**Other**→**Business Modeler IDE**.

The **Business Modeler IDE** perspective is displayed.

5. In the **Business Modeler IDE** perspective, click the **Business Objects** tab to open the **Business Objects** view. This view displays the business objects (types) defined in the data model. Use this view to create new business objects to represent parts, documents, and so on.

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\iiopservers\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.

Instructor Note:

TAO (The ACE ORB) is CORBA compliant ORB that gives c++ mapping. TAO is freeware that can be used instead of ORBIX.

Activities

In the *Introduction to the Business Modeler IDE* section, do the following activities:

- Start the Business Modeler IDE.
- Use the Welcome window buttons.

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.

- 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

Instructor Note:

Answers to review questions

1. True
2. Tabbed window within the UI that provides a view of data.
3. Client-Server communication.
Deploy your extensions.

Activities

In the *Introduction to the Business Modeler IDE* section, do the following activities:

- Create a project.
- Open a server connection profile.
- Navigate views.
- Create a new perspective.

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

Instructor Note:

Answers to review questions

1. A container of resources for developing, managing, packaging, and deploying a single template.

An environment which manages your Teamcenter data model extensions in a custom template.

2. False

Activity

In the *Introduction to the Business Modeler IDE* section, do the following activities:

- Add a template to the Business Modeler IDE.
- Deploy the business object.

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.

Instructor Note:

Activity instructor notes.

Answers to review questions

1. False
2. Determined at runtime.
Not stored in the database.
3. Specify the file where you want your extensions to be saved.
Perform the extension work.
Save your work.
Verify your extensions by deploying them to a test Teamcenter server.

Summary

The following topics were taught in this lesson:

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

Instructor Note:

It is important at this point that the student are comfortable with the BMIDE environment and are ready to start building their custom template with new data model extensions. Managing templates in a customer environment will be covered in more detail in a later lesson.

CreateDescriptor tab

In the BMIDE, look at the Item business object. The CreateDescriptor tab controls what is required and displays in the create Item wizard on the item form. Also on this tab the Master form shows as a compound property. In this case, compound property is read only.

There is also a Visible property constant defined from the Properties tab that controls the display of the property on the Properties dialog.

See CPMarketingBrief for a sample of no master data.

See AUTItem for an example of a split screen in RC showing both Rev properties and rev master properties on the same form.

Secondary

Instructor Note:

Topic instructor notes.

Manage business objects for creation

Primary – must add content.

You can use the **CreateDescriptor** tab on business objects to make properties visible and required on creation dialog boxes in the rich client or thin client. For example, if certain properties are checked off as visible and required in the **CreateDescriptor** tab for the **Item** business object, when a My Teamcenter user chooses **File**→**New**→**Item** to create a new **Item** object, these properties are visible and required in the creation dialog boxes.

Because **Item** business objects and its children have master and revision master form business objects, you can also use the **CreateDescriptor** tab to make properties visible or required for these forms on the creation dialog boxes. (The **CreateDescriptor** tab does not display on children of the **Form** business object.)

The **CreateDescriptor** tab displays the metadata of the properties on the selected business object. You can use this tab to choose the properties that are seen on dialogs when the user creates items. For example, when a My Teamcenter user chooses **File**→**New**→**Item** to create new **Item** object, properties that are checked off as visible and required in the **CreateDescriptor** tab for the **Item** business object are visible and required in the creation dialog boxes.

1. To access the **CreateDescriptor** tab, in the **Business Objects** view, right-click a business object, choose **Open**, and click the **CreateDescriptor** tab.
2. In the **CreateDescriptor** tab, select a property on the table and click the **Edit** button to make that property visible or required on creation dialog boxes.

The **Operation Input Property Descriptor** area on the tab provides details of the selected property.

3. Click the **Add** button to the right of the table to add a new property to the list of visible and required properties on this business object.

1. Access the **Business Modeler IDE** perspective by choosing **Window**→**Open Perspective**→**Other**→**Business Modeler IDE**.
2. In the **Business Objects** view, right-click the project in which you want to make the change and choose **Organize**→**Set active extension file**. Select the file where you want to save the data model changes, for example, **business_objects.xml**.
3. Right-click the business object for which you want to make the change, choose **Open**, and click the **CreateDescriptor** tab in the resulting view.

4. In the **CreateDescriptor** tab, if there is a property in the table for which you want to change its visible or required behavior, select the property in the table and click the **Edit** button.

In the **OperationInput Property** dialog box, select the **Required** or **Visible** check boxes and click **Finish**.

5. If there is a property you want to add to the list of visible and required properties on the business object, click the **Add** button to the right of the table.

The New OperationInput Property wizard runs.

6. In the **OperationInputProperty** dialog box, choose one of the two methods to add a property:

- **Add Property from Business Object.**

Select this option to add an existing property. Click **Next** and perform the following steps:

- a. Click the **Browse** button to the right of the **Property Name** box to select an existing property.
- b. Select **Required** to make the property required for entry on the creation dialog box. The property on the creation dialog box displays a red asterisk indicating that the user is required to fill it in.

Select **Visible** to make the property display on the creation dialog box.

- c. In the **Description** box, type a description of this property.

- d. Under **Usage**, select one of the following:

- **None**

Select if the property is not a relation or reference property that references a secondary object.

- **Type**

Select if this is a property associated with a secondary object that is to be created from the value in the **CompoundObjectType** box.

- **Constant**

Select if this is a property associated with a secondary object that is to be created from the value entered in the **CompoundObjectConstant** box.

- e. The **CompoundObjectType** box is displayed if you selected **Type**. This is the type of secondary object that is to be created.

Click the **Browse** button to the right to select the secondary business object.

- f. The **CompoundObjectConstant** box is displayed if you selected **Constant**. This is the name of the business object constant associated with the primary object that is created. The value of this constant indicates the type of secondary object to be created. Click the **Browse** button to the right to select the business object constant.

- g. Click **Finish**.

- **Define Property from Business Object**

Select this option to create a new run-time property. The run-time property is only associated with the **CreateDescriptor** and not with the source business object. This can be used in cases where certain information required for creation needs to be specified but there is no associated source property for it. An example would be when some information required for creation needs to be specified on the create dialog but there is no associated property for it on the primary object. Another example is when a secondary object needs to be created, but there is no corresponding relation or reference property on the primary object.

The process for creating a run-time property here is similar to creating a run-time property using the **Add** button in the **Properties** tab.

- a. Click **Next**.
- b. In the **Name** box, type a name for the new run-time property.
- c. In the **Attribute Type** box, select the storage type for the attribute, for example, **String**. Choose from the following attribute types:
 - **Character**
A single character, such as **A**, **B**, **Z**.
 - **Date**
A calendar date. A form using this format displays a shortcut date selector.
 - **Double**
An 8-byte decimal number from the range 1.7E to 308.

- **ExternalReference**
Points to data outside of Teamcenter.
 - **Integer**
An integer without decimals from 1 to 999999999.
 - **Logical**
A boolean value of **True** or **False**.
 - **LongString**
A string of unlimited length.
 - **String**
A string of ASCII characters.
 - **TypedReference**
Points to a Teamcenter class.
 - **UntypedReference**
Points to any class of data.
- d. If the attribute is a string attribute, in the **String Length** box, type the character length of the attribute.
 - e. If you selected **TypedReference** as the attribute type, in the **Reference Business Object** box select the class.
 - f. In the **Description** box, type a description of the run-time property.
 - g. Select **Required** to make the property required for entry on the creation dialog box. The property on the creation dialog box will display a red asterisk indicating that the user is required to fill it in.

Select **Visible** to make the property display on the creation dialog box.
 - h. Click **Finish**.
7. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button in the main toolbar.

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

8. Deploy your changes to a test server. Right-click the project in the **Business Objects** view and choose **Deploy Template**, or select the project and click the **Deploy Template** button in the main toolbar.
9. After deployment, test to ensure the property is visible or required when you create an instance of the object in the Teamcenter rich client.

For example, if you made a property required in the **CreateDescriptor** tab on the **Item** business object, when a My Teamcenter user chooses **File→New→Item** to create a new **Item** object, the property is required on a creation dialog box. Required properties are marked with a red asterisk (*).

Lesson

4 *Managing Templates*

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:

- Install templates to the Business Modeler IDE and the server.
- 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 *Basic concepts for using the Business Modeler IDE*)

Instructor Note:

Lesson instructor notes.

Overview of template installation

To successfully use the Business Modeler IDE, you must understand how to install templates to the Business Modeler IDE and the server. The key thing to remember is that the Business Modeler IDE and the server are completely separate, and if you install a template to the server, you must separately install the same template to 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 data model, the new 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 use hot deploy to send operational data such as LOVs and rules to a production server.

You can also package your 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 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\version\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.

Templates overview

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:

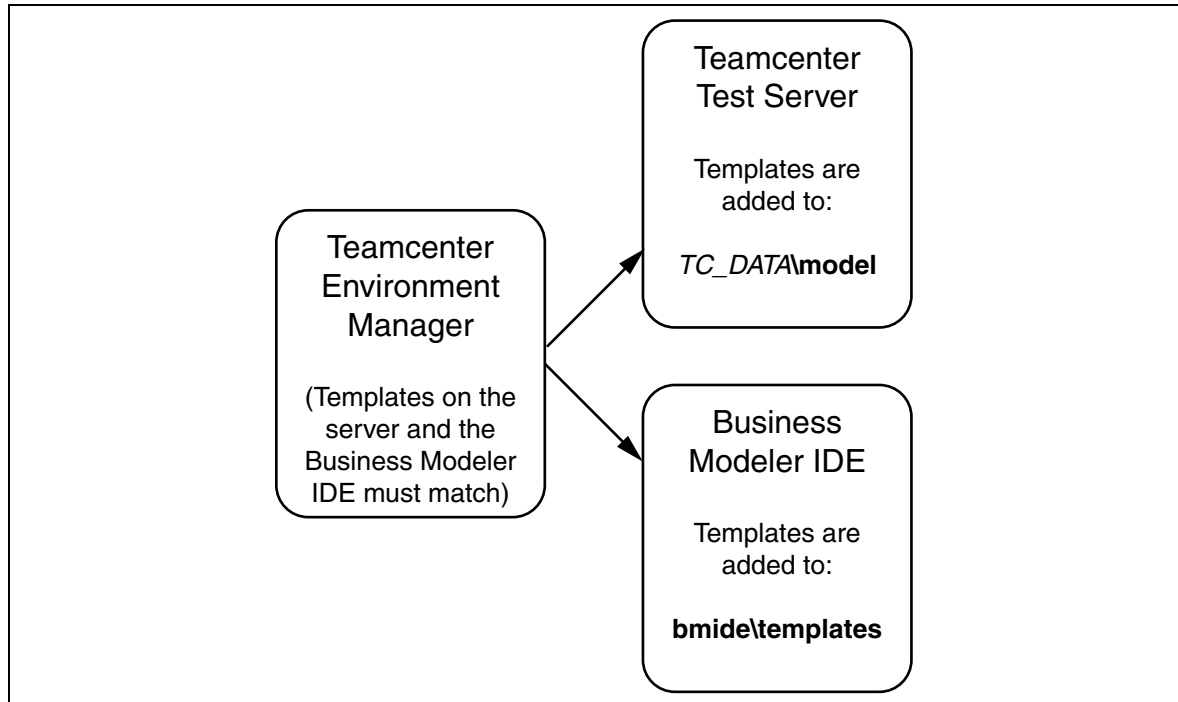
- a consistent method for defining extensions for customers and Siemens PLM Software.
- a mechanism for building and packaging solutions.
- a mechanism for protecting extensions that you share with others.
- a mechanism for concurrent development support.

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.

Install the initial template

When you use Teamcenter Environment Manager (TEM) to initially install a server and the Business Modeler IDE, you must install the same templates to both. The templates on each must match (see the figure below).



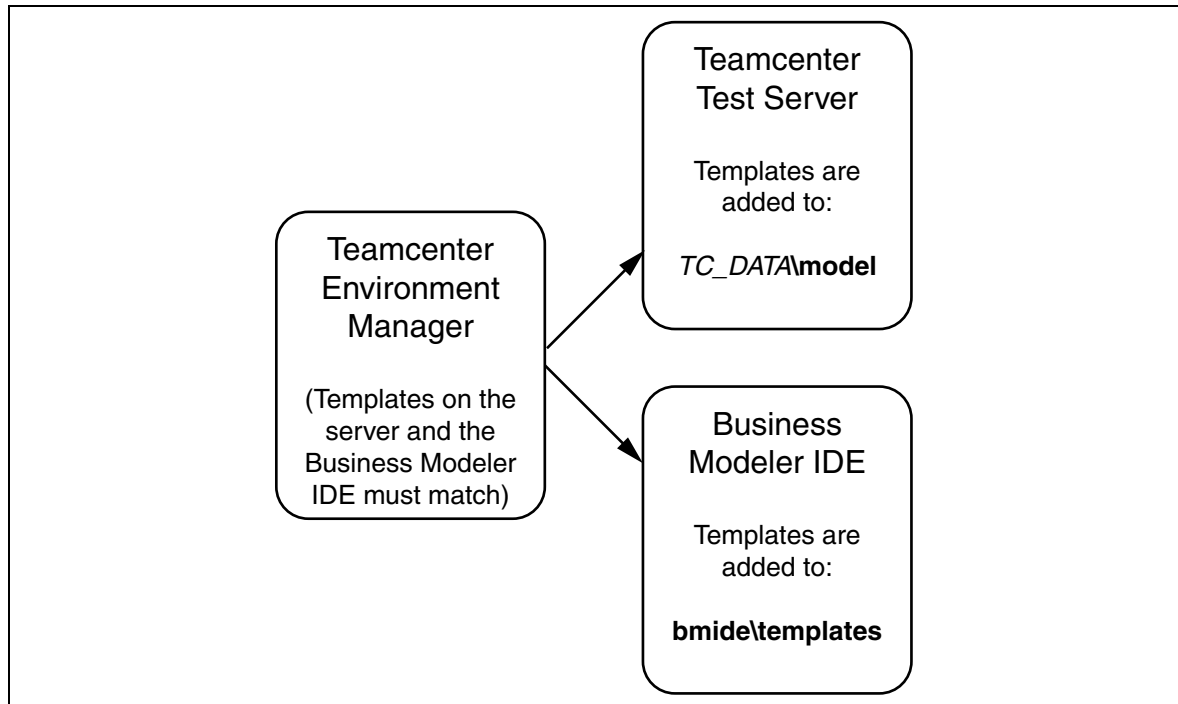
Initial template installation

1. Run the Teamcenter Environment Manager and proceed to the **Select Features** panel.
2. Under **Corporate Server**, select **Teamcenter Foundation** and other templates as needed.
3. Under **Business Modeler IDE**, select **Business Modeler IDE Templates**.
4. Click **Next**.
5. In the **Business Modeler IDE Templates** panel select the **Teamcenter Foundation** template and the same templates that are installed to the server.
6. After you have installed the Business Modeler IDE, run the Business Modeler IDE and create a project by selecting **File→New→Project→Business Modeler IDE→New Business Modeler IDE Template Project**.

When you create the project, in the **Templates** pane select the same templates that are installed on the server.

Install a subsequent template

After you initially install templates to a server and the Business Modeler IDE, you can subsequently go back and add templates (see the following figure). Because the templates on a server and the Business Modeler IDE must match, when you install a new template to an existing server, you must install the same template to the Business Modeler IDE.



Subsequent template installation

1. Perform the following steps to install a new template to the server (test or production):
 - a. Run the Teamcenter Environment Manager and proceed to the **Feature Maintenance** panel.
 - b. Select **Add/Remove Features** and click **Next**.
 - c. In the **Select Features** panel, select the template, or click **Browse** to locate the template.

Note

Standard Teamcenter templates are located on the Teamcenter software distribution image in the **tc** folder.

2. Perform the following steps to install a new template to the Business Modeler IDE:

- a. Run the Teamcenter Environment Manager and proceed to the **Feature Maintenance** panel.
- b. Select **Add/Update Templates for working within the Business Modeler IDE Client** and click **Next**.
- c. In the **Business Modeler IDE Templates** panel, select the template, or click **Browse** to locate the template.

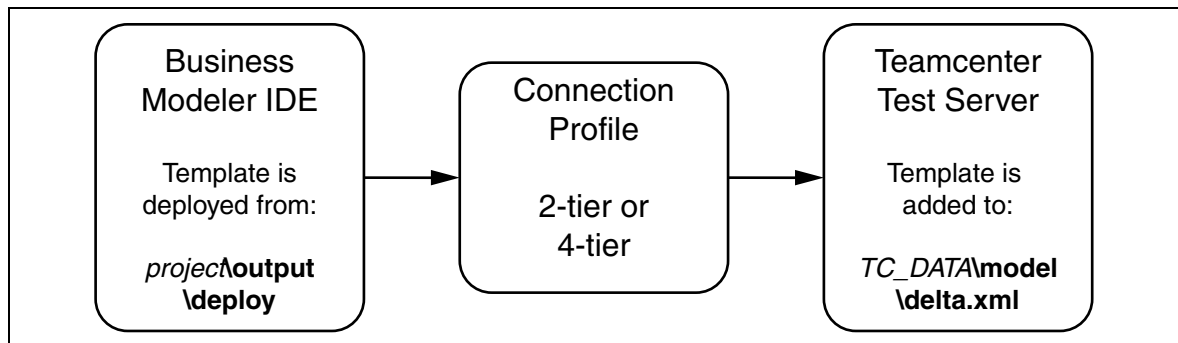
Note

Standard Teamcenter templates are located on the Teamcenter software distribution image in the **tc** folder.

- d. Perform the following steps in the Business Modeler IDE to add the new template to your project:
 - 1) In the **Navigator** view, right-click the project and choose **Properties**.
 - 2) Choose **Teamcenter**→**BMIDE** in the left pane.
 - 3) In the **Templates** pane, select the new template.

Hot deploy a template

To send your template to a Teamcenter server for testing purposes, right-click your project in the Business Modeler IDE and choose **Deploy Template**. This is also known as *hot deploy* (see the figure below). You can also use hot deploy to send operational data (such as LOVs) to a production server.



Hot deploy a template to a test server

Deploy extensions overview

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_DATA\model

Deploying templates

After you make changes to the data model, you can deploy them to a server using the Deploy wizard. Deploying templates 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 have two options to deploy templates:

1. Right-click the project in the **Business Objects** view and choose **Deploy Template**.
2. Select the project and click the **Deploy Template**  button on the main toolbar.

Use the Deploy wizard in two different situations:

- **Deploy a template to a test server**
- **Deploy operational data to a production server**

Deploy a template to a test server

Hot deploy to a test server when you want to verify your custom data model before packaging it into an installable template and installing it to a production server. This is recommended in most situations.

1. Save any changes to the data model. Do this by selecting the project and clicking the **Save Data Model** button on the main toolbar.
2. Ensure that the Teamcenter test server is running where you plan to deploy your updates.
3. Deploy your changes to the data model. Do this by selecting the project and clicking the **Deploy Template** button on the main toolbar.

The Deploy wizard runs.

4. Validate the server connection profile to ensure you are deploying to the correct test server. At a minimum, you must enter the authorized user password.
5. Connect to the test server by clicking **Finish**. A message appears at the bottom of the dialog box stating **Deploying: Consolidating the model**.
6. When deployment finishes, review the deployment log file by clicking the link in the **Console** view. If there are any deployment errors, take corrective action and attempt to deploy again.
7. Verify the data model changes are in the test 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.

Instructor Note:

If you deploy before saving, you will be prompted to save. Deploy and save menu options are also available besides the buttons for these on the button bar.

Deploy operational data to a production server

Hot deploy to a production server when you have *operational data* to place on the server, such as LOVs and rules.

Operational data is nonschema data that must be updated on a regular basis. In this situation, create a template that only contains operational data, and use a source control management (SCM) system to manage the versioning of the template source file.

Following is the operational data that can be deployed to a production server using hot deploy:

- Lists of values (LOVs)
 - LOVs
 - LOV attachments to properties
- Business rules
 - Compound property rules
 - Deep copy rules
 - Display rules
 - Extension rules
 - GRM rules
 - ID context rules
 - Naming rules
- Options
 - Changes
 - ID contexts
 - Occurrence types
 - Statuses
 - Storage media
 - Tools
 - Units of measure
 - View types
- Constants
 - Business object constants
 - Global constants
 - Property constants
- Rules engine framework (RBF)
 - Application extension points
 - Application extension rules
 - Business contexts

The following nonoperational (schema) data cannot be hot deployed to a production server:

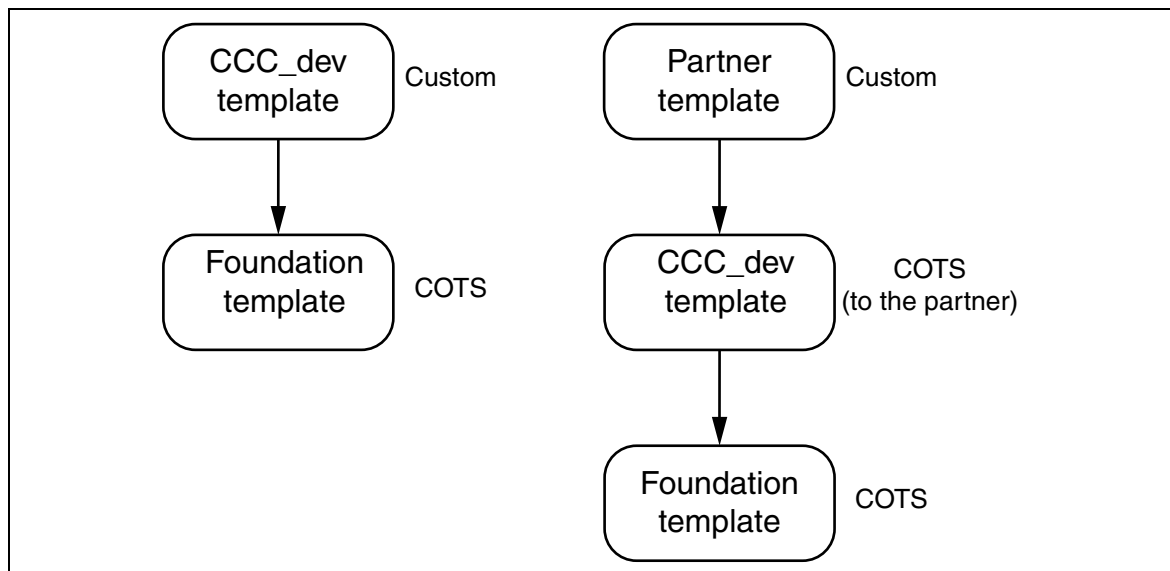
- Classes
- Attributes
- Business objects
- Properties

Note

If your process requires you to add both operational and nonoperational data to a production server, you must use an SCM system and two Business Modeler IDE clients. One client should be used for deploying operational data on a regular basis, and the other for maintaining nonoperational data to be packaged into an installable template. Use the SCM to synchronize both sets of definitions to update the production system with the installable template.

Understanding custom versus COTS templates

Custom data model objects are those objects you create and store in a custom template. *COTS* (commercial-off-the-shelf) data model objects are the objects that your custom data model objects are dependent on. Therefore, custom templates are dependent on COTS templates. You can change or delete only the custom objects in the data model.



Custom versus COTS template example

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.

Note

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.

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. Importing also migrates the template to the latest version of the data model.

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 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 the extensions.
 - c. Click the **Browse** button to the right of the **Reference templates directory** box to choose the directory where the dependent template XML files are stored. The templates directory is created when you install the Business Modeler IDE.
 - d. Click the **Browse** button to the right of the **Code Output Location** directory and browse to the location where you want the project output to be placed.
 - e. Click **Finish**.

If a project from previous release is being imported, a dialog asks if you want to migrate the project to the current data model format.

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 template file

You can import the contents of a template file into a Business Modeler IDE project. The data model elements in the template file are written to an extension file.

This is useful when you want to import a template file to your project. You can also import 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 template name and template 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 template file**. Click **Next**.
3. Perform the following steps in the **Import template file** dialog box:
 - a. In the **Project** box, select the project into which you want to import the template file.
 - b. Click the **Browse** button to the right of the **Template file** box to choose the XML model file to be imported, for example, a template XML file or an extension file.

Note

This wizard does not migrate the file to the latest data model format. You must use a template file that was created using the latest release XML format, or that has already been migrated to this format.

- c. Click the arrow in the **Extension file** box to choose the extension file in your project 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.

Activities

- Package extension into a solution template
- Import a project.

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.

Instructor Note:

Activity instructor notes.

Answers to review questions

1. True.
2. Load and validate your model.

The remaining answers are why the **Reload Data Model** validation tool is important.

Summary

The following topics were taught in this lesson:

- Installing templates from the TEM.
- Package extensions to the data model as a solution template and distribute the template for installation to a production environment.
- The template project directory and all of its contents should be managed in a source control management system (SCM).
- The Business Modeler IDE performs strict validations on the extensions in your template.
- You can import either an entire project template or just a model file.

Instructor Note:

Summary instructor notes.

Lesson

5 *Teamcenter data model*

Purpose

The purpose of this lesson is to introduce using the Business Modeler IDE.

Objectives

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

- Find and create business objects.
- Find and create classes.
- The ...

Help topics

Additional information for this lesson can be found in:

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

Instructor Note:

Lesson instructor notes.

Data model concepts introduction

It is important to understand the business object and class hierarchy in the Business Modeler IDE. Knowing how business objects (also properties), classes (also attributes) relate to one another helps you extend the data model when required.

This section covers the following topics:

- What are business objects and classes.
- Why configure the data model using business objects.
- Data model *inheritance*, and the effective use of inheritance.

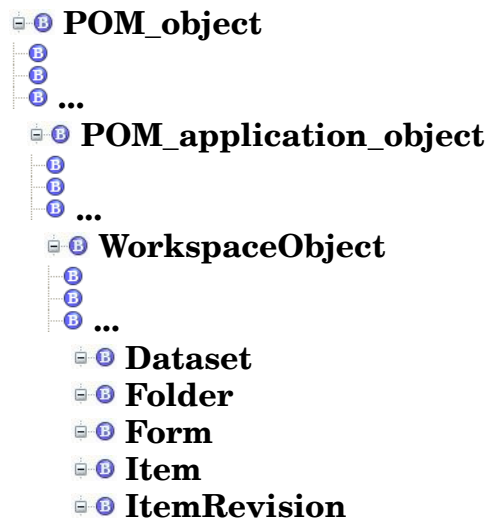
To examine the data model in the Business Modeler IDE, use the **Business Object** and **Class** views. For a graphical view, right-click a business object or class and choose **Open in UML Editor**.

Instructor Note:

Topic instructor notes.

What are business objects

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

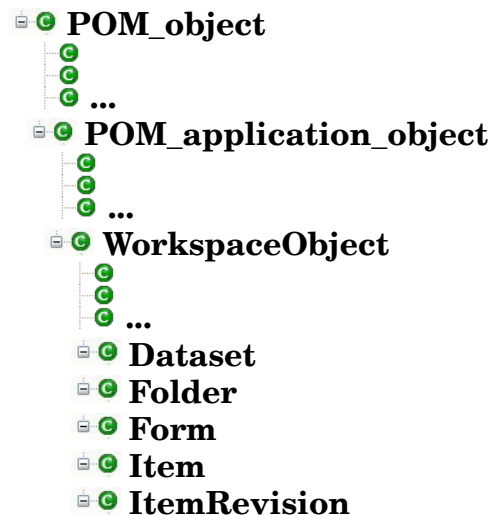


At the very top of the **Business Objects** view the parent of all business objects is displayed as **POM_object**. Each POM class is mapped to a primary business object whose name is the same as the POM class name. Primary business objects are derived from classes such as **Dataset**, **Folder**, **Form**, **Item**, **ItemRevision**, and so on. Sub-business objects are under these primary business objects.

What are classes

Classes are the persistent representations of the POM schema, and can be viewed in the **Classes** view of the Business Modeler IDE. POM defines an architecture (schema) for managing Teamcenter data in a database and in a running Teamcenter session.

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



POM stands for:

- *Persistent*: survives from session to session
- *Object*: an entity being managed (a part, file, user, and so on)
- *Manager*: a method of managing the objects

POM interface to database

POM is a layer between Teamcenter and the database. As such, POM performs many tasks, such as managing concurrent access to persistent data to avoid multiple users overwriting each other's changes. POM manages data in tables within a relational database, such as Oracle.

The basic things to keep in mind about POM are that:

- Each POM class is represented in the database by a table.
- Each instance of a POM class is represented by a row in the class table.
- Each class attribute is represented by a column in the class table.
- An object that is derived from multiple classes requires a *join* across each of the class tables from which it is derived.

Instructor Note:

Topic instructor notes.

Introduction to attributes

An *attribute* is a persistent information tag assigned to all objects in the same class. When you right-click a class in the **Classes** view of the Business Modeler IDE and choose **Open**, the attributes of the class appear in a table.

Attributes are class characteristics, such as name, number, description, and so on. Attributes are attached to classes, and all children of a class inherit the attributes of the parent class.

For example, the **object_name** attribute contains a value because it is a string attribute; it can have a value such as **000001/A**. The **last_mod_user** attribute contains a reference, because it is a typed reference attribute; its value points to a user's name.

Special notes about attributes:

- Attributes can contain one value (scalar) or contain many values (array). Attributes are of a defined data type, for example, integer, string, float, and so on.

- An attribute contains either a *value* (if the attribute is an integer, string, float, date, or logical data type), or it can contain a *reference* to another class if the attribute is a typed reference or untyped reference.

Instructor Note:

Topic instructor notes.

An example of a scalar attribute is the **last_mod_user** attribute, because it contains a single value, the user name. An example of an array attribute is the **project_list** attribute, because it contains a list of projects, for example, **Car05**, **Car06**, **Car08**.

Working with 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.

Why create business objects?


Business objects are the fundamental objects used to model business data. One of the most important jobs you perform in the Business Modeler IDE is to create business objects to represent different kinds of parts, documents, change processes, and so on. Your company uses business objects to organize all the things it produces into categories for accuracy and consistency.

Siemens PLM Software strongly urges you to plan out your business object creation. Perform an object-oriented analysis to determine the optimal business object structure, and the custom properties you want to place on the new business objects.

Instructor Note:

Topic instructor notes.

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.

Instructor Note:

Demonstrate finding and opening a business object.

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.

Teamcenter business objects

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

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.

Note

Data model details are explained in the next lesson.

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)**

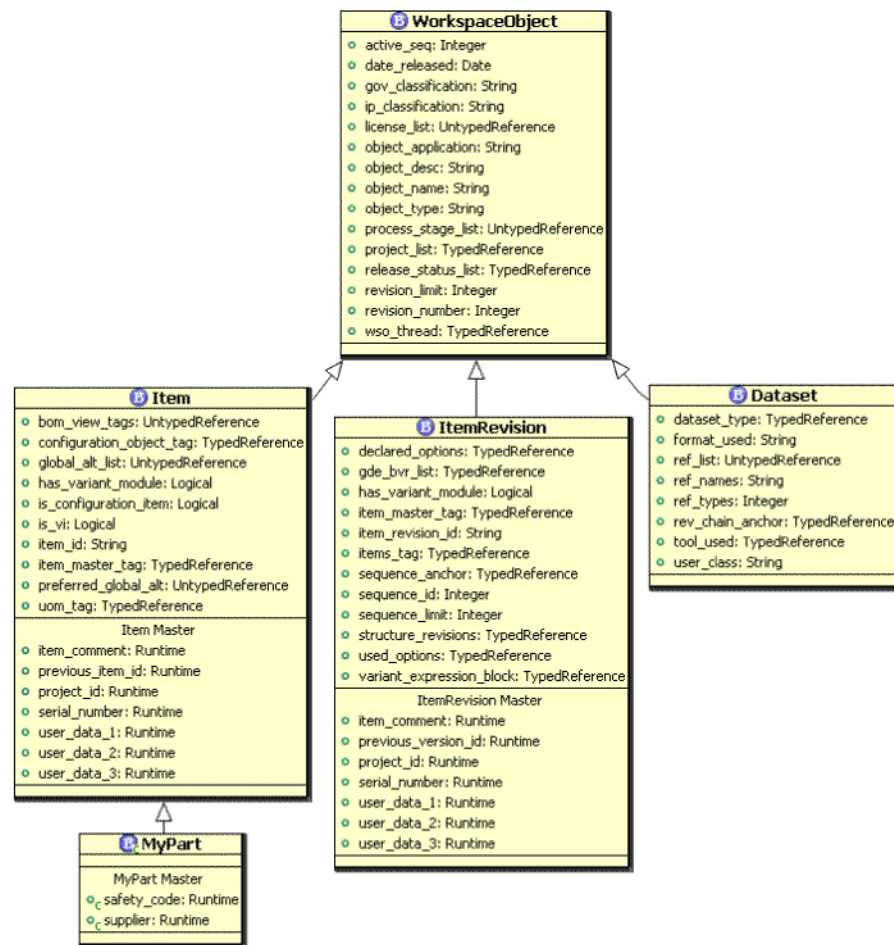
A form object that is often used to extend the stored property data for an item revision to include data unique to the customer.

Business objects and classes

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

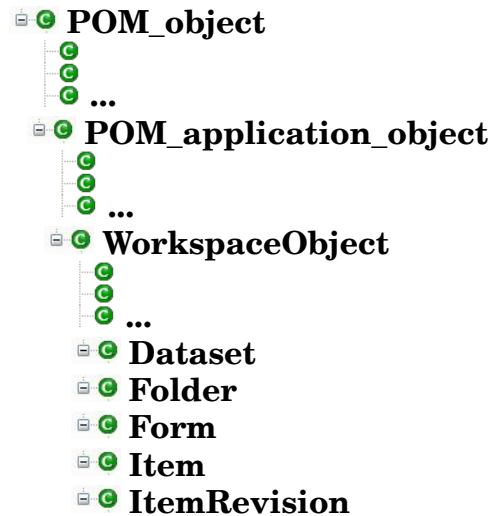
Classes are the persistent representations of the data model schema and provide properties to business objects. The class hierarchy is known as the Persistent Object Model (POM).

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.



Note

POM schema details are explained in the next lesson.

The **POM** (persistent object manager) is the interface between Teamcenter objects and the Relational Database Management System (RDBMS). The **POM** 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.
- 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 **WorkspaceObject** abstract class has subclasses for the most commonly created business objects, include:

- **Item** – represents parts and documents.
- **Form** – displays properties.

- **Dataset** – represents file types.

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 a product (component, assembly, end item) or other data such as procurement specification, test procedure, standard part, shop tooling, change, and so on.

Data modeled using the **Item** business object is generally revision controlled data and all revisions of the information must be maintained, tracked, and recoverable. Data must also be modeled using the Teamcenter concept of item if you want to build a structure of the items such as a bill of material (BOM) for items that represent a product.

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

- **Item**

Generally used for data that represents an element of a 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 as a product or is not defined using CAD applications.

Business objects and properties

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

Introduction to the UML editor

UML (Unified Modeling Language) editor shows business objects and classes in a graphical format. To access the UML editor, right-click a business object in the **Business Objects** view or a class in the **Classes** view and choose **Open In UML Editor**. The business object or class appears in the UML editor.

To work with the business object or class displayed in the editor, right-click the object and make selections from the menu. You can also use the palette on the right side of the editor. The UML editor allows you to graphically view and change the data model for business objects and classes. You can do much of your data model extension work directly from the UML editor.

Note

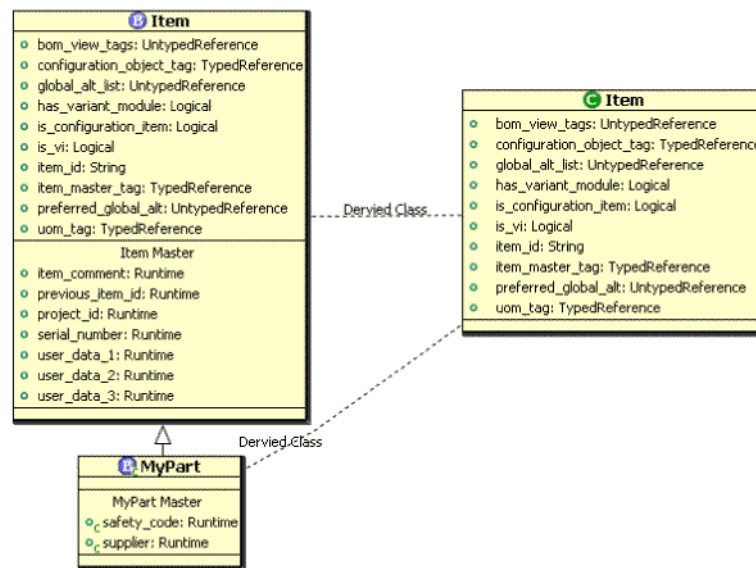
UML files only store the names and relative positions of the data model. The definitions of data model created with the UML editor are not stored in the UML files. They are stored in the project's template XML files.

What is a UML diagram

The Business Modeler IDE UML diagram can be used to quickly understand the *business objects* and their associated *classes*.

Business objects

Class

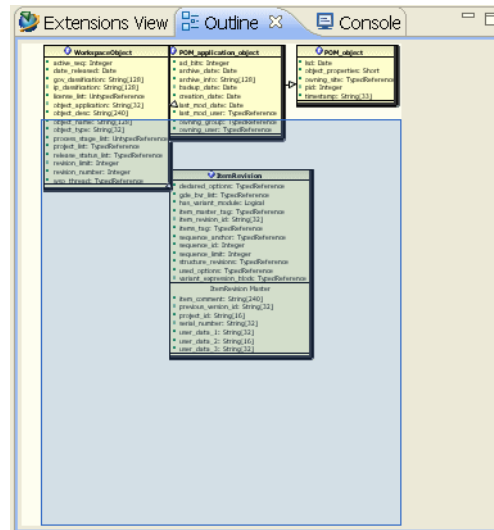


Key points

- Business objects **B** 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 **C** 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 an associated class where the attribute data is persisted.
- The attributes on the associated class are displayed as properties on the business object.

Outline view in the Business Modeler IDE

The **Outline** view is a default Eclipse view that is used for displaying a thumbnail of the contents of the UML editor. Dragging the shaded box changes what is displayed in the UML editor.



Outline view

Managing the data model using the UML editor

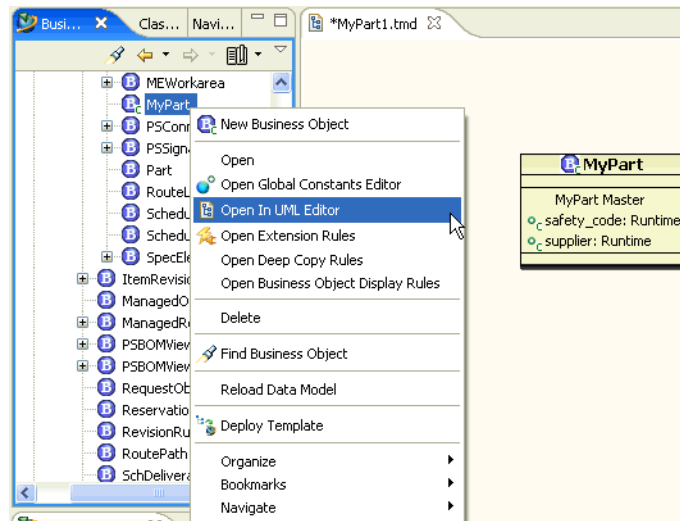
You manage the data model in the UML editor by:

- 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.

Instructor Note:

Demonstrate the features of 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**.

Instructor Note:

Demonstrate opening a business object in the 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. You can also drag other business objects from the **Business Objects** view into the UML editor.

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 see operations on a business object, choose **Open Extension Rules**.
- To see the relationships between business objects, drag business objects from the **Business Objects** view into the UML editor, press

the Ctrl or Shift key and click multiple business objects to select them, and right-click and choose **Show→Relations**.

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

Activities

In the *Data Model* section, do the following activities:

- Find and open a business object.
- Find and open a class.
- Deploy the new business objects.
- Open a business object in the UML editor.
- Use the UML editor.
- Save a UML diagram.

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**

Instructor Note:

Activity instructor notes.

Answers to review questions

1. 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. False

3. **Show→Children**

Show→Inheritance to Root

Show→Parent

Class attributes reference

Attributes are class characteristics, such as name, number, description, and so on. Attributes are attached to classes, and all children of a class inherit the attributes of the parent class.

Secondary

Instructor Note:

Topic instructor notes.

Tasks for using the data model

You must understand class structure and attribute inheritance to effectively perform these rich client tasks:

- Create closure rules in PLM XML Export Import Administration
- Create queries in Query Builder
- Create property sets in Report Builder
- Create cubes in Teamcenter's reporting and analytics

You also must understand business object structure and property inheritance to perform these Business Modeler IDE tasks:

- Create Generic Relationship Management (GRM) rules
- Create compound properties

Secondary

Instructor Note:

Topic instructor notes.

Classes, attributes, and compound properties

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

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"/>
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"/>

◀

|||

▶

⬆

⬇

⬆

Add

Modify

Remove

Class attributes table

When you right-click a class in the **Classes** view of the Business Modeler IDE and choose **Open**, the attributes of the class appear in a table.

Column	Description
Attribute Name	Displays the property name for the attribute.
Storage Type	<p>Displays the storage type of the attribute:</p> <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. 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

Column	Description
	An integer number without decimals from 1 to 9999.
	<ul style="list-style-type: none">• String A string of ASCII characters.• TypedReference Points to a Teamcenter class.• UntypedReference Points to any class of data.
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.
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.

Column	Description
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>

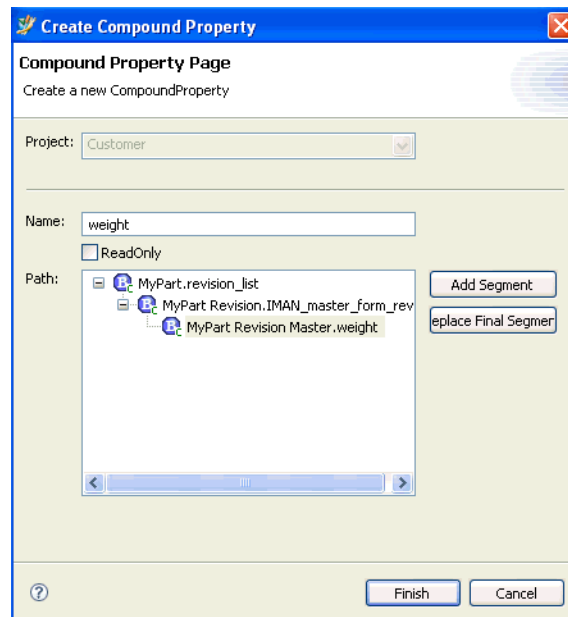
Working with compound properties

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.

Activities

In the *NNNN* section, do the following activities:

- Create a naming rule.
- Create compound property rule.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. X or x
2. False

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).

Extensions overview

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

Generic Relationship Manager

The Generic Relationship Manager (GRM) provides a general way in which two objects can be associated via a relationship. For example, a dataset can be related to an item revision with the specification relationship. You can navigate from object to object using attributes that are references, as well as using GRM. References and GRM can be used in both directions. In some cases, GRMs are exposed as properties.

The GRM module of the Integration Toolkit supports the concept of explicit relationships. With GRM, you can define and enforce specific rules pertaining to relationships, as well as separate the maintenance of relationships from the data itself.

Secondary

Instructor Note:

Topic instructor notes.

Constants introduction

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

Introduction to constants

The **Constants** folder in the **Extensions** view is used for working with *constants*, 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:

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

Instructor Note:

Similar to Engineering site preferences except:

- Configured in the Business Modeler IDE and stored in a Business Modeler IDE template.
- Deployed to all sites where the template is deployed, rather than configured on a per user basis.
- Each template can apply its overrides to a previous template.

Similar to Enterprise class constants except:

- Available as global constants and property constants as well as type constants.
- More robust as the owner can define scope of usage, a list of valid choices, and define protected values that can not be overridden.

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.

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:

- **Complex property**
Specifies a combination of properties and constant strings to assign a value to a selected property.
- **Enabled**
Defines whether a property is enabled in the interface.
- **Exportable**
Defines if a business object property can be exported using PLM XML.
- **Initial value**
Specifies the initial value to be assigned to a property when the corresponding object is created.
- **Modifiable**
Places restrictions on the modifiability of an object property.
- **Required**
Indicates whether a value must be entered for a specific object property.
- **Visible**
Specifies 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** business object.

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.

Working with property constants

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. Some examples are:

- Set whether the property is required.
- Set the initial value of the property.

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.

Activities

In the *Data Model* section, do the following activities:

- Modify property constants.
- Deploy and test updated constants.

Review questions

1. Which constant provides consistent definitions that can be used throughout the system.

Select one.

- Business object
- Class
- Global
- Property

2. Teamcenter provides a property constant for setting a property's initial value.

- True
- False

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Global
2. True

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*)

Instructor Note:

Lesson instructor notes.

Summary

The following topics were taught in this lesson:

- Constants are configuration points within the data model.
- Global constants provide consistent definitions that can be used throughout the system.
- Business object constants provide default behavior to business objects.
- Property constants provide default behavior to business object properties.
- You can override an existing constant by setting a different value in the current template.
- You must add the code on the server to return the constant's value to the caller.

Instructor Note:

Summary instructor notes.

Summary

The following topics were taught in this lesson:

- Extend the **Item** business object with a custom item.
- Locate and navigate business objects.
- Create new storage classes.
- Create forms and modify form properties.
- Hide form properties.

Instructor Note:

Summary instructor notes.

Main topic template

Primary

Secondary

Instructor Note:

Topic instructor notes.

Subtopic template

Primary

Secondary

Instructor Note:

Topic instructor notes.

My Modifier 1

Primary

Secondary

Instructor Note:

Topic instructor notes.

My Modifier 2

Primary

Secondary

Instructor Note:

Topic instructor notes.

Lesson

6 *Item business object configuration*

Purpose

The purpose of this lesson is to ...

Objectives

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

-
-

Help topics

Additional information for this lesson can be found in:

- [*Getting Started with Teamcenter 2007*](#)
- [*Business Modeler IDE Guide*](#)

Instructor Note:

Lesson instructor notes.

Data model custom introduction

You need to decide if the topic type will be just primary (the default setting) or a split screen for the presentation. It would be an exception to ever set the topic type to secondary.

- Use complete sentences here. Be consistent.
-

Instructor Note:

Topic instructor notes.

Configuring a new item business object and properties

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

Extending item business objects

The Business Modeler IDE can be used to define more **Item** business objects in addition to those provided with base Teamcenter).

Reasons for extending the **Item** business objects are:

- Having more types of **Item** business objects 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.
- Different rules for naming conventions, deep copy rules, and so on, can be configured for one type of **Item** business object compared to another type.
- Default process model association for one type of **Item** business object versus another is easier to implement.
- Different designs for the **Item Master** and **ItemRevision Master** forms may be desired. Each type of **Item** business object can have unique and different master form definitions.

In the following example, a new type of **Item** business object (**EndItem**) has been defined so that the customer can define customer specific attribute data that Teamcenter stores for this kind of data.

EndItem

EndItem Master (Form)

EndItem Revision

EndItem Revision Master (Form)

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 or its children when you want to create business objects to represent product parts. When you create a business object using **Item** (or one of its children) 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 file where you want to save the data model changes, for example, **business_objects.xml**.

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 or one of its children 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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

- c. The **Parent** box displays the business object you already selected as the parent business object.
- d. In the **Description** box, type a description for the new business object.
- e. The **Advanced** check box is selected by default to show the **Storage Class** pane.
 - 1) Select **Create primary Business Object** to make a new class that stores the data for the new business object. Clear this option to set the storage class as the same used by the parent business object.
 - 2) Select **Uninstantiable** if instances of the business object will not be created in user interfaces such as the rich client. For example,

you may want to select this if the business object is intended as a folder for children business objects.

- f. Click the **Add** button to the right of the **Properties** table to add a property to the business object.

The New Property wizard runs.

- g. Click **Next**.

6. 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.

- a. In the **Description** box, type a description for the new business object revision.

- b. Click the **Add** button to the right of the **Properties** table to add a property to the revision business object.

The New Property wizard runs.

- c. Click **Finish**.

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

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 **business_objects.xml** file, the changes are saved in that file.

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

9. 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.

Note

If you selected the **Uninstantiable** check box during creation of the new business object, you cannot create an instance of the business object.

Activities

In the *Data model custom* section, do the following activities:

- Create CCC_CommonItem business object.
- Create CCC_SampleItem business object.
- Add properties to the File® New® Item wizard.
- Deploy the business object.
- Verify the new item business object.

Review questions

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

Select one.

- 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.

- None
- One
- Two
- Four

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Business object model
2. Are the most common business objects created.
Uses the storage class of its parent business object.
3. Two

Extensions overview

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

Basic tasks using the Business Modeler IDE

Whenever you extend the data model using the Business Modeler IDE, you follow this process. (Before performing any of these tasks, you must have already created a project.)

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

Right-click a project and choose **Organize**→**Set active extension file**.

2. Perform the extension work.

For example, create a new business object, list of values, and so on.

3. Save your work.

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

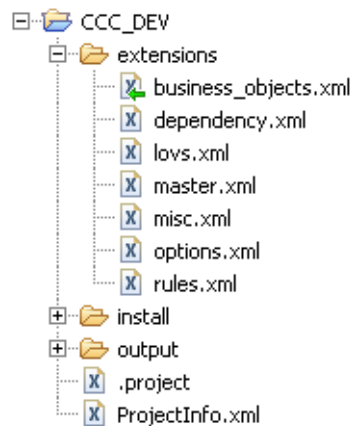
4. Verify your extensions by deploying them to a Teamcenter server.

Right-click a project and choose **Deploy Template**.

After you verify your extensions, you can package your data model into a template that can be installed on a production server.

Setting active extension files

You can set the extension file by right-clicking a project and choosing **Organize→Set active extension file**. A green arrow symbol appears on the active extension file (see the following figure). You can move a custom object to another extension file by right-clicking a project and choosing **Organize→Move to extension file**.



Extension files

Before you extend the data model, choose the extension file where you want your work to be saved.

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.

Defining commonly used 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.
- **Form**
Displays properties.
- **Dataset**
Represents file types.

Note

When you create a child of the **Item** business object, a child **ItemRevision** business object is created automatically. You cannot create a child of the **ItemRevision** object directly.

Instructor Note:

Topic instructor notes.

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

ItemMaster (Form)

ItemRevision

ItemRevision Master (Form)

- **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 **ItemRevision** object to include data unique to the customer.

Note

In Teamcenter 8 MP1 the master forms are created even though properties should be added directly to the **Item** and **ItemRevision** business objects.

Data model extension example

Beginning with Teamcenter 8 MP1 properties should be added directly to the new business objects.

Business objects	Classes
 Item  MyPart <i>New properties:</i> safety_code supplier	 Item  MyPart <i>New attributes:</i> safety_code supplier
 ItemRevision  MyPart Revision <i>New properties:</i> material_code material_description weight	 ItemRevision  MyPart Revision <i>New attributes:</i> material_code material_description weight

In this example, by extending the business object, the class is automatically extended.

- MyPart business object was created under Item.
- MyPartRevision business object was created under ItemRevision automatically.
- A primary business object has the same name as its associated storage class.
- Beginning with Teamcenter 8 MP1, new properties are added to the new business object(s) and indirectly associated storage class gets new attributes.

Typically, most extensions to the existing business object model are primary business objects.

Defining business objects

To extend the **Item** business object

Note

When you create a child of the **Item** business object, a child **ItemRevision** business object is created automatically. You cannot create a child of the **ItemRevision** object directly.

To add properties to the **CCC_Item** business object

Property Name	Storage Type	Reference Class

Instructor Note:

Topic instructor notes.

Add a persistent property

This is a basic outline to add a persistent property to a business object.

- From the **Business Object** view, click the **Find Business Object** button to locate the business object to which you want to add the property.
- Right-click the business object, choose **Open**, and click the **Properties** tab in the resulting view. The properties of the business object appear in a table.
- Click the **Add** button to the right of the properties table. The Business Modeler IDE runs the New Property wizard.
- Under **Property Types** select **Persistent**.
- Enter values in the **Persistent Property** dialog box to define the property.
- When done making changes, click **Finish**.

The new property appears in the properties table and is marked with a **c** indicating it is a custom property.

Note

You can add persistent properties to COTS and custom business objects.

Instructor Note:

Topic instructor notes.

There are many various properties that can be added to a business object, but this page is purposely kept brief to just show the steps. You may want to give a demo. Another page has more example properties, but even that page does not have all variations. Have the students refer to the Business Modeler IDE for all possible property types.

Create a persistent property

The **Persistent Property** dialog box for a **String** persistent property.

Details to define a persistent property

The following is an example values to enter in the **Persistent Property** dialog box for a **String** persistent property:

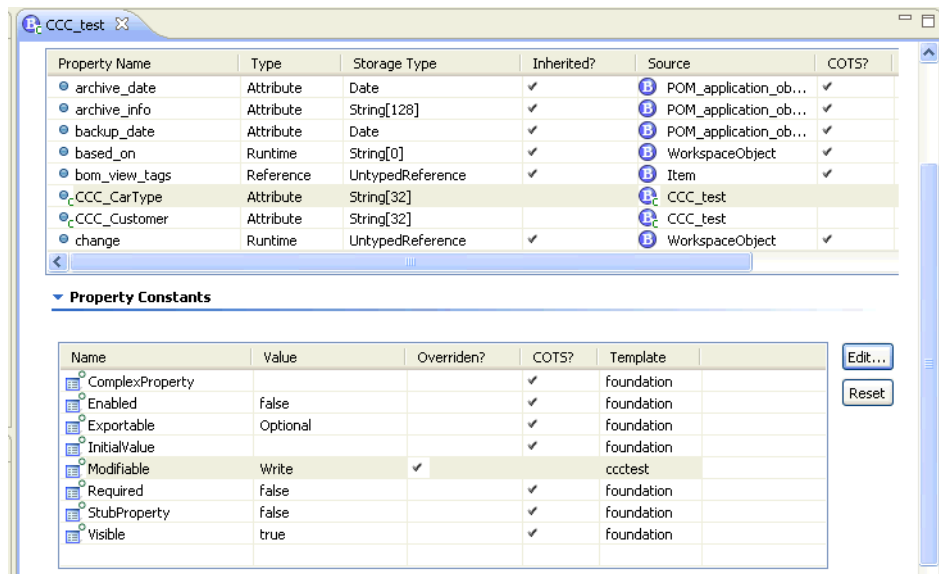
Name	The property name cannot contain spaces. When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.
Attribute Type	Select the storage type for the attribute, for example, String .
String Length	If the attribute is a String attribute, type the character length of the attribute.
Keys	Keeping the check in Nulls Allowed will allows an empty value for the attribute.

Instructor Note:

Topic instructor notes.

Property constants

Property constants allow you to control whether a business object property is modifiable, visible, enabled, exportable, or required. Property constants defined on parent business objects are inherited by sub-business objects.



The following constants can be applied to business object properties:

- Complex property** Specifies a combination of properties and constant strings to assign a value to a selected property.
- Enabled** Defines whether a property is enabled in the interface.
- Exportable** Defines if a business object property can be exported using PLM XML.
- Initial value** Specifies the initial value to be assigned to a property when the corresponding object is created.
- Modifiable** Places restrictions on the modifiability of an object property.
- Required** Indicates whether a value must be entered for a specific object property.
- Visible** Specifies whether a specific object property is visible in the interface.

Note

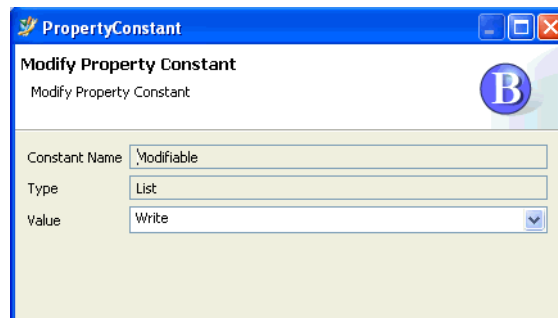
Access Manager (AM) rules take precedence over property constants. For example, if you define a property constant stating that an object property is modifiable, but the user has not been granted write access to the object, the property remains read-only and the user cannot modify it.

Instructor Note:

Topic instructor notes.

Modifiable property constant

The modifiable constant allows you to place restrictions on the modifiability of an object property.



The following restrictions can be applied:

Note

The default for a new property on a new business object is **Read**.

- **Read**

Allows users to view the value of the property but does not allow them to modify the value.

- **Write**

Allows users to modify the value of the property if they have write access to the object to which the property belongs.

- **WriteIfNull**

Allows users to modify the property only if the existing value is null, or empty.

Instructor Note:

Topic instructor notes.

Deploy with a two-tier server connection profile

To deploy to the Teamcenter server, right-click a project and choose **Deploy Template**. In the two-tier server connection profile in the Business Modeler IDE, choose IIOP as the protocol.

Tip

The IIOP connection information is obtained from the following file:

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

RELEASE represents the current product release number and equates to something like 8000.0.0.

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

Configuring a compound property

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

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

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 to display a custom property from a form on a business object.

Note

You can add compound properties to COTS and custom business objects.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. Click the **Business Objects** tab to display the **Business Objects** view.
3. Select the project in which you want to make the change. 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, **business_objects.xml**.
4. In the **Business Objects** view, browse to the business object to which you want to add the compound property. To search for a business object, you can click the **Find Business Object** button at the top of the view. 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. Click the **Add** button to the right of the properties table.

The Business Modeler IDE runs the New Property wizard.

6. Under **Property Types**, select **Compound**. Click **Next**.
7. 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.

- b. Select the **ReadOnly** check box if you do not 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 **Choose a property** message.

8. 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. Use the **Reference**, **Relation**, and **Runtime** check boxes to filter out the kind of properties to display.

Note

If you have added a relation business object to the **ImanRelation** business object, it does not appear in this list until you add the new business object as a relation property. To add the relation business object as a relation property, go back to the **Properties** tab, click the **Add** button to the right of the property table, select the **Relation** check box on the **Property Definition** dialog box and click **Next**, and on the **Relation Property** dialog box click the **Browse** button to the right of the **Relation Business Object** box and choose the relation business object. Now you can go back and add this new relation property as a segment on the compound property.

- b. Click **Next**.

The **Compound Property Segment** dialog box displays the **Choose a business object** message.

- c. Select the business object you want to use for the next segment of the compound property. Use the **COTS** and **Custom** check boxes to filter the kind of business objects to display.

- d. Click **Finish**.

The **Compound Property Page** dialog box appears, showing the compound property path thus far.

9. To add more segments to the compound property, click the last segment in the **Path** pane and click **Add Segment**. To replace a segment, select the segment and click **Replace Segment**.
10. 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.

11. When you are done adding segments in the **Compound Property Page** dialog box, click **Finish**.

The new compound property appears in the property table.

12. 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.

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

14. After deployment, verify your new compound property 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.

Activities

In the *Data model custom* section, do the following activities:

- Modify property constants.
- Hide CCC_CommonItem master form properties.
- Add a property to the CCC_SampleItemRevision.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. X or x
2. False

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).

Configuring a property constant

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

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' configuration window. The 'Form : MyPart Master' tab is active. Below the 'Main' and 'Properties' tabs, there is a table of properties. The 'Property Constants' section is expanded, showing a table of constants. The 'Visible' constant is highlighted, and the 'Modify' button is visible.

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"/>

Name	Value	Overridden	COTS	Template	
Enabled	false	<input type="checkbox"/>	<input checked="" type="checkbox"/>	foundation	<input type="button" value="Modify"/>
Exportable	Optional	<input type="checkbox"/>	<input checked="" type="checkbox"/>	foundation	<input type="button" value="Reset"/>
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

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 file where you want to save the data model changes.
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. The **Project** box defaults to the already-selected project.
 - b. In the **Name** box, type the name you want to assign to the new constant.

When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

- c. In the **Description** box, type an explanation of how the constant is to be used.
 - d. Click the **Add** button to the right of the **Scope** box.

Perform the following steps in the **Business Object and 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 children 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_desc**
The constant can be applied only to the **object_desc** property on the **Item** business object and its children.
- e. If desired, click the **Add** button to the right of the **Scope** box to narrow the scope further.
- f. Click the arrow in the **Data Type** box to select 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.
- g. If you selected 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**.
- h. 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 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.

- i. Click **Finish**.

The new constant appears under the **Property Constants** folder in the **Extensions** view.

Note

You can also see the property constant on the business object it is applied to. Right-click the business object, choose **Open**, and 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.
7. Deploy your changes to the server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project and 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.

Activities

In the *Data model custom* section, do the following activities:

- Modify property constants.
- Deploy and test updated constants.

Review questions

1. Which constant provides consistent definitions that can be used throughout the system.

Select one.

- Business object
- Class
- Global
- Property

2. Teamcenter provides a property constant for setting a property's initial value.

- True
- False

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Global
2. True

Summary

The following topics were taught in this lesson:

- Extend the **Item** business object with a custom item.
- Locate and navigate business objects.
- Create new storage classes.
- Create forms and modify form properties.
- Hide form properties.

Instructor Note:

Summary instructor notes.

Configuring new business object rules

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

Configuring a new dataset business object

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

Secondary

Instructor Note:

Topic instructor notes.

Extensions overview

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

Naming objects

The Business Modeler IDE performs a validation at the time you create an object to ensure its name is unique. You are not allowed to create duplicate objects with the same name. This includes objects names that differ only in their case (uppercase or lowercase).

When you create new data model objects, Siemens PLM Software recommends you create a prefix to designate the objects as belonging to your organization, such as an acronym. For example, when you create a new **Item** class, you could name it **a3_Item**. This prevents naming collisions in future versions if you name an object the same as a Teamcenter object in a future release. Siemens PLM Software recommends you apply this consistent naming convention to all your new data model.

The following is the policy that you should use when choosing a prefix:

- Use the prefix on all class names and business object names within a template.
- Ensure the prefix contains two to four characters.
- Place a letter in the first character position.
- Place a digit in character position two, three, or four.
- You can use digits 3-9. Do not use digits 0, 1, or 2, because 0 is reserved for Siemens PLM Software templates, and 1 and 2 are reserved for third-party template development.

Following are examples of acceptable prefixes:

A3
ab5
A5C
c4
B9
F8_
x99_

You can rename a business object by right-clicking it in the **Business Objects** view and choosing **Rename**.

Note

For each class there is a database table with the same name. Each attribute on the class equates to a column in a database table. Database tables must be uniquely named so that data can be properly mapped from the application to the database. Database systems such as Oracle convert all table names to uppercase. This means that the Teamcenter class spelled **ItemRevision** is mapped to a table spelled **ITEMREVISION** in the database. Because all table names are converted to uppercase, in Teamcenter you cannot define two class names that differ only in casing, because both would result in the same table space name. Because every class has a business object, business objects must also follow the same convention for uniqueness.

Instructor Note:

Topic instructor notes.

Lesson

7 *Form business object configuration*

Purpose

The purpose of this lesson is to .

Objectives

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

- Define a

Help topics

Additional information for this lesson can be found in:

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

Instructor Note:

Lesson instructor notes.

Configuring a new form business object

Primary

REVIEW NOTE

Issue: DK: Move next sections for fall under this topic.

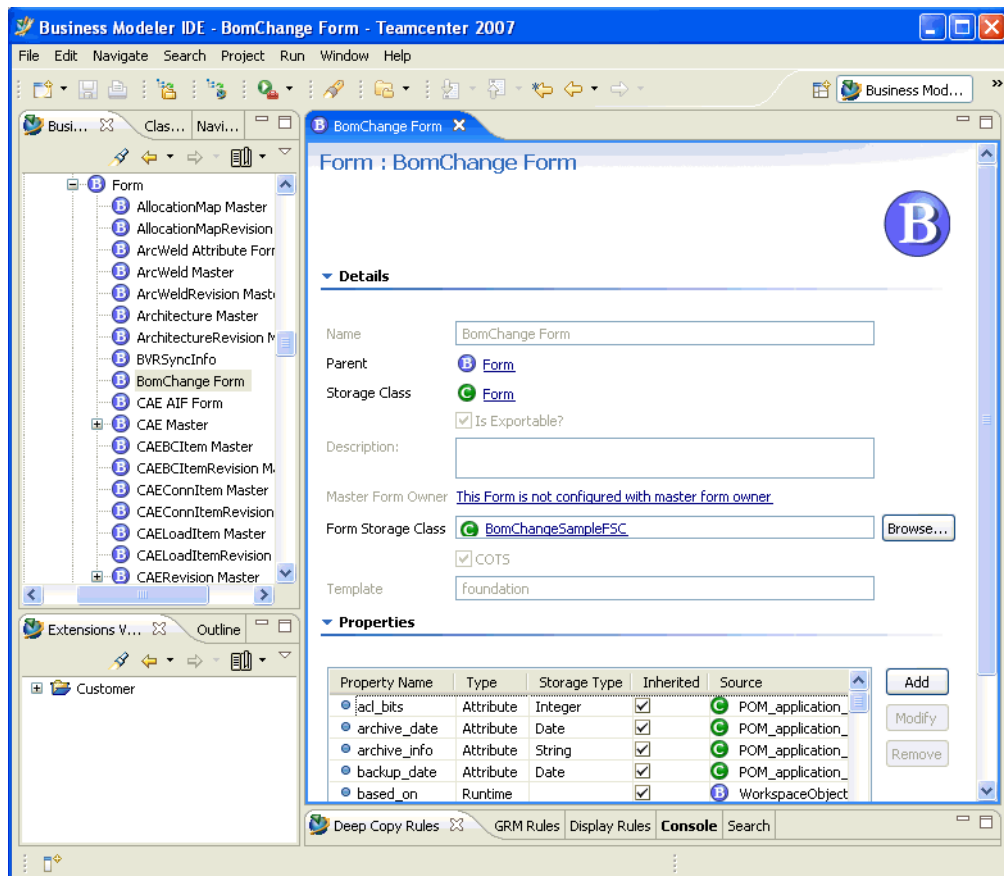
Secondary

Instructor Note:

Topic instructor notes.

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 or its children 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 use 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 file where you want to save the data model changes, for example, **business_objects.xml**.
3. In the **Business Objects** view, right-click the **Form** business object or one of its children 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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.
 - c. The **Parent** box displays the business object you already selected as the parent business object.
 - d. In the **Description** box, type a description for the new business object.
 - e. 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 if you want to select a different class you want to be the parent of the new form storage class.

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

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

- f. Click the **Add** button to the right of the **Properties** table to add a property to the business object.

The New Property wizard runs.

- g. 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 the project in the **Business Objects** view and choose **Deploy Template**, or select the project and 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	Logical data model
 Form  MyPart Master	 Form
Properties: safety_code supplier	
 POM_object  MyPartMaster	 POM_object  MyPartMaster
Properties: safety_code supplier	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.

The screenshot shows the 'Form : BomChange Form' configuration window. It has tabs for 'HTML', 'BomChange Form', and 'Item'. The 'Details' section includes fields for Name (BomChange Form), Parent (Form), Storage Class (Form), Is Exportable? (checked), Description, Master Form Owner (This Form is not configured with master form owner), Form Storage Class (BomChangeSampleFSC), COTS (checked), and Template (foundation). The 'Properties' section contains a table with columns for Property Name, Type, and Storage Class, and buttons for Add, Modify, and Remove.

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 configuration window. The 'Properties' pane is active, displaying a table of properties. Below it, 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.

Activities

In the *Forms* section, do the following activities:

- Create CCC_ERPDataForm business object.
- Import CCC_ChangeForm and CCC_ProductImpact definitions.
- Add a property to the CCC_ChangeForm.
- Verify the new form business objects.

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.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. An item can contain multiple forms.

Form properties are inherited from the form's parent.

2. **POM_object**

Configuring a new class

Primary

REVIEW NOTE

Issue: **DK:**

Secondary

Instructor Note:

Topic instructor notes.

Topics Held for Inserting Later

Primary definition.

Secondary definition.

Instructor Note:

Topic instructor notes.

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

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 file where you want to save the data model changes, for example, **business_objects.xml**.
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 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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.
 - c. The **Parent** box displays the already-selected parent class.
 - d. Select the **Exportable** check box if the class can be exported using PLM XML.
 - e. Select the **Uninheritable** check box if the class cannot have children classes.
 - f. Select the **Uninstantiable** check box if instances of the class will not be created in user interfaces such as the rich client. For example, you may want to select this if the class is intended as a folder for children classes.
 - g. Click the 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 if you want 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 symbol 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 the project in the **Classes** view and choose **Deploy Template**, or select the project 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.

Add or change attributes on classes

Attributes are item characteristics, such as name, number, description, and so on. You can add or modify attributes on custom classes, and some COTS classes.

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 file where you want to save the data model changes, for example, **business_objects.xml**.
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 **Class Attribute** dialog box:

- a. In the **Name** box, type the attribute name.
When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an 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 Length** 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. Select **Set Initial Value to NULL** if you want the default value to be a null value.
- f. If you did not select **Set Initial Value to NULL**, in the **Initial Value** box, type the value to populate the attribute.
- g. In the **Lower Bound** box, type the lower numerical limit for a numerical or alphanumerical attribute.

- h. In the **Upper Bound** box, type the upper numerical limit for a numerical or alphanumeric attribute.
- i. In the **Array Keys** section, check the properties that apply:
- j. In the **Keys** section, check the properties that apply:
- k. 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 **Edit** 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.

Note

There is no limit on the number of attributes you can add to a class when you use an Oracle or SQL database. However, there is a limit using DB2 due to the fact that a row has to be in one block with a 32K limit; so the number of attributes allowed depends on the total all of the attribute sizes. As a rule, do not use over 100 attributes of any type.

- 8. To save the changes to the data model, choose **File**→**Save Data Model**, or click the **Save Data Model** button on the 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 test server. Right-click the project in the **Classes** view and choose **Deploy Template**, or select the project 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 custom subclass of the **Item** class, in the My Teamcenter application, select a business object that inherits from that custom class and view its attributes in the **Viewer** tab.

Activities

In the *Additional business object configuration* section, do the following activities:

- Add operation on CCC_ItemRevision to auto-create a form.
- Create a new Relation business object.
- Verify the new Relation business object.

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**

Instructor Note:

Activity instructor notes.

Answers to review questions

1. False
2. **POM_application_object**
POM_object
WorkspaceObject

<Type Name>_viewerprops

DESCRIPTION

Customizes identifier property columns of the viewer tab in the rich client interface.

Valid types for this preference are any identifier types.

Values are set in a triplet, determining the property configured, whether its field is optional or mandatory, and whether its field is enabled or disabled.

For example:

```
Identifier_viewerprops=  
IdContext,O,E
```

VALID VALUES

Accepts one or more strings as values; each string must contain one value from the three following components:

Property name

The property must be valid for the type for which the preference is being defined.

Optional or mandatory value:

O

Optional field.

M

Mandatory field. These fields are marked by red in the upper right corner, visually indicating that input is required.

Enabled or disabled value:

E

Enabled field.

D

Disabled field. Initializing a value for a property, then disabling the field allows you to specify an initial value in the field that users cannot modify.

DEFAULT VALUES

None.

SCOPE

Site preference.

Summary

The following topics were taught in this lesson:

- Define the .

Instructor Note:

Summary instructor notes.

Lesson

8 *Relation business object configuration*

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*)

Instructor Note:

Lesson instructor notes.

Summary

The following topics were taught in this lesson:

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

Instructor Note:

Summary instructor notes.

Lesson

9 *LOV (list of value) extensions*

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*)

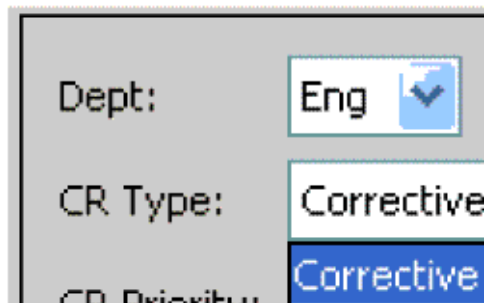
Instructor Note:

Lesson instructor notes.

Introduction to lists of values (LOVs)

The **LOV** folder in the **Extensions** view is used for working with *lists of values* (LOVs). LOVs are pick lists of data entry items. They are commonly accessed by Teamcenter users when they click an arrow in 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. The user clicks the LOV button and an object selection list displays allowable entries for that property.



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), and Reference. There are buttons for 'Browse...', 'Add', 'Remove', 'Modify', 'Move Up', 'Move Down', 'Clear', and 'Load'. A 'Show Cascading View' checkbox is also present. Below the details is a table with 'Value' and 'Description' columns, containing 'Racing' and 'Show'. At the bottom is the 'LOV Attachments' section with a table showing 'Business Object:Property(Property)', 'COTS', and 'Template' columns. The first row is 'CCC_Car Master.CCC_Make' with COTS checked and Template 'customer'. Buttons for 'Attach', 'Detach', and 'Edit' are on the right.

Value	Description
Racing	
Show	

Business Object:Property(Property)	COTS	Template
CCC_Car Master.CCC_Make	<input checked="" type="checkbox"/>	customer

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 in 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 file where you want to save the data model changes, for example, **lovs.xml**.

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

When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.
 - c. In the **Description** box, state the purpose of the LOV.
 - d. Click the **Browse** button to the right of the **Type** box to select the LOV business object for the LOV you want to create.

Note

Entries containing **Extent** must match an existing value in the database.

- **ListOfValuesChar**

Single ASCII character. Only alphabetic characters can be used (for example, A-Z). You cannot use any other kinds of characters (numbers, punctuation, and so on).

- **ListOfValuesDate**

Date and time in the format used at your site.

- **ListOfValuesDouble**

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

- **ListOfValuesExternalCharExtent**

Single ASCII character in the database.

- **ListOfValuesExternalDateExtent**

Date and time in the database.

- **ListOfValuesExternalDoubleExtent**

Double-precision, floating-point decimal number in the database.

- **ListOfValuesExternalIntExtent**

Whole number in the database.

- **ListOfValuesExternalStringExtent**

A string in the database.

- **ListOfValuesFilter**

Reference to another nonfiltered LOV with the capability to filter the referenced LOV's values.

- **ListOfValuesInteger**

Whole number.

- **ListOfValuesIntegerExtentSite**

A site name in the database. Valid values are all existing instances of the selected site.

- **ListOfValuesIpClassification**

An intellectual property classification.

- **ListOfValuesString**

String of ASCII characters.

- **ListOfValuesStringExtent**

A string of ASCII characters in the database. Valid values are all existing instances of strings.

- **ListOfValuesStringExtentGrName**

A group name in the database. Valid values are all existing instances of group names.

- **ListOfValuesStringExtentPubrType**
Workspace object in the database. Valid values are all existing workspace object types.
 - **ListOfValuesStringExtentStatus**
Status in the database. Valid values are all existing release status names.
 - **ListOfValuesStringExtentUserId**
A user ID in the database. Valid values are all existing instances of the active **user_id** attribute in the **POM_user** class.
 - **ListOfValuesStringExtentUsName**
A user name in the database. Valid values are all existing instances of the active **user_name** attribute in the **POM_user** class.
 - **ListOfValuesStringExtentWSOClass**
Workspace object class in the database. Valid values are all existing instances of the selected class.
 - **ListOfValuesTagExtent**
Reference to a unique tag in the database.
 - **ListOfValuesTagRDVSearchRevRule**
Repeatable Digital Validation (RDV) tag.
- e. 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.
- f. The **Reference** box appears if you selected string, tag, or tag extent types. Click the **Browse** button to the right of the **Reference** box to select the class and attribute to use for the LOV values. You can use this to obtain values for the LOV from a class attribute, such as **item_id**.

Note

If a class attribute is specified for a string usage or tag LOVs, click the **Load** button in the lower right of the dialog box 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.

- 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. Click the **Add** button to the right of the **LOV Values** table.
The Add LOV Value wizard runs.
- i. Perform the following steps in the **Create** dialog box:
 - 1) In the **Value** box, type the value of the LOV.

Caution

Special characters are limited to those supported by the ASCII character set only. Copy and pasting characters from Microsoft Word into Business Modeler IDE boxes is an unsupported operation.

- 2) In the **Description** box, type a brief description of the LOV value.
 - 3) Click the **Browse** button to the right of the **Condition** box if you want the value to be available only if a certain condition applies. If you select **isTrue** as the condition, the LOV always applies.
 - 4) Click **Finish**.

The value is added to the **LOV Values** table.

- j. Change the LOVs as desired by using the **Remove**, **Edit**, **Move Up**, **Move Down**, and **Clear** buttons.

If you used the **Reference** box, click the **Load** button to look up values on the server.

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

- k. 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.

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 file where you want to save the data model changes, for example, **lovs.xml**.

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 Attachment** dialog box:
 - a. Click the **Browse** button to the right of the **Property** box and choose the property to which you want to attach the LOV.
 - b. Click the **Browse** button to the right of the **Condition** box if you want the value to be available only if a certain condition applies. If you select **isTrue** as the condition, the value always applies.
 - c. Select the **Override** check box if you want to override attachment of the parent business object. Override can be set only with **isTrue** condition.
 - d. Click **Finish**.

The LOV is attached to the property on all the business objects that use that property. The business objects and properties appear in the **Property Attachments** table for the LOV.

7. Click the **Attach** button to attach the LOV to another property.

Click the **Detach** button to detach the LOV from a business object.

Click the **Edit** button to attach an LOV value description and sub-LOV values to a cascading LOV. This is known as an *interdependent 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.
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.

10. After you attach the LOV, you can deploy your changes to the server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project and 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 an 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.

- **Cascading LOVs (Hierarchical)**

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.

Filter LOV

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

Define the item master form attributes.

CCC_Model:

CCC_Make:

CCC_Location:

LT
XLT
Classic
Sport

General

This screenshot shows a form with three dropdown menus labeled CCC_Model, CCC_Make, and CCC_Location. The CCC_Location dropdown is open, displaying a list of car models: LT, XLT, Classic, and Sport. The General tab is selected at the bottom.

Filtered LOV

Define the item master form attributes.

CCC_Model:

CCC_Make:

CCC_Location:

LT
XLT

General

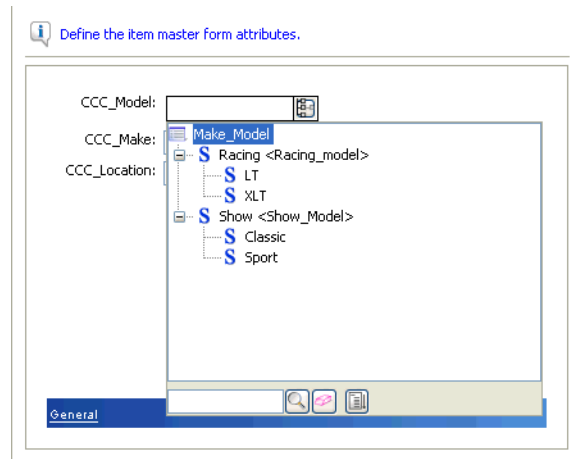
This screenshot shows the same form as the Unfiltered LOV, but the CCC_Location dropdown is filtered to only show LT and XLT. The General tab is selected at the bottom.

Cascading LOV

Cascading LOVs (also known as Hierarchical) 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

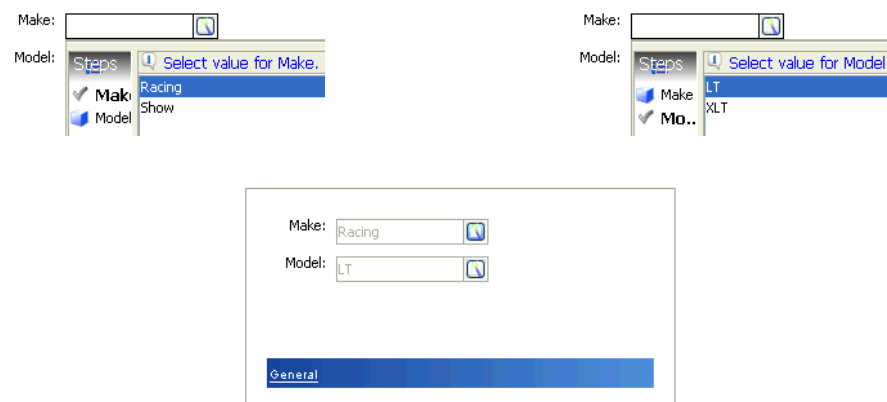
The list contains all car models organized by make, then by model.



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

You can create a list of values (LOV) based on another LOV by using the **Filter** LOV type.

1. Right-click the **LOV** folder and choose **New LOV**.

The New LOV wizard runs.

2. In the **Name** box, type the name you want to assign to the new LOV.
3. In the **Type** box, select **ListOfValuesFilter**.
4. Click the **Browse** button to the right of the **Based on LOV** box to choose the LOV you want to base the filter LOV on.

The **Find LOV** dialog box appears.

5. Perform the following steps in the **Find LOV** dialog box:

- a. Select the LOV you want to base the new LOV on.

The values for the LOV display in the **Preview Values** pane.

Note

If no values display, you cannot select values to filter on.

- b. Click **OK**.

The values of the original LOV are displayed in the **LOV Values** table.

6. Select the **Direct Dependency** check box if you want all the new LOV values to be derived directly from the original LOV.
7. In the **Show** box, select **All** to show all the values of the original LOV or **Selected** to show only selected values.
8. In the **LOV Values** pane, select only the values you want to use.

Note

If you had previously selected **Direct Dependency**, it is cleared when you clear any values in the **LOV Values** pane.

9. Click **Finish**.

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

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

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.

New LOV

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

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 in the main LOV and choose **Add Sub LOV**.

In the **LOV Selection** dialog box, choose the sub-LOV for that value, and if you want the value to be available only if a certain condition applies, select the **Condition**.

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**.

Example cascading LOV

An example cascading LOV would be a list of car makes that each contain a list of models.

The screenshot shows the 'New LOV' dialog box with the following configuration:

- Project:** Customer
- Name:** ccc_models
- Description:** Classic Car Company car models
- Type:** String
- Usage:** Exhaustive (selected), Suggestive, Range
- Reference Class and Attribute:** Reference (empty), Browse...
- LOV Values:**
 - ☒ Show Cascading View
 - | Value | Description |
|--|-------------------------|
| [-] Best of Show(ccc_best_of_show) | Best of Show models |
| Cruiser | Cruiser show model |
| Low Rider | Low Rider show model |
| [-] High Performance(ccc_high_performance) | High Performance models |
| Drag | Drag racing model |
| Racer | Track racing model |
| | |
| | |
| | |
 - Buttons:** Add Sub LOV, Remove Sub LOV, Expand All, Collapse All
- Footer:** ? (help icon), Finish, Cancel

Create an interdependent LOV

You can create an interdependent attachment so that the user is prompted to enter values for each level in a cascading LOV.

1. Create a cascading LOV.
2. Right-click the cascading LOV in the **LOV** folder, 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. Ensure that the cascading LOV is attached to a property in the **LOV Attachments** table.
4. Select the attachment in the **LOV Attachments** table and choose **Edit**.
The **Interdependent LOV** dialog box appears.

5. Perform the following steps in the **Interdependent LOV** dialog box:
 - a. Select the property and click **Attach** to attach additional properties to it. (To remove properties, click **Detach**.)

You are building a tree in the table that represents each level of the cascading LOV. For example, if you have a cascading LOV of states that contains cities subLOVs, you can use this so that the end user selects the state and then is prompted to select the city.

For example, to create this states and cities example, attach the states LOV to the **Item Master** business object on the **user_data_1** property. Select the **user_data_1** attachment in the **LOV Attachments** table and choose **Edit**. In the **Interdependent LOV** dialog box, select **user_data_1** and attach **user_data_2** to it. The cities subLOVs are automatically attached to **user_data_2**.

- b. To add a property to describe the attachment, click **Attach Description**. (Click **Detach Description** to remove a description.)

This adds the description of the LOV to the value displayed to the user.

- c. Click **Finish**.
6. After you save the data model and deploy it to the server, you can test the interdependent LOV.

For example, if you used the states and cities example, test it in My Teamcenter:

- a. Choose **File**→**New**→**Item** and click **Next** on the New Item wizard.
- b. Assign a name and number to the item and click **Next**.

- c. In the **Define the item master form attributes** dialog box, click the button next to the **User Data 1** box to select the state, and click the button next to the **User Data 2** box to select the city.
- d. Click **Finish**.
- e. Open the form on the new item. You should see the **User Data 1** and **User Data 2** boxes. If you want to change the values in these boxes, click the **Check-Out and Edit** button.

Interdependent LOV Attachment

Interdependent LOV
Attach Interdependent LOV

Type:

Property	Attachments
Make	Make_Model
Model	Make_Model

Buttons: Attach, Detach, Attach Description, Detach Description

LOV:

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

Buttons: Finish, Cancel

Activities

In the *Lists of values* section, do the following activities:

- Create the CCC_Torque_lov.
- Attach the existing Make Buy LOV.
- Import LOV definition values.
- Update and attach the CCC_Effects_lov.
- Create a cascading LOV.
- Create an interdependent LOV.
- Test the new LOV definitions.
- Optional - LOV with values loaded from the Teamcenter.

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.
 - Exhaustive
 - Range
 - Suggestive
3. What type of LOV variation allows you to organize and present your lists in a hierarchy?
Select one.
 - Filtered LOV
 - Cascading LOV
 - Interdependent LOV

Instructor Note:

Activity instructor notes.

Answers to review questions

1. True
2. Exhaustive
3. Cascading LOV
Interdependent LOV

Summary

The following topics were taught in this lesson:

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

Instructor Note:

Summary instructor notes.

Lesson

10 Dataset business object configuration

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:

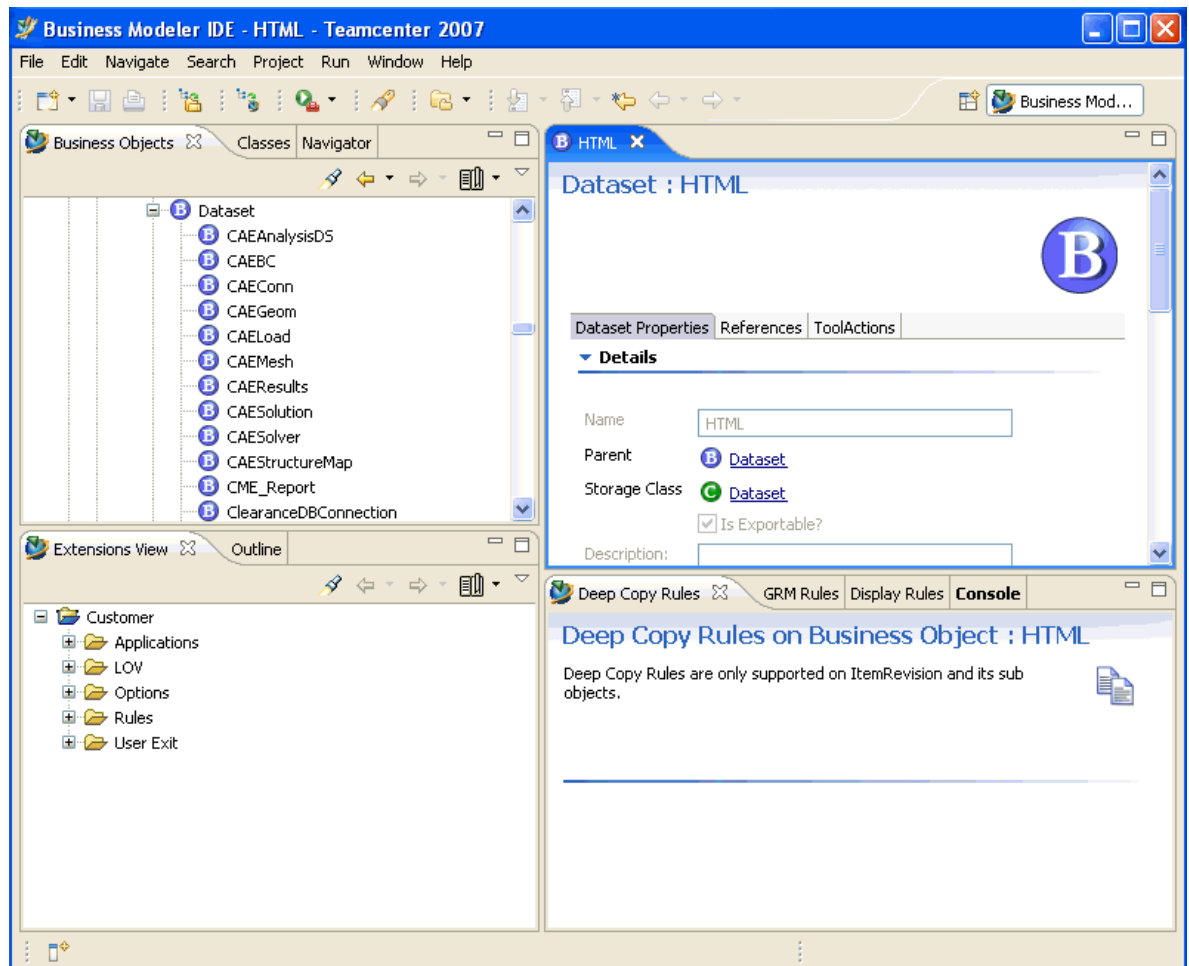
- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Instructor Note:

Lesson instructor notes.

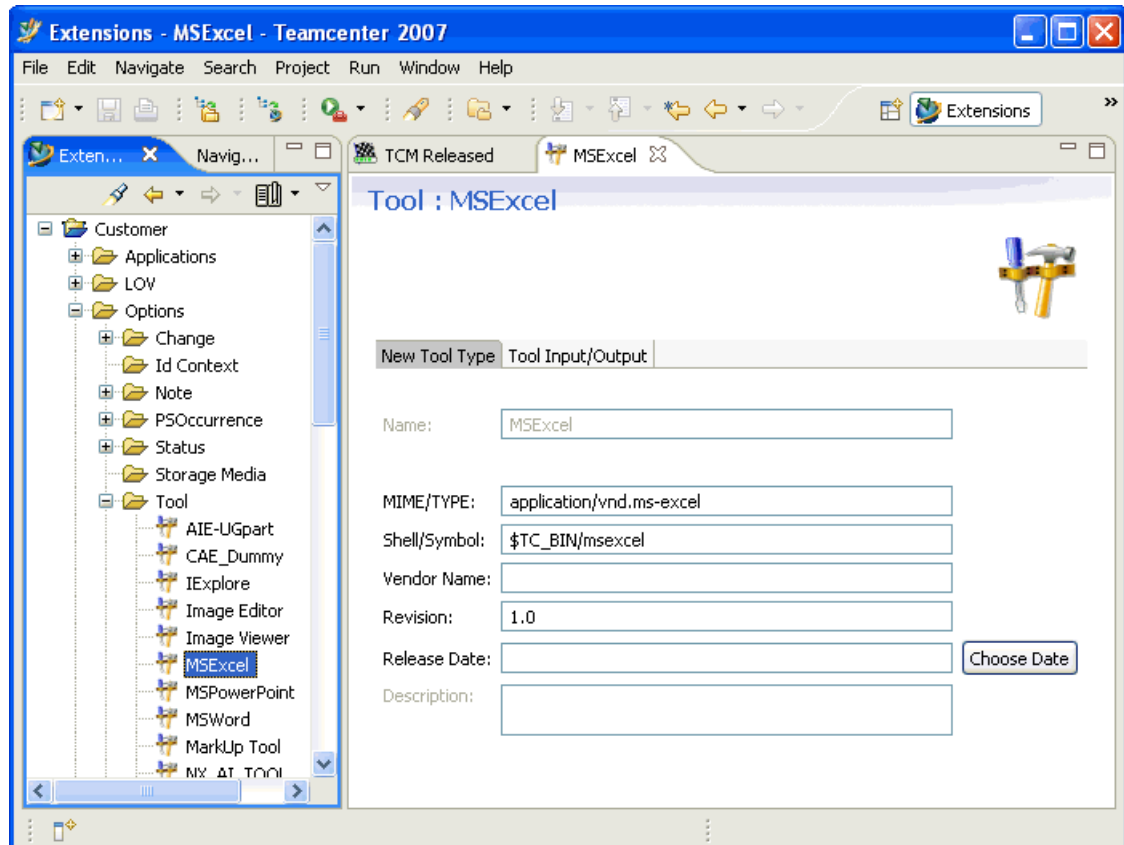
Introduction to dataset business objects

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 types to be able to manage your site's specific application data files and the viewing/editing software applications associated with these files.



Introduction to tools

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 file where you want to save the data model changes, for example, **options.xml**.
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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

- c. In the **MIME/TYPE** box, enter the kind of application association. For example, for Acrobat, type **application/acrobat**, or for Microsoft Word, type **application/msword**.

This setting is not required by Windows systems, because Windows systems use the file extension to determine the MIME type.

- d. In the **Shell/Symbol** box, type the full path and program name to be run in a shell. For example, for Microsoft Word, type **\$TC_BIN/msword**.

- 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.

Now the new tool is available for use in a dataset. When you create a new dataset business object in the Business Modeler IDE, 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.

You can associate a tool with 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 file where you want to save the data model changes, for example, **business_objects.xml**.

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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an 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. The **Advanced** check box is selected by default to show the **Storage Class** pane.
 - 1) Select **Create primary Business Object** to make a new class that stores the data for the new business object. Clear this option to set the storage class as the same used by the parent business object (in this case, the **Dataset** class).
 - 2) Select **Uninstantiable** if instances of the business object will not be created in user interfaces such as the rich client. For example,

you may want to select this if the business object is intended as a folder for children 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. 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 to add file name references to associate with the dataset.

The Add Dataset Reference wizard runs. 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** in 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 to modify the action that a software application takes when a dataset is launched.

The Add/Modify Dataset Tool Action wizard runs. 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. Select the **Export** check box 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 the project in the **Business Objects** view and choose **Deploy Template**, or select the project and 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

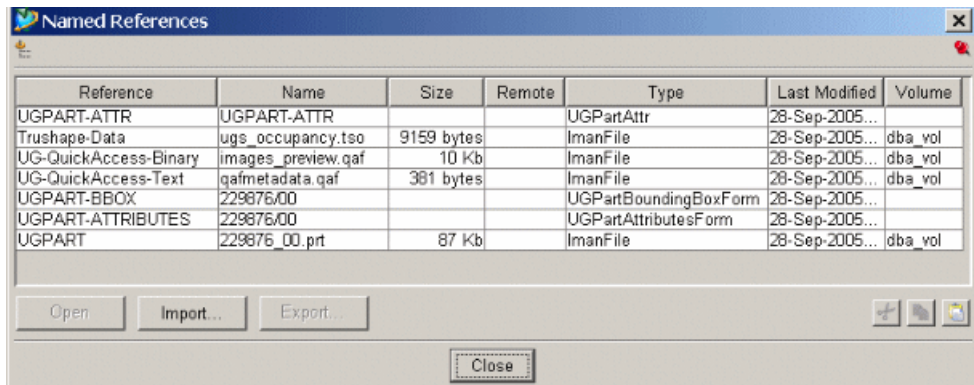
You can use both interactive and batch mechanisms to import data. You can import data to Teamcenter using:

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

Named references

Named references are files attached to a dataset object. A single dataset object may have one or more named references. To view the named references of a dataset from the Teamcenter rich client, in My Teamcenter, select the dataset and choose **View**→**Named References**, or right-click and choose **Named References**.

Named references are not to be confused with dataset references, which are the kind of applications associated with a dataset type.



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

Activities

In the *Datasets* section, do the following activities:

- Create a tool option.
- Create a new dataset business object.
- Test the new tool and dataset deployment.

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.

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

4. What is the batch mechanism to import data?

Select one.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Consists of a single application data file or logical groupings of application data files.

Objects used to manage file data associated with external software applications.

2. False
3. **Named References** are Teamcenter objects that relate to a specific data file. A single dataset object may have one or more named references.
4. Use the **import_file** utility to import files in the database.

Summary

The following topics were taught in this lesson:

- Purpose and implementation of new datasets.
- How tools support datasets.
- How to import files to the database.

Instructor Note:

Summary instructor notes.

Lesson

11 *Compound properties*

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*)

Instructor Note:

Lesson instructor notes.

Summary

The following topics were taught in this lesson:

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

Instructor Note:

Summary instructor notes.

Lesson

12 *Option extensions*

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:

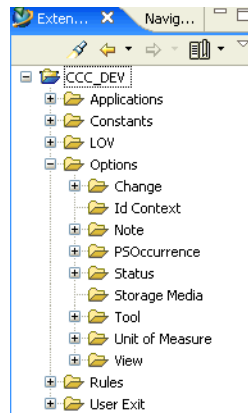
- [*Business Modeler IDE Guide*](#) (see *Using the Business Modeler IDE*)

Instructor Note:

Lesson instructor notes.

Introduction to options

The **Options** folder in the **Extensions** view is for working with *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 in a workflow, a view item holds structure information for a business object, and so on.

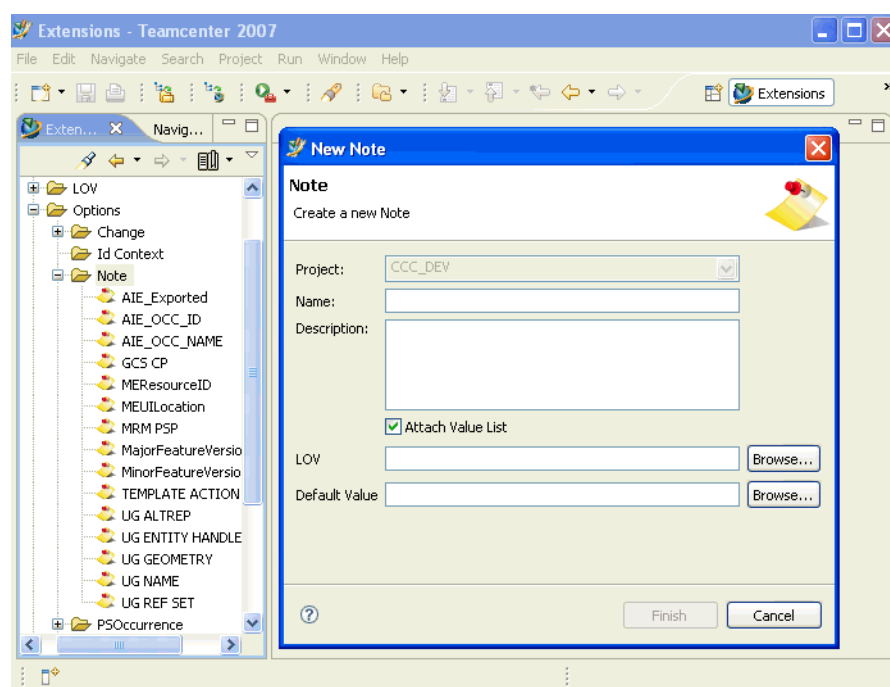


Introduction to notes

A note type is an object associated with a product structure occurrence in a Structure Manager bill of material (BOM). Note types support list of values, but the LOV must be of type **String**.

Users can specify a value for any note type that has been defined for the site. The occurrence note objects that are defined are listed in the Structure Manager **Columns**, **BOMLine Properties**, and **Notes Editor** dialog boxes. The user can use any of these dialogs to enter a value for a particular occurrence.

The initial list of note types shown are standard note types supplied with the system that are required for Teamcenter's manufacturing process management and for synchronizing object attributes from NX. These should not be deleted.



Add a note type

A note type is an object associated with a product structure occurrence in a Structure Manager bill of materials (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 file where you want to save the data model changes, for example, **options.xml**.
3. Expand the project and the **Options→List of Note Types** folders.
4. Right-click the **List of Note Types** folder and choose **New Note Type**.
The New Note Type wizard runs.

5. Perform the following steps in the **Note Type** 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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an 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).

LOV and **Default Value** boxes are displayed.

- 1) Click the **Browse** button to the right of the **LOV** box to locate the list of values to attach to the note.

Note

The LOV must be a **String** type because notes are string types.

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

- e. Click **Finish**.

The new note object appears under the **List of Note Types** 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.

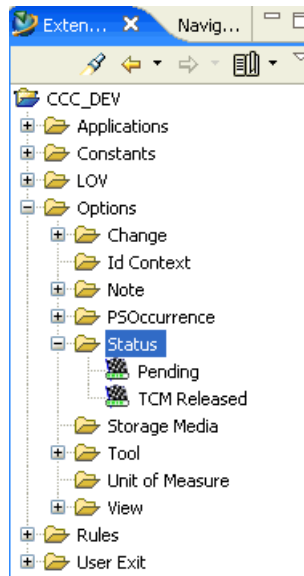
6. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button on the main toolbar.
7. Deploy your changes to the test server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project click the **Deploy Template** button on the main toolbar.
8. After deployment, test your note in the Teamcenter rich client by viewing **BOMLine** properties in Structure Manager:
 - a. In the My Teamcenter application, right-click any item or item revision and choose **Send To→Structure Manager**.
 - b. In the Structure Manager 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. Your new note appears near the bottom of the dialog box.

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

Introduction to 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, Structure Manager (Structure Manager) can be used to display different bills of materials (BOMs) in different stages of development.
- Structure Manager can also display Work-in-Process parts, which are those that have not had a final release status applied.
- Structure Manager (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 **Pending** 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 file where you want to save the data model changes, for example, **options.xml**.
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.
- c. In the **Description** box, type a description of the new status object.
- d. Click **Finish**.

When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

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.
7. Deploy your changes to the server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project and 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, select **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.
- f. Select your new status and close the **Attributes** dialog box.
- g. 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)**.

Introduction to units of measure

A *unit of measure* is a measurement category (for example, inches, millimeters, and so on).

By default, **Item** business objects have no units of measure (UOMs). 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).

Units of measure (UOMs) are created so that **Item** and **Item Revision** business objects can be expressed in standardized units (for example, inches, millimeters, and so on) across an entire Teamcenter site. When a user chooses the selector in the unit of measure box of either the **New Item** or **Properties** dialog boxes in the Teamcenter rich client, the user is restricted to entering one of the predefined values.

In Structure Manager, if no specific quantity value is associated with Items, the default quantity is each (one component). Also in Structure Manager, if UOM 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 file where you want to save the data model changes, for example, **options.xml**.
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, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.
 - c. In the **Symbol** box, enter the unit (for example, **in** for inches, **oz** for ounces, and so on).

Note

To add subscripts or superscripts, type them in a text editor that supports subscripts and superscripts, and then paste them into the **Symbol** box.

- 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.

7. Deploy your changes to the server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project and 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.

Introduction to change

A *change* represents an alteration to requirements. The change process begins with the creation of change objects in the Business Modeler IDE, which act as templates of the different change processes to be used at your site.

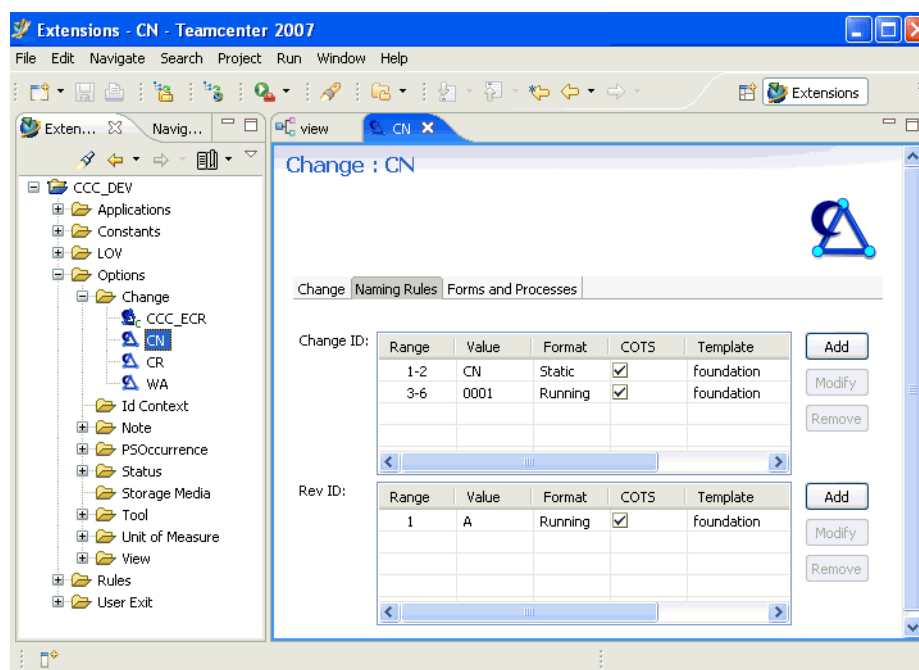
In a typical change process, change requests are created by end users when needed. After change requests are completed and approved, change notices are created. End users use Change Viewer to create and manage the change process.

Three change items are provided with the default Teamcenter foundation installation:

- **CN**
Change notice. Modeled using Configuration Management II (CMII) process requirements.
- **CR**
Change request. Modeled using Configuration Management II (CMII) process requirements.
- **WA**
Work authorization (WA).

When you create a change option in the Business Modeler IDE:

- Give each change a naming rule comprised of a change ID and a revision ID.
- Define the change forms and workflow process to support and manage the change object.



The naming rule for each change breaks down into two parts:

- Change ID

The change ID holds the identifier for the change option, and 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.

- Revision ID

The revision ID holds the identifier for the revision level of the change, and 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 change ID scenarios follow.

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

The following are some invalid change ID scenarios.

Range	Value	Format
2	AB	Static
3-6	0001	Running

Add a change

A *change* represents an alteration to requirements. The change objects you create with the Business Modeler IDE are templates of the different change processes to be used at your site. End users create instances of these change objects, such as change requests, and use them in their workflow processes.

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 file where you want to save the data model changes, for example, **options.xml**.
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. The **Project** box defaults to the already-selected project.
- b. In the **Name** box, type the name you want to assign to the new change item.

When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

- c. In the **Description** box, type a description of the new change.
 - d. Select the **Is Effectivity Shared** check box if revisions of this type of change are each allowed to have a separate effectivity.
 - e. 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. In the following steps, the example ID resolves to **PartECR001/RevA**.
 - 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) Click **Finish**.

The ID is added to the **Change ID** table.

- b. 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**.

- c. 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) Click **Finish**.

The ID is added to the **Rev ID** table.

- d. If desired, click the **Add** button to the right of the **Rev ID** table to put in additional IDs.

For example, for the **PartECR** example, click the **Add** button and enter **4** in the **Range** box, enter **A** in the **Value** box, and enter **Running** in the **Format** box.

If you followed the **PartECR** example, the first assigned ID for a new change would be **PartECR001/RevA**.

- e. Click **Next**.

7. Perform 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.

- 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.

Note

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.
9. Deploy your changes to the server. Right-click the project in the **Extensions** view and choose **Deploy Template**, or select the project and 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. In the left pane of the **Create Change** dialog box, 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.

Activities

In the *Options* section, do the following activities:

- Create a note.
- Create statuses.
- Create units of measure.
- Deploy new options.
- Test new options.

Review questions

1. What is a note?

Select one.

- Associated with an occurrence in a Structure Manager 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.

- Associated with an occurrence in a Structure Manager 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.

- Associated with an occurrence in a Structure Manager 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.

4. A view defines what?

Select one.

- Name of **BOMView** objects
- Name of **Change** objects

- Name of product structure objects
 - Name of **PSView** objects
5. What are valid naming rules formats?
Select all that apply.

- Fixed
- Incremental
- Running
- Static

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Associated with an occurrence in a Structure Manager bill of materials (BOM).
2. Can be set on almost any Teamcenter data to designate a release state.
3. Created so that items and item revisions can be expressed in standardized units.
4. Name of **PSView** objects
5. Running
Static

Summary

The following topics were taught in this lesson:

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

Instructor Note:

Summary instructor notes.

Lesson

13 *Rule extensions*

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*)

Instructor Note:

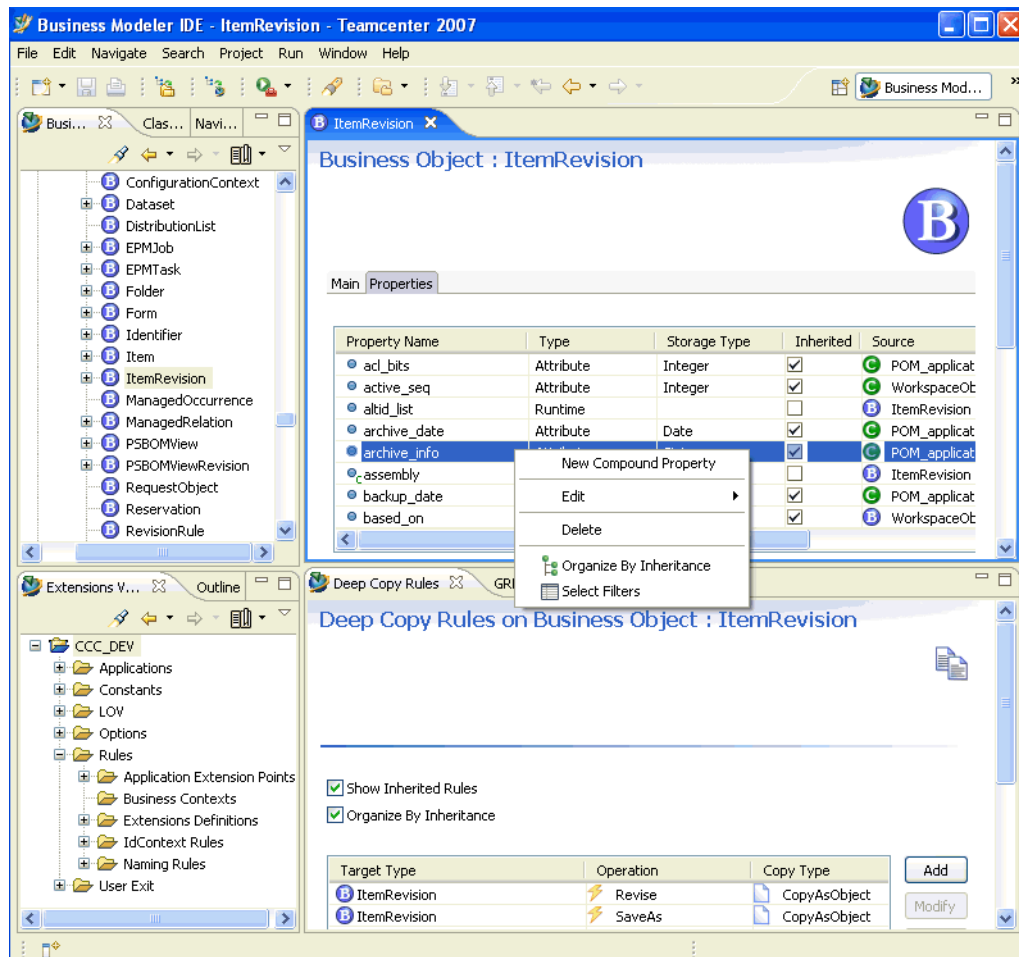
Lesson instructor notes.

Introduction to rules

The **Rules** folder in the **Extensions** is used to work with *rules*, decision points that govern the behavior 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
- Deep copy rules



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 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.

For 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.

Working with business object display rules

By default, all business objects are available for creation by all users. Business object display rules suppress the display of objects in the creation dialog boxes, thus preventing the users in a specific group or role from being able to create objects of those types.

Business object display rules can be applied to the following business objects and their children:

- **Dataset**
- **Folder**
- **Form**
- **Item**

Note

Although users cannot create objects associated with restricted object types, they can view them and perform other operations, such as copying, modifying, or deleting the objects.

Business object display rules are subject to the principles of group hierarchy and inheritance. Rules defined at the site level are inherited by all groups and roles within groups. Rules defined for a group are inherited by all subgroups, but rules applied to a subgroup do not affect the parent groups or other subgroups at the same level in the hierarchy (sibling groups). Rules applied to roles within groups apply to all users with that role, but do not affect other roles in the same group.

In addition to being subject to the principles of group hierarchy and inheritance, business object display rules are also subject to inheritance. Display rules set at the subobject level take precedence over rules set at the business object level.

Note

You can also use the Command Suppression application to suppress the display of menus and commands for certain groups.

Add 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** editor displays the groups and roles 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 kinds of business object.

You can also use the Command Suppression application to suppress the display of menus and commands for certain groups.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. 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 file where you want to save the data model changes, for example, **rules.xml**.

3. 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**, **Identifier**, and **Item**.

In the **Business Objects** view, right-click the relevant business object or one of its children, choose **Open**, and click the **Display Rules** tab.

4. Click the **Add** button to the right of the **Hide Business Object Rules** table. The Hide Display Rule wizard runs.
5. Perform the following steps in the **Display Rule** dialog box:

- a. Click the **Browse** button to the right of the **Organization** box.

The Teamcenter Repository Connection wizard prompts you to log on to a server to look up the available groups and roles in the organization.

The groups and roles from the server appear in the **Select Organization** dialog box. Select the groups and roles that have the business object hidden from them in the create menus in Teamcenter rich client applications.

- b. Select the **Propagate to sub Business Objects** check box if all children of the business object inherit the display rule.
- c. Click the **Browse** button to the right of the **Condition** box to select the condition for which this display rule runs. If you select **isTrue** as the condition, the rule always applies.

- d. Click **Finish**.

The **Hide Business Object Rules** table displays the groups and roles that have the business object hidden from them in the create menus in Teamcenter rich client applications.

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 **rules.xml** file, the changes are saved in that file.
7. After you create a rule, you can deploy your changes to the server. Right-click the project in the **Business Objects** view and choose **Deploy Template**, or select the project and click the **Deploy Template** button on the main toolbar.
8. After deployment, test your new display rule in the Teamcenter rich client.

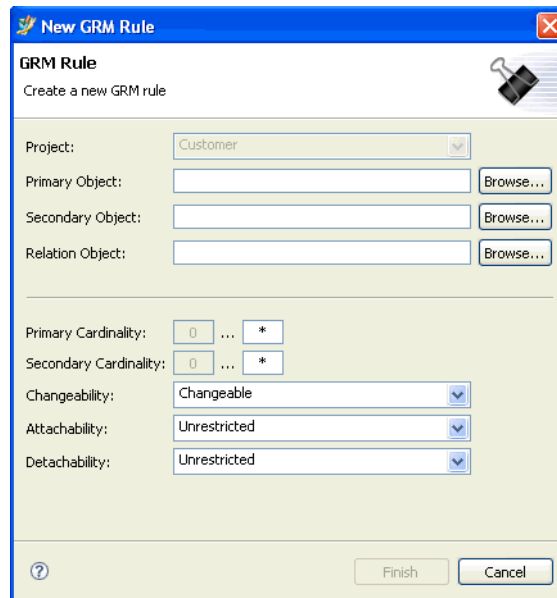
For example, if you created a display rule on the **Item** business object, log on to the My Teamcenter application as a user in a group that has rights to create items and choose **File→New→Item**. You should be able to create new **Item** business objects. Then log on as a user in a group that does not have rights to create **Item** business objects. You should not be able to create **Item** business objects.

Working with GRM rules

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. Available relationships are children of the **IMANRelation** business object.

About GRM rules

You can use Generic Relationship Management (GRM) rules to limit what objects can be pasted to other objects. For example, if you do not want a certain type of object to have a specification relation to another type, you can set the cardinality to **0** to deny pasting of one type of object to another with the specification relation.



When you create the GRM rule, the following constraints must be defined:

- **Cardinality**

Determines the number of allowed occurrences of the primary object in relation to the secondary object, and 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.

You can see the relations between business objects in the UML editor by right-clicking and choosing **Show→Relations**.

Note

The GRM rule applies for all children of the primary and secondary objects. For 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. 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 file where you want to save the data model changes, for example, **rules.xml**.
3. Right-click in a view (such as the **Business Objects** view) and choose **Open GRM Rules Editor**.

The **GRM Rules** editor displays the active GRM rules in the project.

4. 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**, **Edit**, 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. Click the **Browse** button to the right of the **Relation Object** box to choose the relationship between the primary and secondary business objects. 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** editor.

- 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 server. Right-click the project in the **Business Objects** view and choose **Deploy Template**, or select the project and 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 allow only one **UGPART** object to be related.

Activities

In the *Rules* section, do the following activities:

- Create a GRM rule for CCC_ERPDataForm.
- Create a GRM rule with CCC_justification_rel.
- Verify the GRM rules for CCC_ERPDataForm.
- Verify the GRM rules for CCC_MSWord.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. False

The user must select the **Propagate to sub Business Objects** box if all children of the business object inherit the display rule.

2. Primary Cardinality

Working with naming rules

Naming rules define the data entry format for a business object property. 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. After you create a naming rule, you must attach it to the business object property.

For example, you can create a rule to define how new **Item** business objects are named and attach it to the **item_id** property on that business object. If you want all your items to be named **AcmeCoPart_** followed by an incremental number, you create a naming rule to automatically assign that name when items are created.

Naming rules can be used to name items, item revisions, datasets, forms, projects, and work contexts. You can also attach the naming rule to a property on all business objects that use that property. For example, if you attach the rule to the **item_id** property itself, then all business objects use the rule on that **item_id** property.

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. 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.

Example

You can create a naming rule for an item ID that starts with **CCC** plus a six digit number so that it generates numbers from **CCC000001** to **CCC999999**.

First enter a pattern of **"CCC"** . Then enter a second pattern of **NNNNNN** (for numbers) with an initial value of **000001** and a maximum value of **999999**. Then attach it to the **item_id** property of the item business object you want.

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 rule. 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, **rules.xml**.
3. Expand the project and the **Rules→Naming Rules** folders.
4. Right-click the **Naming Rules** folder and choose **New Naming Rules**.
The New Naming Rules wizard runs.

5. Enter the following information in the **Naming Rule** 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 naming rule.

The name cannot contain spaces. When you name a new data model object, Siemens PLM Software recommends that you add a prefix to the name to designate the object as belonging to your organization, such as an acronym.

- c. Click the **Add** button to add a naming rule pattern in the **Patterns** pane.

The Add Naming Rule Pattern wizard runs.

- d. Perform the following steps in the **Pattern** dialog box:

- 1) In the **Pattern** box, type a naming pattern.

For example, if you want a three-digit number pattern running 001 to 999, type **nnn**. If you want a two-character alphabetic pattern running from aa to zz, type **aa**, or if you want AA to ZZ, type **AA**. If you want a mixture of letters and numbers, such as A001 to Z999, you can mix the two, for example, **Annn**.

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.

- 2) Click the **Insert LOV** buttons to add an LOV as part of the pattern.
- 3) Click the **Insert Rule** button to use an existing naming rule for the pattern.
- 4) In the **Description** box, type a brief description of the naming rule.
- 5) Select the **Generate Counters** check box if you want to add a counter to the naming rule.
- 6) If you selected the **Generate Counters** check box, type characters in the **Initial Value** box and the **Maximum Value** box that match the pattern.

For example, if you entered **nnn** for the pattern, type a three-digit number in the **Initial Value** box and the **Maximum Value** box, such as **100** and **899**. Or if you entered a pattern of **Annn**, you could type **A001** and **Z999**.

- 7) 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.

- e. Change the patterns as desired by using the **Add**, **Edit**, **Remove**, **Copy**, **Move Up** and **Move Down** buttons.
- f. Click **Finish**.

The naming rule is added to the **Naming Rules** folder.

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

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 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 **Item** business objects items are named, attach the naming rule to the **item_id** property of the **Item** business object.

Note

The following procedure describes how to open the naming rule and add property attachments in the **Naming Rule Attachments** table. You can also attach a naming rule to a property by opening the business object, selecting a property in the **Properties** tab, and clicking the **Add** button to the right of the **Naming Rule Attaches** table.

1. Access the **Business Modeler IDE** perspective by choosing **Window→Open Perspective→Other→Business Modeler IDE**.
2. In the **Business Objects** view, 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, **rules.xml**.
3. In the **Extensions** view, open the **Rules→Naming Rules** folders, right-click the naming rule you want to assign, and choose **Open**.

The Naming Rule editor is displayed.

4. Click the **Attach** button to the right of the **Naming Rule Attachments** table.

The Naming Rule Attachment wizard runs.

5. Perform the following steps in the **Property Selection for Naming Rule** dialog box:
 - a. Click the **Browse** button to the right of the **Property** box to choose the business object and property to which you want to attach the naming rule.
 - 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 the **Browse** button to the right of the **Condition** box to select the condition that determines when the naming rule is applied. If you select **isTrue** as the condition, the value always applies.

d. Select **Override** if you want to allow this LOV to be overridden by other templates.

e. Click **Finish**.

The naming rule is attached to the property on the business object. The business objects and properties display in the **Naming Rule Attachments** table.

To detach the naming rule from any particular property, select the business object in the table and click the **Detach** button.

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 **rules.xml** file, the changes are saved in that file.

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

8. 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 on a custom item business object, in the My Teamcenter application, try creating some items to see if the rule is applied. For example, choose **File→New→Item** and choose the custom item as the type. When you click the **Assign** button, your new naming rule pattern appears in the **ItemID / Rev - Name** boxes.

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 would allocate 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
```

Working with ID context rules

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.

Instructor Note:

Only provide a high level overview on this topic. ID context rules are considered an advanced topic.

Activities

In the *Rules* section, do the following activities:

- Create naming rules for an item.
- Create naming rules for an item revision.
- Verify the Naming Rules.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. X or x
2. False

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).

Working with deep copy rules

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.

Deep copy example



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 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. 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 file where you want to save the data model changes, for example, **rules.xml**.

3. 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.
4. Right-click the **ItemRevision** business object or one of its children, choose **Open**, and click the **Deep Copy Rules** tab.

The **Deep Copy Rules** editor displays the active rules on the item revision business object.

5. 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**, **Edit**, 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. Click the arrow in the **Operation Type** box to select **Save As** or **Revise** as the operation to invoke the copy action.
 - b. The **TargetType** displays the business object that the deep copy rule is applied to, for example, **ItemRevision** or one of its children.

- c. Click the **Browse** button to the right of the **Relation Type** box to select the relationship business object to use for the relationship between the copied object and its item revision.

Available relationships are children of the **IMANRelation** business object. Objects with a relationship that matches the selected relationship are only copied from the source item revision to the destination item revision. To match all relationships, select **Match All**.

- 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.

Select the **MatchAll** value if you want all objects to be copied forward no matter what type of business object they are.

- e. Click the **Browse** button to the right of the **Condition** box to select the condition for which this deep copy rule runs. If you select **isTrue** as the condition, the deep copy rule always applies.

Any condition other than **isTrue** must use one of the following conventions:

- Uses the **UserSession** business object as its only input parameter
- Uses three input business object parameters in this order:
DeepCopyRule, **ItemRevision** (or one of its children),
POM_object (or one of its children)
- Uses four input business object parameters in this order:
DeepCopyRule, **ItemRevision** (or one of its children),
POM_object (or one of its children), **UserSession**

- f. In the **Action** box, choose the kind of copying to be allowed for the business object:

- **CopyAsObject**

Creates a new object of the same type as the related object and relates to the new 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**

Creates a new relation between the new revision and the related object. Therefore, modifications performed on the copied object are propagated to the source object.

- **NoCopy**

Does nothing. Objects of the selected type and relation are not copied forward to the new object during the save as or revise operation.

- **RelateToLatest**

Finds the latest revision of a related object and creates a relation to the new revision. You can only use this with the **Revise** operation type.

This action replaces the **AutoCopyRel** business object constant, which is now deprecated.

- **ReviseAndRelateToLatest**

Finds the latest revision of a related object, revises it, and creates the relation to the new revision. You can only use this with the **Revise** operation type.

This action replaces the **AutoRevise** business object constant, which is now deprecated.

- g. Select the **Is Target Primary** check box to indicate that the business object shown in the **Target Type** box is the primary object of the relationship specified in the **Relation Type** box. When the item revision is revised or saved, the secondary objects are carried forward and related via the relation in the **Relation Type** box.

Clear the **Is Target Primary** check box to indicate that the business object shown in the **Target Type** box is the secondary object of the relationship specified in the **Relation Type** box. When the item revision is revised or saved, the primary objects are carried forward and related via the relation in the **Relation Type** box.

- h. Select the **Copy Properties on Relation** check box to specify that persistent properties on relation objects need to be carried forward when the primary objects participating in relations are revised or saved as new objects. If it is unselected, only the mandatory properties are carried forward.
- i. Select the **Required** check box if you want to prevent users of the Teamcenter rich client from overriding the rule at run time.
- j. Select the **Secured** check box to prevent the deep copy rule from being modified or overridden by another template.
- k. Click **Finish**.

The rule is created and appears in the table in the **Deep Copy Rules** editor.

8. To save the changes to the data model, choose **File→Save Data Model**, or click the **Save Data Model** button in 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 server. Right-click the project in the **Business Objects** view and choose **Deploy Template**, or select the project and click the **Deploy Template** button in 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.

Working with extensions

Extensions 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.

To see extensions defined for business objects, right-click a business objects and properties, choose **Open**, and click the **Operations** tab in the resulting editor. The available C++ style signature operations for the business object are found in the **Operations** folder, whereas operations from previous versions of Teamcenter are in the **Legacy Operations** folder at the bottom of the **Operations** folder. Available operations for the properties are found in the **Property Operations** folder. To see the extension points defined for an operation, select the operation and click the **Extensions Attachments** link on the lower right side of the tab.

Note

All the operations in the **Legacy Operations** folder will be migrated in the future and will be available under the **Operations** folder. For example, the **ITEM_create** operation under the **Legacy Operations** folder will be migrated to the **create(datatypes)** operation under **Operations** folder. Siemens PLM Software recommends that from now on you attach pre/post-actions on the **create(datatypes)** operations instead of on the **ITEM_create** operation. All the existing pre/post-actions already attached on the **ITEM_create** operation will continue to work.

You can also use predefined extensions to perform actions. The predefined extensions rules can be viewed under the **Rules→Extensions** folder.

To create your own extension:

1. Write code for the extension.
2. Create the extension definition.
3. Assign the extension to a business object or property.

Predefined extensions

Following are the predefined extensions rules that can be viewed under the **Rules→Extensions** 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.

- **CreateUpdateBBoxAndTSO**

- **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 option, 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.

- **restrictDataCreationToProgram**

- **setIdentifierProperties**

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.

- **setOrgOnCreation**

- **validateFile**

Validates proper file format after a dataset file import.

- **validateImport**

- **validateOrgOnCreation**

- **validateProjectForImport**

- **validateStdPrtRevoke**

Add a predefined extension to a business object

You can use predefined (internal) extensions 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.

1. In the **Extensions** view, expand the project and the **Rules→Extensions** 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, choose **Open**, and click the **Operations** tab in the resulting editor. The available operations for the business object are found in the **Operations** folder.
4. Select the operation in the **Operations** folder and click the **Extensions Attachments** link on the lower right side of the editor.
5. 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.

The Add Extension Rule wizard runs.

6. In the **Extension** dialog box, click the **Browse** button to the right of the **Extension** box and choose the extension.
7. If arguments are required on the operation, the **Add** button to the right of the **Arguments** box is available. 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.

8. Click **Finish**.

The extension is added to the table under the extension point.

9. Save the data model by choosing **File**→**Save Data Model**, and deploy the data model by right-clicking the project in a view and choosing **Deploy Template**.

Verify that the extensions 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.

Activities

In the *Rules* section, do the following activities:

- Add deep copy rules.
- Modify display rules for a business object.
- Verify the latest deployment.

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**

Instructor Note:

Activity instructor notes.

Answers to review questions

1. **MyPart Revision (2)**
IMAN_specification (relation)
 MyPartSpec002

2. Post-Action

Pre-Action

Pre-Condition

Summary

The following topics were taught in this lesson:

- Rules govern the behaviors of business objects.
- Business object display rules limit the objects that can be created by particular groups or roles.
- The GRM rule constrains copy and paste capabilities between objects.
- Naming rules define how objects are named and consists of rule patterns and a counter.
- ID context rules assign IDs to alias and alternate items depending on their context.
- Compound property rules allow a property on a business object to be displayed as a property of an object.
- Deep copy rules define whether objects belonging to an item revision are copied or referenced when a revision is saved or revised.
- Extension rules allow you to write custom code and attach the rules to predefined hook points in Teamcenter.

Instructor Note:

Summary instructor notes.

Lesson

14 *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:

- Install templates to the Business Modeler IDE and the server.
- 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 *Basic concepts for using the Business Modeler IDE*)

Instructor Note:

Lesson instructor notes.

Introduction to managing templates

Managing templates in Teamcenter includes installing, deploying and packaging data model files to both test and production environments.

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.

Using an SCM system

The Business Modeler IDE manages the master XML files that contain your extension definitions. Although these XML definitions are converted to data model objects in the database during a deploy, these files are not stored in the database and should be cared for like source code. Back up these files regularly.

If only one person will be using the Business Modeler IDE to manage the data model, Siemens PLM Software strongly suggests you use a source control management (SCM) system plug-in with Eclipse to manage the project and source files.

If more than one person will be working on the data model of a single Business Modeler IDE template, *you are required* to use an SCM to integrate all developer's Business Modeler IDE clients with the central SCM repository. The SCM plug-in can be used to maintain version control of the source files, and to ensure protection of the data in the source files. In addition, with an SCM plug-in, multiple developers can work on the same project concurrently.

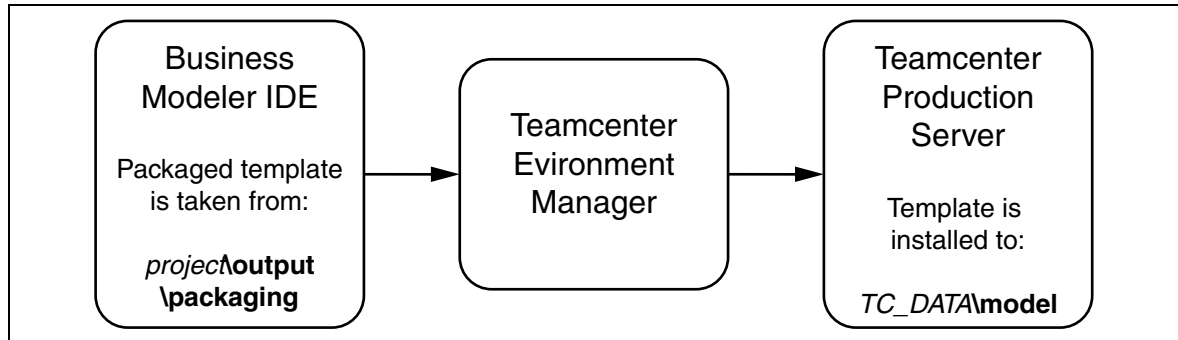
You can use any SCM system that provides plug-ins to Eclipse, such as ClearCase, CVS, Subversion, or Perforce. Each developer connects to the repository to share and synchronize definitions.

As a starting point to find SCM plug-ins, go the following URL:

<http://www.eclipse.org>

Install a template to a production server

To install your template to a Teamcenter production server, package it using the Business Modeler IDE and install it using the Teamcenter Environment Manager (see the figure below).



Install a packaged template to a production server

1. In the Business Modeler IDE choose **File**→**New**→**Other**→**Business Modeler IDE**→**Package Template Extensions**.

The packaged template is saved to:

`project\output\packaging`

2. Copy the template files to a directory that is accessible by the production server.
3. In Teamcenter Environment Manager on the production server, proceed to the **Feature Maintenance** panel.
4. Select **Add/Remove Features** and click **Next**.
5. In the **Select Features** panel, click **Browse** to locate the template.
The template is added to the **Select Features** panel.
6. In the **Select Features** panel, choose the new feature.

Package extensions into a template

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

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 in your workspace. To find the *workspace* location, choose **File**→**Switch Workspace**.

For example, on a Windows system, they are saved by default to:

```
install-location\bmide\workspace\
version\project\output\packaging
```

On UNIX, users need to have permissions to the workspace directory.

The following template files are placed in the *project/output/packaging* directory:

- **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***_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**).

- *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.

If you have generated C++ classes or services artifacts, the following files are added:

- *template-name***_rtserver.zip**

This file contains C++ run-time libraries (*template-name***_rtserver.xml**).

- *template-name***_soa_client_kit.zip**

This file contains new services.

- **Web_tier**

This folder contains new services Web-tier deployment files for .NET and J2EE servers.

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.

- After the Business Modeler IDE first loads 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 load and validate your model using the **Reload Data Model** menu command (available in most views within the Business Modeler IDE).

This 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 release. Whenever any of these conditions occur, you should right-click in a Business Modeler IDE view and choose the **Reload Data Model** command to reload your model and have it validated. If no errors occur, then you can be sure that your template can install correctly into a server.

Install a template using TEM

After you package extensions, install the resulting template to a production environment using Teamcenter Environment Manager (TEM). You can also use this procedure to install a third-party template.

1. Copy the template files from the **packaging** directory on your Business Modeler IDE client to a directory that is accessible by the server.

By default, packaged template files are located in the Business Modeler IDE workspace directory in the folder under the project.

On UNIX, users must have permissions to the workspace directory.

2. Start Teamcenter Environment Manager (TEM).
3. In the **Maintenance** panel, choose **Configuration Manager** and click **Next**.
4. In the **Configuration Maintenance** panel, choose **Perform maintenance on an existing configuration** and click **Next**.
5. In the **Configuration Selection** pane, select the configuration from which the corporate server was installed. Click **Next**.
6. In the **Feature Maintenance** panel, under the **Teamcenter** section, select **Add/Remove Features**. Click **Next**.

Note

If you already installed a template to the database and want to update the template, under the **Teamcenter Foundation** section, select **Update the database**. This option should not be used to install a new template but only to update an already installed template.

Use the **Add/Update templates for working within the Business Modeler IDE client** option under **Business Modeler Templates** only if you want to add a dependent template to your Business Modeler IDE.

7. In the **Select Features** panel, click the **Browse** button on the lower right side of the panel.
8. Browse to the directory where you have copied the template files. In the **Files of type** box, ensure that **Feature Files** is selected so that you see only the installable template (feature) file. Select your template's feature file (**feature_template-name.xml**) and click the **Select** button.

The template appears as a new feature under **Teamcenter Corporate Server** on the **Select Features** panel.

9. Select the new template in the **Select Features** panel under **Teamcenter Corporate Server**. Click **Next**.
10. In the **Teamcenter Administrative User** panel, enter your username and password to log on to the server. Click **Next**.
11. The **Database Template Summary** panel displays the list of templates that are installed as part of your template install. Click **Next**.
12. In the **Confirm Selections** panel, click **Next**. The new template is installed.

If the installation fails because of invalid data model, perform the following steps:

- a. Fix the incorrect data model and repackage the template.
 - b. Locate the *template-name_template.zip* in your project's **packaging** directory and unzip it to a temporary location. Copy the following files to the server in the *TC_ROOT/install/template-name* folder:
 - template-name_template.xml*
 - template-name_dependency.xml*
 - template-name_tcbaseline.xml* (if the file exists)
 - c. Launch Teamcenter Environment Manager in the maintenance mode and continue with recovery.
13. To verify the installation of the new template, confirm that the *TC_DATA* directory on the Teamcenter server contains the new template files.

Also log on to the server and confirm that you can create instances of your new data model.

Activities

In the *Data model files* section, do the following activities:

- Package extension into a solution template
- Import a project.

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.

Instructor Note:

Activity instructor notes.

Answers to review questions

1. True.
2. Load and validate your model.

The remaining answers are why the **Reload Data Model** validation tool is important.

Summary

The following topics were taught in this lesson:

- Installing templates from the TEM.
- Package extensions to the data model as a solution template and distribute the template for installation to a production environment.
- The template project directory and all of its contents should be managed in a source control management system (SCM).
- The Business Modeler IDE performs strict validations on the extensions in your template.
- You can import either an entire project template or just a model file.

Instructor Note:

Summary instructor notes.

Lesson

15 Organization

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.
- Search the organization.
- Set up accounts manually.
- Import and export the organization.

Help topics

Additional information for this lesson can be found in:

- [Organization Guide](#)
- [Security Administration Guide](#)
- [Utilities Reference](#) (see *Volume and database management utilities*)
- [Utilities Reference](#) (see *Data sharing utilities*)

Instructor Note:

Start out by giving a high level overview of the organization structure, then dive into the details. The beginning is concept information only. Later, you will show them how to build the organization in Teamcenter.

Introduction to Organization

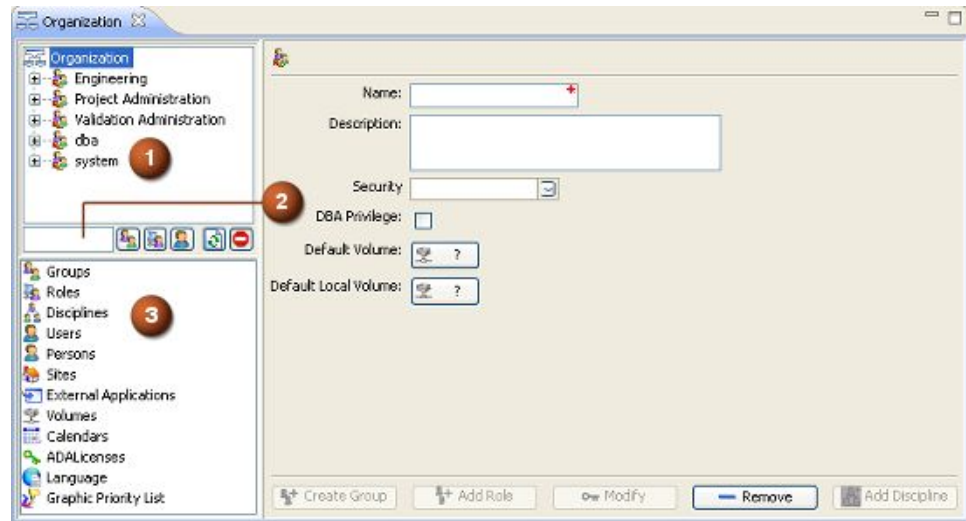
The Organization application enables you to create and maintain your company's organization within Teamcenter. This organization is a way of organizing user accounts and their respective permissions and user groups. User accounts help you:




- Track changes to objects.
- Control access and privileges.
- Manage default object ownership.

Use this administrative application to perform tasks like:

- Setting up your organization for the first time
- Required organizational tasks:
 - Defining sites
 - [Defining volumes](#)
 - [Defining roles](#)
 - [Defining groups](#)
 - [Defining persons](#)
 - [Defining users](#)
- Optional organizational tasks:
 - [Defining administrative privileges](#)
 - Defining disciplines
 - Defining calendars
 - [Defining authorized data access \(ADA\) licenses](#)
 - Defining external applications

Organization interface



- 1 **Organization tree** 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. Selecting a node starts **Organization** wizards used to create groups, subgroups, roles, disciplines, and users.
- 2 **Find box** Use the find box to filter the **Organization** tree to find groups , roles , and users  within the organization. You can also use the find box to reload the **Organization** tree and to locate inactive group members.
- 3 **Organization List tree** The **Organization List** tree enables you to view and manage the components of your organization by listing groups, roles, disciplines, users, and persons. You can also use the **Organization List** tree to manage sites, external applications, volumes, calendars, and ADA licenses.

Organization hierarchy

The Organization application supports two ways to create and manage your company's organization within Teamcenter:

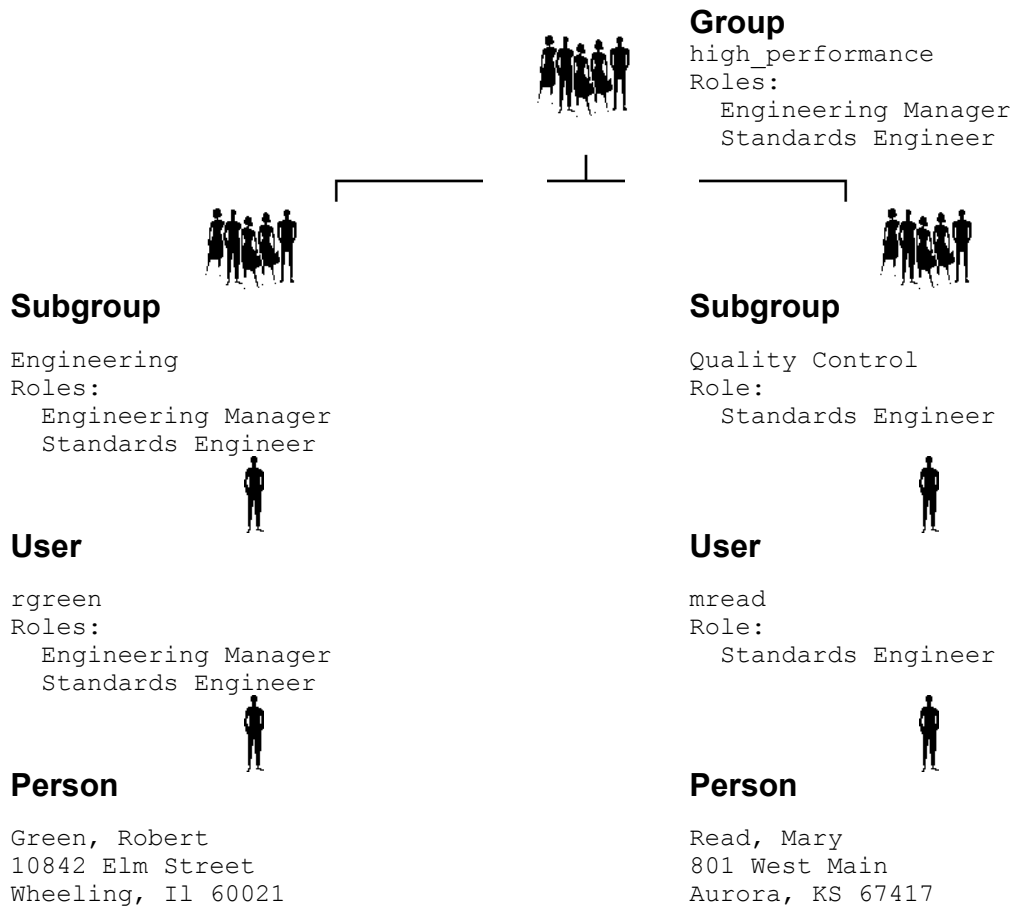
- From the bottom up using the **Organization List** tree.

You also use the **Organization List** tree to manage sites, volumes, calendars, and authorized data access (ADA) licenses.

- From the top down using the **Organization** tree.

Basic concepts for using Organization

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



- **Group**

A *group* is a grouping of users who share data.

You can configure access to data owned by the group by:

- Using the **Security** setting, which allows or restricts access to data.
- Setting authorized data access (ADA) and International traffic in Arms Regulations (ITAR), which allows or restricts access to data based on clearance levels and data classification.

- **Role**

A *role* represents specific skills and/or responsibilities. The same roles are typically found in many groups. The system grants data access based on group and role.

- **User**

A *user* can belong to multiple groups and must be assigned to a default group. Each user in the group is assigned a role.

In addition, you can associate the following with users by:

- Creating a calendar, which allows you to set days off, holidays, and hours in a day for individual resources.
- Setting ADA and ITAR attributes to allow or restrict data access.
- Set the licensing level to either author or consumer, which determines if the user can create and modify data or only view data.

- **Person**

A *person* is an object containing real-world information attached to a Teamcenter user, such as name, address, and telephone number.

Typical organization administration tasks

Typical administrative tasks related to maintaining your organization include:

- Creating and maintaining persons, users, groups, roles, and disciplines.
- Activating user accounts and maintain passwords.
- Creating the organizational hierarchy by assigning persons to users, users to roles, and roles to groups and disciplines.
- Assigning administrative privileges to users and groups.
- Defining sites.
- Creating and maintaining the master calendar that enables users to build project schedules using the Schedule Manager application.
- Creating and maintaining volumes.

Defining persons

Persons are individuals who work at your site. A person has properties such as **Person Name**, **Street Address**, and **Employee Number**.

- Consider creating all persons at your site using the following naming convention:

last-name, first-name, middle-initial

- Person definitions that are referenced by a **User** object cannot be deleted.

You must define a person for each Teamcenter user. As a user with **DBA** or group administrator privileges, you use the Organization application to:

- Create person definitions.
- Modify person definitions.
- Delete person definitions.

Example



```
Green, Robert, M
10824 Elm Street
Wheeling, IL 60021
Organization: Company BB
Employee Number: 150
e-mail: robertg@bblades.com
```

Defining users

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 operating system user account.

A user is assigned to a default group and takes on a role in the group. As a user with **DBA** privileges, you use the Organization application to:

- Create, modify, and delete user accounts.
- Maintain user password restrictions.
- Deactivate or activate user accounts.
- Assign group administrator privileges.
- Assign intellectual property and government clearances to data stored in Teamcenter for user accounts.

Example



Person

Green, Robert, M
10824 Elm Street
Wheeling, Il 60021



User

rgreen
Default group:
 high_performance
Role:
 Standards Engineer

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
```

Teamcenter Security Services

In addition to application authentication within Teamcenter, Security Services allows you to move from one Teamcenter product solution, such as Enterprise Knowledge Management, to another solution, such as Lifecycle Visualization, without encountering multiple authentication challenges.

Security Services includes the following features:

- Single sign-on to Teamcenter products, for both the rich client and the thin client.
- Common authentication through LDAP v3-compliant directory servers, such as Microsoft Active Directory and the Sun iPlanet Directory Server, which can be customized to work with other authentication services.
- Interoperation with commercial single sign-on products.
- Lightweight directory access protocol (LDAP) referrals, which allow you to be distributed across multiple LDAP servers.

Organizations maintained on a corporate LDAP directory server can be synchronized with Teamcenter using the **ldapsync** utility.

The synchronized organization data is considered to be *externally managed*, and when maintained using the Organization application, is subject to restrictions that do not apply to internally managed organizations.

Example

Externally managed passwords are never synchronized with Teamcenter and user status should not be mapped to an LDAP attribute.

Security Services are installed and configured separately from Teamcenter 8 MP1.

Defining roles

A *role* is an object that models the type of work a user is expected to perform in a group.

- A role can be assigned to multiple groups.
- Roles add another layer of data access control.
- Roles are created along functional lines.

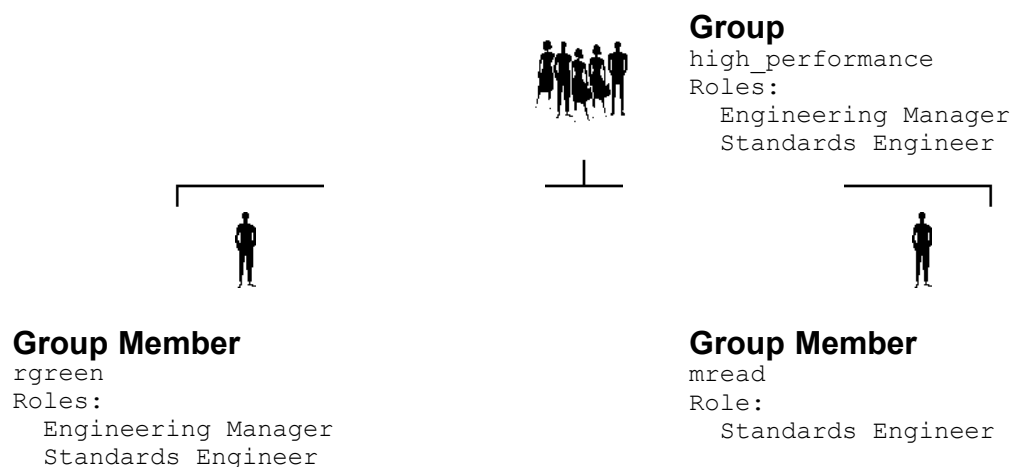
Tip

Use real-world descriptions, skills, and/or responsibilities.

Roles refine the group definitions of your organization structure. As a user with **DBA** privileges, you use the Organization application to:

- Create, modify, and delete role definitions.
- Add existing roles to the **Organization** tree.
- Add new roles to the **Organization** tree.

Example



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**.

Defining groups

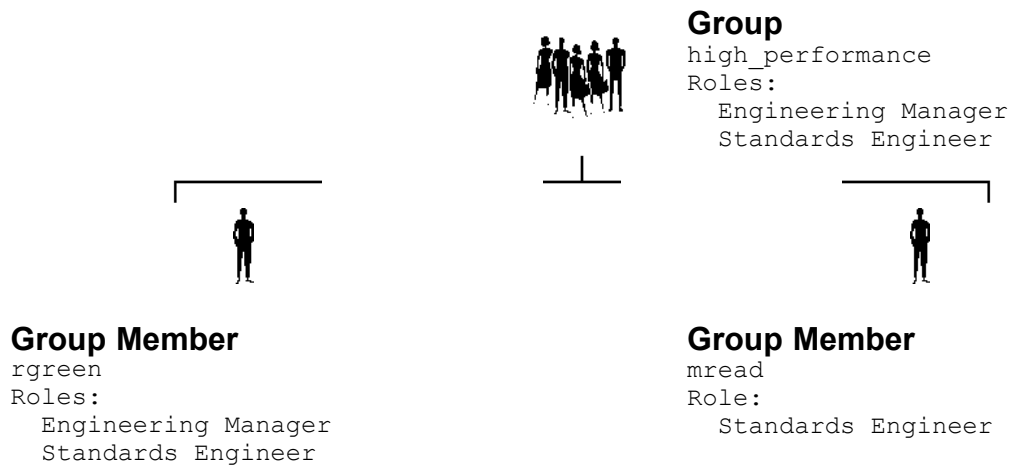
Groups contain members (users) who take on a role or multiple roles in the group. Groups represent data ownership and therefore control data access.

- Groups are defined along project lines, not functional lines, but can define third-party organizations such as suppliers.
- A group member can be a member of many groups. For example, Robert Green can belong to the **high_performance** and **standards** groups.

Groups make up the core of your organization structure. As a user with **DBA** privileges, you use the Organization application to:

- Create, modify, and delete groups.
- Manage subgroups within the **Organization** tree.
- Assign default volumes to a group.
- Assign authorized data access privileges to a group.

Example



Defining subgroups

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.

- Subgroups are an excellent way to organize your users.
- Subgroups inherit access permissions, volumes, and preferences from their parent.

Instructor Note:

Volumes are a location where files are stored. Volumes will be discussed in more detail later.

Example



Group hierarchies

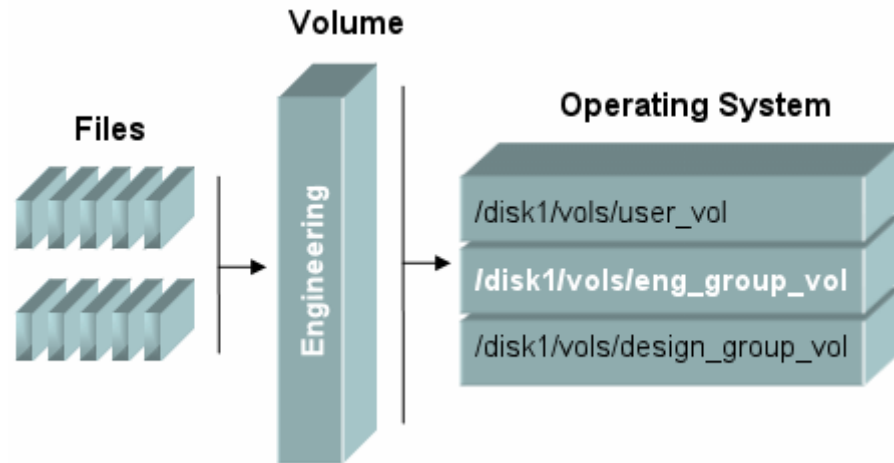
Groups are organized into one or more trees or hierarchies. Each group has exactly one parent group (unless it is at the top or root of the hierarchy, when it has no parent) and can have one or more subgroups.

The following list indicates the functional areas in Teamcenter that use group hierarchies.

Functional area	Group hierarchy use
Access Manager	A group can inherit access permissions 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.
Workflow	Signoffs can be assigned to members of a group's subgroups and members of the named group.

Defining volumes

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.



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

Volumes are assigned to groups and users and define file locations for your organization structure. As a user with **DBA** privileges, you use the Organization application to:

- Create and delete volumes.
- Modify volume location and properties.
- Control volume access.

Defining authorized data access licenses

Authorized data access (ADA) is a security solution that complements other Teamcenter security features, such as Access Manager rules and access control lists (ACLs). Authorized data access controls sensitive data through the use of data classification, user clearance, and authorizing documents. When users or groups attempt to access classified data in Teamcenter, their clearance level is evaluated against the classification of the object based on Access Manager rules. If the user or group clearance level is equal to or greater than the classification on the object, access is granted.

There are three types of licenses for authorized data access:

- IP license

Grants discretionary access to data for a specific user for a specific period of time. You can configure the rule tree to check for a valid IP license associated with an object and user. If found, other access checks are bypassed.

- Exclude license

A mechanism for denying users access to data for a specific period of time. You can configure the rule tree to check for a valid execution license associated with an object and user. If found, other access checks are bypassed.

- ITAR (International Traffic in Arms) license

Grants access for U.S. nationals outside the United States or foreign nationals named by an effective Technical Assistance Agreement to access protected product data, which in the United States could contain technical information that is restricted by ITAR.

You can configure logging and menu suppression (blocking) to be applied when classified data is loaded in Teamcenter Integration for NX.

As a user with an ADA administrator role (**IP Admin** or **ITAR Admin**), you use the Organization application to create and maintain licenses in Organization. Once created, access is either granted or denied to users and groups by associating the license directly with the data object.

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. A volume is a location where files are stored.

- True
- False

3. A subgroup must have a unique group name.

- True
- False

Instructor Note:

Review the workbook scenario.

Ask them to fill out the worksheet for the organization according to the scenario.

Answers to review questions

1. Role, group, person, user

A role must be created before the group.

A group must be created before the user.

A person must be created before the user.

2. True

3. False

A subgroup must have a unique group name but only within the parent group.

Defining administrative privileges

Administrative privileges are required to manage Teamcenter administrative data like organization, access rules, and workflows. Administration accounts have powerful access to data and must be controlled and managed with caution.

When managing administration accounts, consider these key points:

- The **infodba** account has all the Teamcenter system-level privileges.
- The **Bypass** administrative setting supersedes other privileges.
- 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. Group administrators can assign other group administrators in their group.
 - The **DBA** role for a user in a non-**dba** group has no additional privileges over any other group member.

infodba account

The **infodba** account is the only account in Teamcenter after the installation.



Person
infodba

The **infodba** account consists of:

A person object named **infodba**

A user object named **infodba**

A group object named **dba**

A role object named **DBA** and a group member object



User
infodba



Group Member
infodba (infodba)
Group= **dba**
Role= **DBA**

The **infodba** account has all the Teamcenter system-level privileges, including a bypass switch to override access protections. The **infodba** account has special access permissions applied to data they create.

System administration accounts

Administrator privileges include the following settings:

Setting	Privileges
Member of the dba group or another dba group	All system administration privileges. Users in the dba group have a Bypass toggle that can be turned on with the User Setting dialog box (Edit® User Setting® Administrative).
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. Member of a group with DBA privileges.
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.

Creating your virtual organization

As a Teamcenter administrator, you use the Organization application to create and maintain your company's virtual organization within Teamcenter.

The basic process of creating a virtual organization includes the following stages:

1. Establish Teamcenter sites.
2. Define the structure of your organization.

Note

Roles, volumes, and persons are independent definitions. Any of these can be created without consideration of the other definitions.

- a. Create roles.

Warning

Do not delete the roles provided with Teamcenter. They are required for the product to function properly.

- b. Create volumes.
- c. Enable File Management System (FMS) control of new volumes.
- d. Create groups and subgroups.

Warning

Do not delete the **system** group provided with Teamcenter. It is required for the product to function properly.

Note

Groups and users are dependent definitions. To create group and user definitions, other definitions must exist:

- Group definitions require that a role is defined and suggest a volume be defined.
- User account definitions require a person definition and a group definition.

- e. Add roles to groups.

3. Create users and persons.
 - a. Create person definitions.
 - b. Create user accounts.

Note

The **make_user** utility allows you to create your organization from a command line or script.

Creating the organization structure

Create an organization structure by creating groups, subgroups, roles, and volumes.

1. The system administrator is responsible for providing a list of volumes.
2. Create new roles to be used in groups.
3. Create groups and subgroups. Two groups are provided: **dba** and **system**.

Creating a volume

The screenshot shows a 'dba_vol' dialog box with the following fields and controls:

- Volume Name:** dba_vol
- Node Name:** dtclxxx
- Machine Type:** ☐ Unix ☒ Windows
- UNIX Path Name:** (empty)
- Windows Path Name:** D:\es1_vols\dba_vol
- FSC Path Name:** (empty)
- ID Type:** ☒ FSC ☐ Filestore Group ☐ Load Balancer
- ID:** (empty)
- FMS Configuration:** Reload, Report, Display (buttons)
- Statistics:** Size: 20473 Mb, Used: 6443 Mb, % Full: 31%
- Mirrored:** (checkbox)
- Accessors:** dba, Engineering (list box)
- Buttons:** Revoke, Grant, Create, Modify, Delete, Clear

Note

Because of limitations and restrictions in NTFS file systems, Siemens PLM Software recommends creating the volume on the machine where the disk physically resides. It is important to choose a location that is constantly accessible to all users.

1. Select the top-level **Volumes** node from the **Organization List** tree. Teamcenter displays the **Volumes** pane.
2. Type a unique descriptive character string in the **Volume Name** box.
3. Type the name of the network node that physically contains the new volume in the **Node Name** box.
4. Select the machine type on which the volume will reside: **Unix** or **Windows**.

5. Depending on the machine type, type the full UNIX or Windows path of the new volume in either the **UNIX Path Name** box or the **Windows Path Name** box.
6. Type the FMS server cache (FSC) path name in the **FSC Path Name** box. This is the path to the volume in the event that the mount point from the FSC differs from that of Teamcenter File Services (TCFS). In most installations, this box is left blank.
7. Select the ID type (**FSC, Filestore Group, Load Balancer**) to indicate the element in the FMS master configuration to which the new volume element is to be added and type the ID in the **ID** box.

The value you enter into the **ID** box varies depending upon where the new volume is to be added in the FMS master configuration. That location in the configuration is determined by the ID type selection.

Following are examples of each ID type:

- **FSC:**

```
<fscGroup id="fscGroup1"
  <fsc id="fsc1" address="http://csun17.ugs.com:4444">
    <volume id="vol1" root="/data/vol1"/>
  </fsc>
  <fsc id="fsc2" address="http://csun18.ugs.com:4444">
    <volume id="vol2" root="/data/vol2"/>
  </fsc>

  <clientmap subnet="146.0.0.1" mask="255.0.0.0">
    <assignedfsc fscid="fsc1"/>
  </clientmap>
</fscGroup>
```

To add a volume served by the **fsc1** FSC, click the **ID** button for **FSC** and then type **fsc1** in the **ID** box.

- **Filestore Group:**

```
<fscGroup id="fscGroup1"
  <filestoregroup id="fsgroup1">
    <volume id="vol1" root="/data/vol1"/>
    <volume id="vol2" root="/data/vol2"/>
  </filestoregroup>

  <filestoregroup id="fsgroup2">
    <volume id="vol3" root="/data/vol3"/>
  </filestoregroup>
  ...
</fscGroup>
```

In this case, to add a volume to the second filestore group, click the **ID** button for **Filestore Group** and type **fsgroup2** in the **ID** box.

- **Load Balancer:**

```
<fscGroup id="fscGroup1"
  <loadbalancer id="loadBal1" address="http://lb1.ugs.com:4454">
    <volume id="vol1" root="/data/vol1"/>
  </loadbalancer>
  ...
</fscGroup>
```

In this case, click the **ID** button for **Load Balancer** and type **loadBal1** in the **ID** box.

When the new volume is created, you must specify where in the FMS master configuration the definition of this volume should be placed: the **FSC element** section, the **filestore group** section, or the **load balancer** section.

Use the following FMS-related buttons as needed:

- **Reload**

Makes an FMS configuration the current and active configuration. This is useful for updating the configuration with any manual changes that are made.

- **Report**

Displays the master and slave FSCs currently configured and the status of each.

- **Display**

Displays the contents of the FMS master configuration file. It can be useful for determining what to add to a volume, for example, an FSC ID or filestore group.

8. Grant users or groups access to the volume.

9. Click **Create**.

Modifying volume properties

Warning

To ensure data integrity, modify volume properties only when no users are logged on to Teamcenter. When modifying the path name, the new path must be a valid operating system directory. Changing the path to an invalid directory results in loss of data.

1. Ensure that all users are logged off Teamcenter.
2. Select the volume to be modified from the **Organization List** tree.
Teamcenter displays the properties of the volume in the **Volumes** pane.
3. Modify information in the **Volume Name**, **Node Name**, **Machine Type** or **Path Name** boxes.
4. Click **Modify**.

Controlling volume access

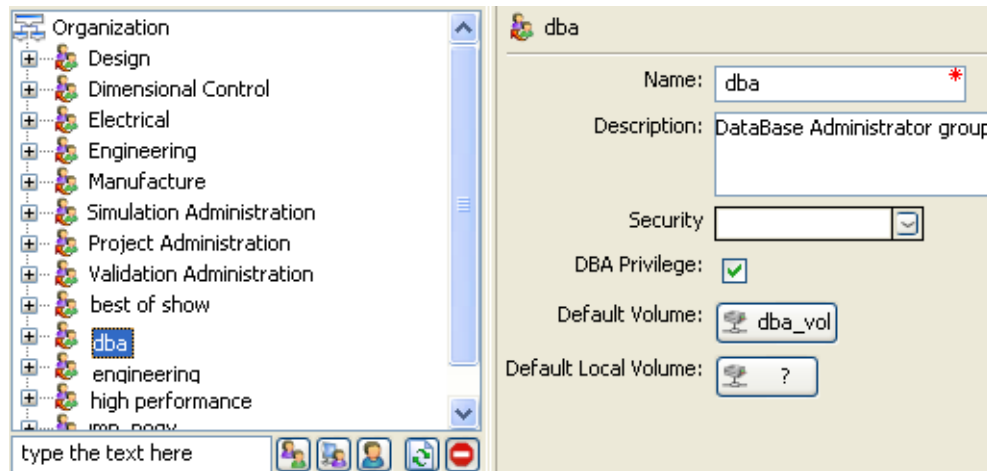
Users inherently have read access to newly created volumes. However, you must explicitly grant write access to the volume. To grant access, you must be logged on to the operating system as the root user and logged on to Teamcenter as a member of the **dba** group.

When you grant users access to volumes, the system generates a subdirectory identified by the user's name and ownership of the subdirectory is assigned to the user.

When you grant access to a group, the system generates a subdirectory identified by the group name. The system does not generate subdirectories for subgroups, regardless of whether access inheritance is enabled. Subgroups share the directory of the original group.

Granting volume access to a group does not implicitly grant access to all subgroups unless specified by the **TC_allow_inherited_group_volume_access** preference. The default value of this preference is **0**, indicating that subgroups do not inherit write access to the volume by default. Change the preference value to any non-zero number to allow inherited access.

Creating a group



As a user with **DBA** privileges, you create project-oriented groups to organize clusters of users.

You can create parent groups using the following two methods:

- Using the **Organization List** tree.

This method presents all the group properties including a parent group property for creating subgroups.

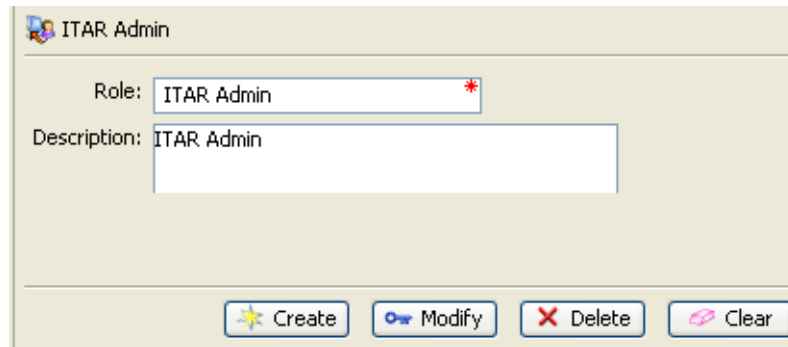
- Using the **Organization** tree.

This method presents fewer group properties, but uses wizards instead to add subgroups and roles.

Groups have several settings to configure access to data owned by the group.

- The internal/external **Security** setting allows or restricts access to data. For example, members of external groups can only access data in their group.
- The **DBA Privilege** setting, when selected, allows system administration privileges to members of the group.

Creating a role

The screenshot shows a dialog box titled "ITAR Admin" with a small icon on the left. It contains two text input fields: "Role:" and "Description:". The "Role:" field contains the text "ITAR Admin" and has a red asterisk icon to its right. The "Description:" field also contains the text "ITAR Admin". At the bottom of the dialog, there are four buttons: "Create" (with a star icon), "Modify" (with a pencil icon), "Delete" (with a red X icon), and "Clear" (with a pink eraser icon).

You create roles to reflect the skills and responsibilities of the users in your organization. Roles can be created using the Organization List method described or you can create and add a role to the **Organization** tree using the Organization Role wizard.

1. Select the top-level **Roles** node  from the **Organization List** tree.

The **Roles** pane appears.

2. Type the following information:
 - A new role in the **Role** box.
 - Optionally, a descriptive character string in the **Description** box.
3. Click **Create**.

The new role is saved in the database and displayed in the **Organization List** tree.

Creating a person

Person definitions contain real-world information about individual Teamcenter users.

Gordon, Jack

Name: *

Address:

City:

State: Zip Code:

Country:

Organization:

Employee Number:

Internal Mail Code:

E-Mail:

Telephone:

User Image: ...

Person definitions can be created:

- Simultaneously with the user definition when using the Organization User wizard.
 - Manually using the **Organization List** tree and corresponding pane.
1. Select a node from the **Persons** list in the **Organization List** tree. If you do not see nodes under the **Persons** list, double-click the top-level **Persons** node to display them.

The **Persons** pane displays the properties of the person definition.

Note

The entry in the **Name** box must be unique. You cannot create two persons with the same name.

2. Type a unique name in the **Name** box. All other boxes are optional.
3. Click **Create**.

The new person definition is saved in the database and displays in the **Organization List** tree.

These optional **Person** properties have resulting behavior:

- **E-Mail** address is required for workflow notification.

- **User Image** allows a graphic to be added for the person. You can add a picture of the person that is displayed when a **Person** object is selected.

Note

Often, the real-world information is similar for users residing in the same physical location. In such cases, you can minimize the amount of data entry required by selecting the node of a person definition (from the **Organization List** tree) that possesses similar attributes to the person you want to create. To begin a definition from scratch with a blank **Persons** pane, select the top-level **Persons** node from the **Organization List** tree.

Creating a user

You create user accounts to identify each individual who interacts with Teamcenter.

The screenshot shows a user creation form for 'Gordon, Jack (jgordon)'. The form includes the following fields and options:

- Person Name:** Gordon, Jack (with a red asterisk indicating a required field)
- User ID:** jgordon (with a red asterisk)
- OS Name:** jgordon (with a red asterisk)
- Password:** (empty field) with a checkbox for 'Clear the password'
- Last Login Time:** 07-Apr-2009 14:17, with a 'Reset' button
- Default Group:** dba (with a red asterisk)
- Default Volume:** ?
- Default Local Volume:** ?
- User Status:** Active (selected) or Inactive
- Change Ownership to:** ?
- ADA/ITAR Attributes:**
 - IP Clearance:** (empty field)
 - Gov't Clearance:** (empty field)
 - TTC Date:** No date set. (with a calendar icon)
 - Geography:** us
 - Nationality:** us
- Licensing Level:** Author or Consumer (Consumer is selected)

User definitions can be created:

- From a role in the **Organization** tree using the Organization User wizard.
- Manually using the **Organization List** tree and corresponding pane.


These optional **User** properties have resulting behavior:

- **Password** must conform to password restrictions.
- **Last Login Time** displays the last time this user logged onto Teamcenter. This helps to determine when a user is deactivated according to the **TC_days_non_login_timeout** preference. The default setting for this preference is set to 0, allowing users to always log on. When deactivation does occur, **Reset** activates the user.

1. Select the top-level **Users** node  from the **Organization List** tree.

The **Users** pane appears.

2. Complete the following information:

- a. Click  to the right of the **Person Name** box to display the **List of Defined Persons** list and select, by double-clicking, a person name from the list.

Caution

If autologon is used at the site, the operating system user name must be the same as the Teamcenter user ID and the password must be valid for both accounts or autologon does not work.

- b. Type a unique user name in the **User ID** box.
- c. Type a valid OS name in the **OS Name** box.
- d. Click **Default Group** to display the **List of Defined Groups** list and select a group from the list by double-clicking.

Default Group specifies the group the user will be placed in the organization and the default group assigned on logon.

- e. Click **Default Volume** to display the **List of Defined Volumes** list and select a volume from the list by double-clicking.

Note

Siemens PLM Software recommends that you *do not* define a default volume for each user. If the default volume is not specified, the group's default volume information is used.

- f. Click **Default Local Volume** to display the **List of Defined Volumes** and select, by double-clicking, a default local volume for the group.

Use this temporary storage to upload a file into FMS volume storage. This temporary local volume allows the file to be stored locally before it is automatically transferred to the final destination in the background. Once the file is stored in the default local volume, the user can continue working without having to wait for the upload to take place.

Note

The **Default Local Volume** value must be different from the value in the **Default Volume** box. Also, either box, the **Default Volume** box and the **Default Local Volume** box, can be set without the other being set.

- g. Select **User Status**, either **Active** or **Inactive**. The default setting is **Active**.

- h. You can assign authorized data access (ADA) and International Traffic in Arms Regulations (ITAR) attributes to a user using the boxes in the **ADA/ITAR Attributes** section.

- **IP clearance**

Specifies the intellectual property (IP) clearance level, which is the level of access the user has to sensitive (classified) information. This box is optional.

- **Government clearance**

Specifies the level of clearance that users have to classified data. This box is optional.

- **TTC date**

Specifies the technology transfer certification (TTC) date, which is the date when the user's qualification for viewing exporting data marked as government classified lapses. Teamcenter revokes the user's access rights after the TTC date expires unless renewed. As administrator, you can manually cancel a user's TTC date at any time. This box is optional.

- **Geography**

Specifies the geographical location of the user. Appropriate values are two-character codes from ISO 3166. If not specified, the user is assumed to be at the same location as the database. This box is optional.

- **Nationality**

Specifies the nationality of the user, which you can set using the LOV containing two-character codes from ISO 3166. This box is optional.

- i. Select **Licensing Level**, either **Author** or **Consumer**. The default setting is **Consumer**.

- **Author** is a user that creates or modifies data for product and process information.

- **Consumer** is a user that views, approves, rejects, or comments on product and process information.

3. Click **Create**.

The new user definition is saved in the database.

Note

A default role, as defined by the default group, is associated with the user definition.

Repeat steps 2 and 3 to create additional users.

Group member settings

Group Member Settings

Group Administrator:

☒

Group Member Status:

☒ Active

☐ Inactive

Default Role:



☐

Externally Managed:

☐



Setting	Description
Group Administrator	<p>A group administrator is a group member with the following special privileges, such as:</p> <ul style="list-style-type: none">• Adding or modifying group members.• Adding or modifying person information for group members. <p>Group administrators must be members of the group they are administering and these privileges are only valid within that group.</p>
Group Member Status	<p>You can designate the group member status as either Active or Inactive. The default setting is Active.</p>
Default Role	<p>You can add one or more roles to the group. The first role added to the group becomes the <i>default</i> role for that group. A <i>default</i> role, as defined by the default group, is associated with the user definition.</p>
Externally Managed	<p>You can organize your user base on a corporate LDAP directory server that is also used as a central user authentication repository for applications such as Teamcenter. Users maintained in this manner are considered to be externally managed. This user data can be synchronized with Teamcenter using the ldapsync utility.</p> <p>Externally managed user, group, and role attributes mapped between the LDAP server and the Teamcenter database cannot be updated using the Organization application.</p> <p>Users, groups, and roles created and maintained in Teamcenter are considered to be <i>internally</i> managed. User constructs that are synchronized from an external directory at one site and distributed to remote sites are considered to be <i>remotely</i> managed.</p>

Creating an authorized data access license

1. Select the **ADALicenses** node  from the **Organization List** tree.
The **ADALicense** pane appears.
2. Type the license ID.
3. Select the license type from the **License Type** list:
 - **IP_License**
 - **ITAR_License**
 - **Exclude_License**
4. Set the license expiration date by selecting a date and time using the **License Expiry** box. Click **Calendar**  to display the calendar.
5. (Optional) Type a reason in the **Reason** box.
6. You can set the licenses for users and groups. To set licenses on a list of users, select each user by searching on their user name. Enter the user name in the **Search on Username** box.

If you do not know the user name for which to search, click the **Load All** button to display a list of all users in the **List of Users** box.

As you select each user, the name is copied to the **List of Users** box.

7. From the **List of Users** box, select the user or users to place in the **List of Licensed Users** box and click the **Add** button .
8. To assign groups, click the **Groups** tab. From the **List of Groups** box, select the group or groups to place in the **List of Licensed Groups** box and click **Add** button .
9. Click **Create**.

Add a new role to a group using the Organization Role wizard

You can use the Organization Role wizard to add a new role to a group during the process of creating the group or subgroup in the **Organization** tree.

New roles can also be added to existing groups or subgroups in the **Organization** tree. The procedure for using the Organization Role wizard to add a new role is the same, regardless of the activity being performed when the wizard is invoked.

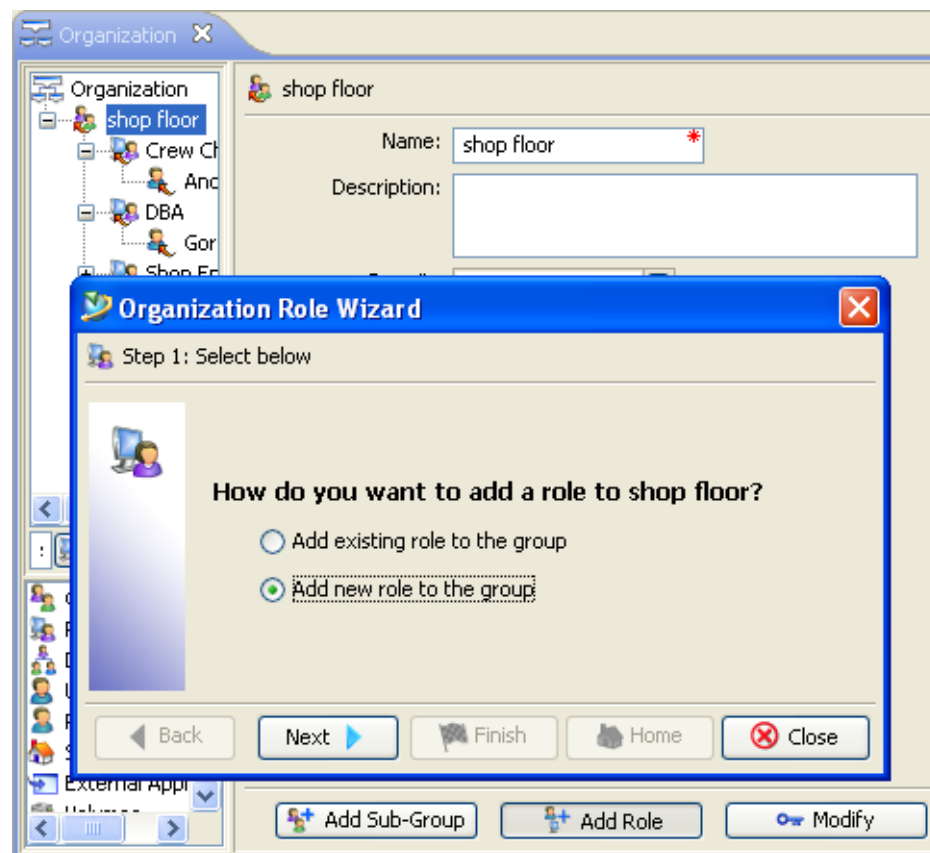
1. Select the group node in the **Organization** tree to which you want to add the role.

The **Groups** pane appears.

2. Click **Add Role**.

The Organization Role wizard appears.

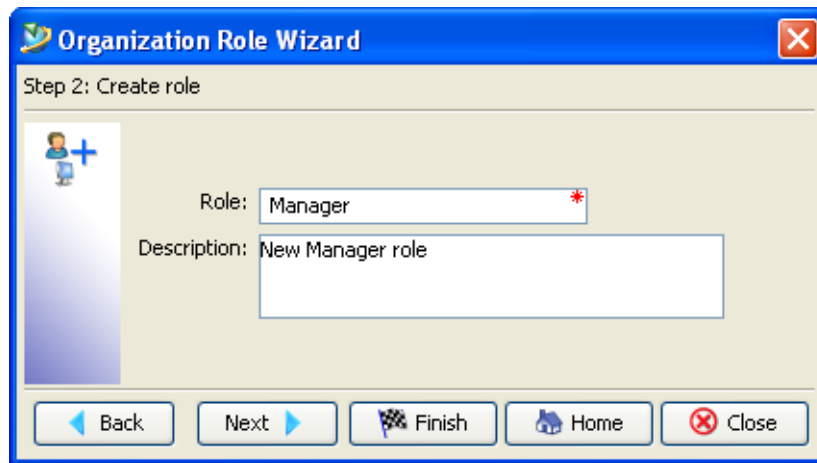
3. Select **Add new role to the group** and click **Next**.



Note

The first role added to a group becomes the default role for that group.

4. Type the following information and click **Next** or **Finish**:
 - A new role in the **Role** box.
 - A descriptive character string in the **Description** box.



5. Click **Yes** to create the new role.
The **Role(s) added** dialog box appears. Click **OK**.
6. Perform one of the following actions:



- Select a **What is next?** option from the wizard. You can add another role to the selected group or add a user to the role you just added.
- Click **Home** to return to step 1 of the Organization Role wizard.
- Click **Close** to dismiss the wizard.

The role appears in the **Organization** and **Organization List** trees.

Add a new user to a group/role using the Organization User wizard

You can add a new user to the **Organization** tree as part of the process of creating the group/role hierarchy.

You can also add a new user to an existing group/role combination in the **Organization** tree. The procedure for using the Organization User wizard to add a new user is the same, regardless of the activity being performed when the wizard is invoked.

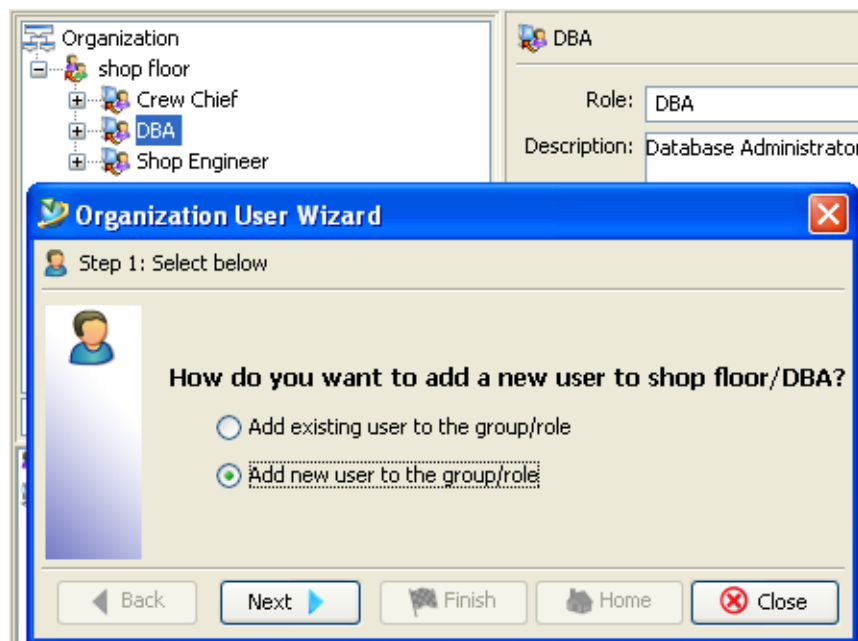
1. Select a role node from the **Organization** tree.

The **Roles** pane appears.

2. Click **Add User**.

The Organization User wizard appears.

3. Select **Add new user to the group/role** and click **Next**.



4. Create the user by performing the following substeps:
 - a. Define the person to be associated with this user. If the person definition already exists, click **Person Name** to display the **List of Defined Persons** list and select, by double-clicking, a person name from the list. If a person definition does not yet exist, type the name of the individual in the **Person Name** box.
 - b. Type a unique user name in the **User ID** box.
 - c. Type a valid OS name in the **OS Name** box.

- d. Optionally, type a password. **Default Group** and **Roles** are already filled in for you.
- e. Click **Default Volume** to display the **List of Defined Volumes** list and select, by double-clicking, a default volume for the group.

The screenshot shows the 'Organization User Wizard' dialog box at 'Step 2: Create user'. The fields are as follows:

- Person Name: Andretti, Maria
- User ID: mandretti
- OS Name: mandretti
- Password: masked with asterisks
- Default Group: shop floor
- Roles: DBA
- Default Volume: std_vol
- Default Local Volume: ?

Navigation buttons at the bottom: Back, Next, Finish, Home, Close.

Caution

If you create a group without assigning a default volume, group members cannot save datasets. Therefore, Siemens PLM Software recommends that you assign a default volume for the group.

- f. Click **Default Local Volume** to display the **List of Defined Volumes** list and select, by double-clicking, a default local volume for the group.

Note

The **Default Local Volume** value must be different from the value in the **Default Volume** box.

- g. When all required input is complete, click **Next** or **Finish**.
- h. Click **Yes** to create the new user.

If you created a new person definition while creating the new user, go to step 5. If you did not create a new person definition, go to step 6.

5. The **Create Person** dialog box is displayed. Click **Yes** to confirm that you want to create the new person.

6. The **User(s) added** dialog box appears. Click **OK**.

The **User(s) added** dialog box is dismissed and you are returned to step 1 of the Organization User wizard.

The new user appears as a child of the selected role in the **Organization** tree. Additionally, the new user and person (if applicable) are displayed in the **Organization List** tree.

Note

Persons are not displayed in the **Organization** tree.

7. Click **Close** to dismiss the Organization User wizard or repeat steps 3 through 6 of this section to add additional users to the same role/group combination.

Add an existing user to a role/group using the Organization User wizard

The Organization User wizard can be used to add an existing user to a group during the process of creating a group/role combination in the **Organization** tree.

Existing users can also be added to existing group/role combinations within the **Organization** tree. The procedure for using the Organization User wizard to add an existing user is the same, regardless of the activity being performed when the wizard is invoked.

1. Select a role node from the **Organization** tree.

The **Roles** pane is displayed.

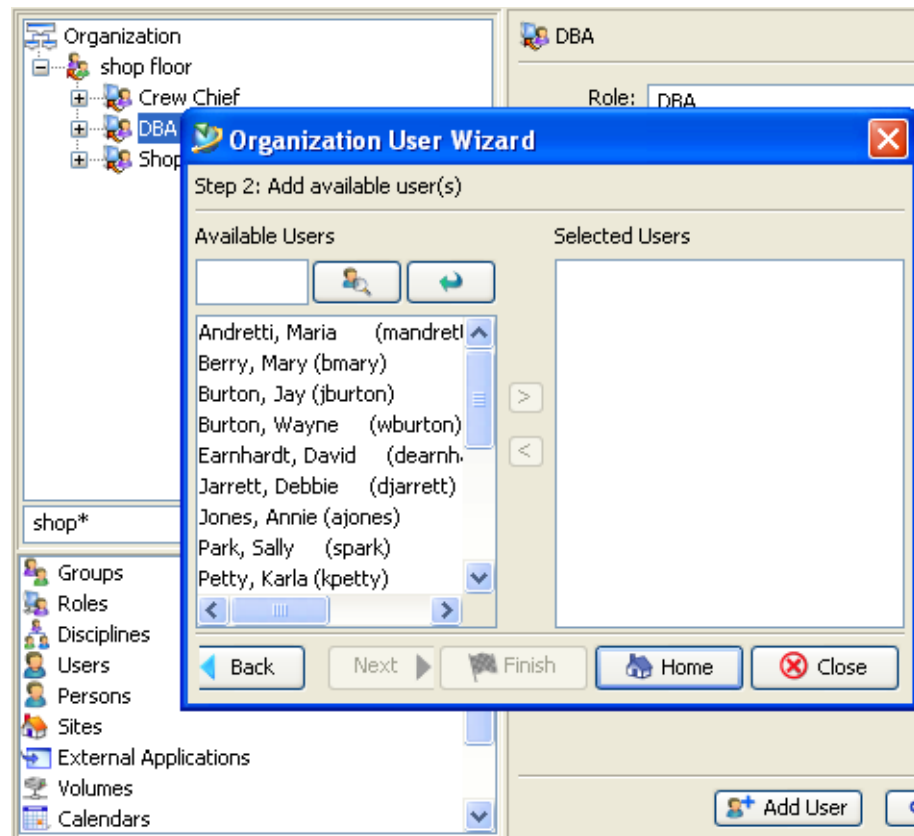
2. Click **Add User**.

The Organization User wizard is displayed.



3. Select **Add existing user to the group/role** and click **Next**.



4. Select the users to add from the **Available Users** list.



You can also use either of the following buttons in the search pane:

- Find users 
- Reload all available users 

You can move items between the **Available Users** and **Selected Users** lists by double-clicking a user or selecting a user and clicking the plus (+) or minus (–) buttons. After you select all the users to be added, click **Next** or **Finish**.

5. A message appears asking if you want to add the selected users. Click **Yes**.
6. Click **OK**.

The **User(s) added** dialog box closes and you are returned to step 1 on the Organization User wizard.

The user appears in the **Organization** tree as a child of the selected role.

7. Click **Close** or repeat steps 1 through 6 to add additional users to the same role/group combination.

Changing user status

Siemens PLM Software recommends that you first deactivate an obsolete user account in the Teamcenter database, then delete it when all references to the account cease.

Because the names of inactive users do not display in the list of values (LOVs) throughout the rich client, inactive accounts cannot continue to be referenced by other users. Inactive user accounts cannot be logged on to the database, yet account records remain in the database so that an audit trail can continue to reference the data.

An account can be deleted when all references to objects owned by the deactivated account are cleared from the database, and an audit trail of the account's actions is no longer required.

Deactivate a user account

1. Perform the following:
 - Select a user definition from the **Organization** tree.

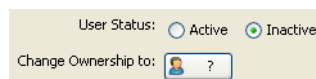
The **Users** pane displays the properties of the user definition.

Note

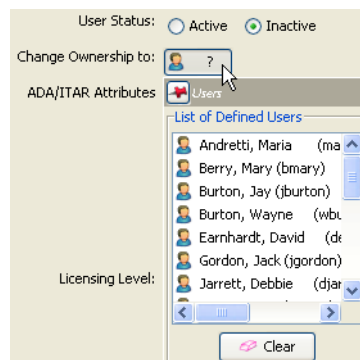
Because database objects are owned by users, you must decide what to do with any objects owned by the user being deactivated. You can either change ownership of these objects to another user or allow ownership to be retained by the inactive user.

2. Click **Inactive**.

The **Change Ownership** button becomes available.



3. Perform one of the following substeps:
 - If you want to change the ownership of the user's database objects, go to step 4.
 - If you want ownership of the database objects to be retained by the inactive user, go to step 6.
4. Click **Change Ownership**.

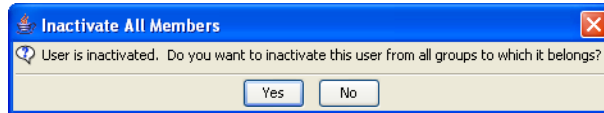


5. Select a new owner for the database objects from the **List of Defined Users** list (double-click a user).

The name of the new owner is displayed on the **Change Ownership** button.

6. Click **Modify**.

7. If the **Inactivate All Members** dialog box displays and you want to set the user's **Group Member Status** to **Inactive** for all groups that the user belongs to, click **Yes**. If you want to keep the user's **Group Member Status** set as they currently are for all groups, click **No**.



The user account is deactivated.

Activate a user account

1. Perform the following:
 - Select a user definition from the **Organization** tree.
2. Click **Active**.
3. Click **Modify**.






The user account is activated.

Using Organization find

If you experience difficulty browsing through the organization hierarchy to find a certain group, role, or user, you can use the Organization find function to locate it.



Use any of the following buttons in the **Organization** tree pane:

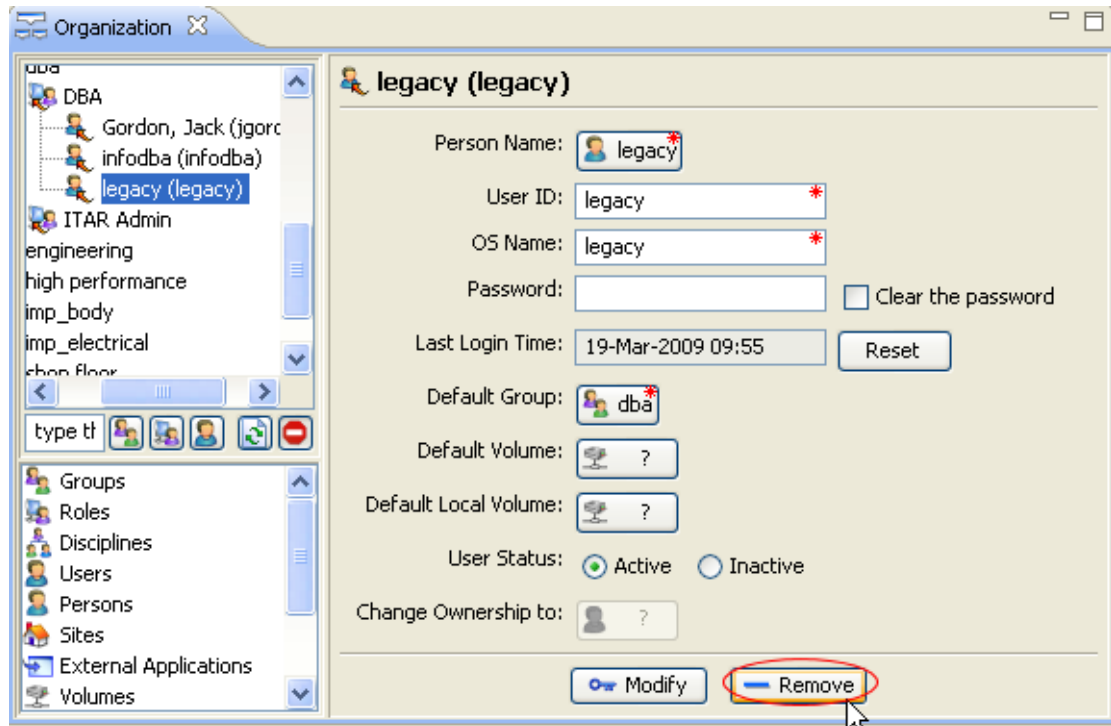
- Find group 
- Find role 
- Find user 
- Reload the **Organization** tree 
- Suppress/display inactive group members 

The Organization find feature works in an identical way for each mode. To use Organization find, type the text into the box and click the appropriate button. Note that the wildcard (*) character is accepted. The **Organization** tree is reloaded with the results of your search. If there are no matches, a message informing you of this is displayed.

Managing group members

As your real-world organization evolves and changes, your Teamcenter virtual organization also changes. Implementing new projects, promoting personnel, and restructuring your organization are all examples of real-world events that would necessitate changes involving group members.

Remove a member from a group



Note

You cannot remove the last instance of a user from the **Organization** tree if the group from which you are removing the user is the user's default group.

1. Select the user (group member) you want to remove from a group or subgroup in the **Organization** tree.

Teamcenter displays the user's information.

2. Click **Remove**.

The system displays the **Remove User Confirmation** dialog box.

3. Click **Yes** to remove the user from the group.

Activate a group member

1. Select the user (group member) you want to activate in a group or subgroup in the **Organization** tree.

Teamcenter displays the user's information.

2. In the **Group Member Settings** section, choose the **Active** option.

User Status: ☒ Active ☐ Inactive

3. Click **Modify**.

Deactivate a group member

When a user leaves the organization or changes groups or roles within the organization, you can deactivate their membership within a group. This prevents them from logging in to the system as a member of the group and denies them access to information related to their previous group and role.

Note

- Only an administrator or a user designated as a group administrator can change a group member's status from active to inactive.

Database objects are owned by individual users; therefore, object ownership does not change when a group member is deactivated.

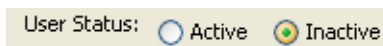
- You cannot deactivate a group member if they have any pending Workflow tasks. You must first delegate these tasks to another group member and then deactivate the user. You can use the **global_transfer** utility to transfer one user's tasks to another user.

You can also reassign the user's tasks using the My Teamcenter inbox feature.

1. Select the user (group member) you want to deactivate from a group or subgroup in the **Organization** tree.


Teamcenter displays the user's information.

2. In the **Group Member Settings** section, choose the **Inactive** option.



3. Click **Modify**.

Suppress the display of inactive group members in the Organization tree

1. Expand the group in the **Organization** tree to display the roles and users within the group.
2. Click **Suppress Inactive Group Members/Show Inactive Group Members** .

Teamcenter filters the display to suppress group members who have been designated as inactive within a group or groups.

Note

This feature suppresses the display of active and inactive users who are designated as inactive group members. However, users can be designated as inactive but not be designated as inactive group members. In this case, the users are still displayed as group members when the **Suppress Inactive Group Member** filter is applied.

You can restore the display of inactive group members in the tree by clicking the button again.

Activities

In the *Organization* section, do the following activities:

- Create a new group and role.
- Add a new user to a new group and role.

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.

Instructor Note:

Answers to review questions

1. True
2. Special access privileges for data owned by the group.

Utility account generation

You can use the **make_user** utility to create new users, groups, persons, roles, and volumes outside of a Teamcenter session.

The **make_user** utility allows you to:

- Create your organization from a command line or script.
- Modify properties of existing user, group, and role objects.
- Set group and volume defaults.
- Create more than one user at a time.

All users created become members of the specified group.

The **make_user** utility supports batch mode processing using an input file. You can run the **make_user** utility script multiple times.

Warning

The **make_user** utility cannot be used to delete the organization objects.

Use the **make_user** utility script to:

- Load your entire organization with one command.
- Back up your organization.
- Populate your test and production environments.

The make_user utility 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

make_user -p="Smith, John" -u=smith -g=design
-role=engineer
```

Note

The first group listed is the default group. If the group and role do not exist, they are automatically created.

- **Create an organizational hierarchy using a file associated with the -file argument**

Example

```
make_user -u=smith-p=smith -g=dba -file=org.txt
```

The **-file** argument specifies that the input file is read to create users or to modify existing users, groups and roles after other arguments are processed.

Each record in the file contains the following information:

```
person|user|password|group|role||option_name1|option_value1|
option_name2|option_value2|...|update
```

option_name is any command line argument and *option_value* is a valid value for *option_name*.

Each field is delimited by the (|) character.

The password and role fields can be null (| |).

The role defaults to the last value specified in either the file or on the command line using the **-role** argument.

If a password is not specified for the new user, Teamcenter assigns the user ID as the password.

When modifying an existing user, group, or role, specify the properties to be modified by *option_name* | *option_value* pairs followed by the **-update** option.

- **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
```

Activity

In the *Organization* section, do the following activity:

- Create the organization hierarchy using the `make_user` utility.

Introduction to Authorization

Authorization enables you to control the display of Teamcenter administrative applications and utilities to users based on their group membership or their role in a group.

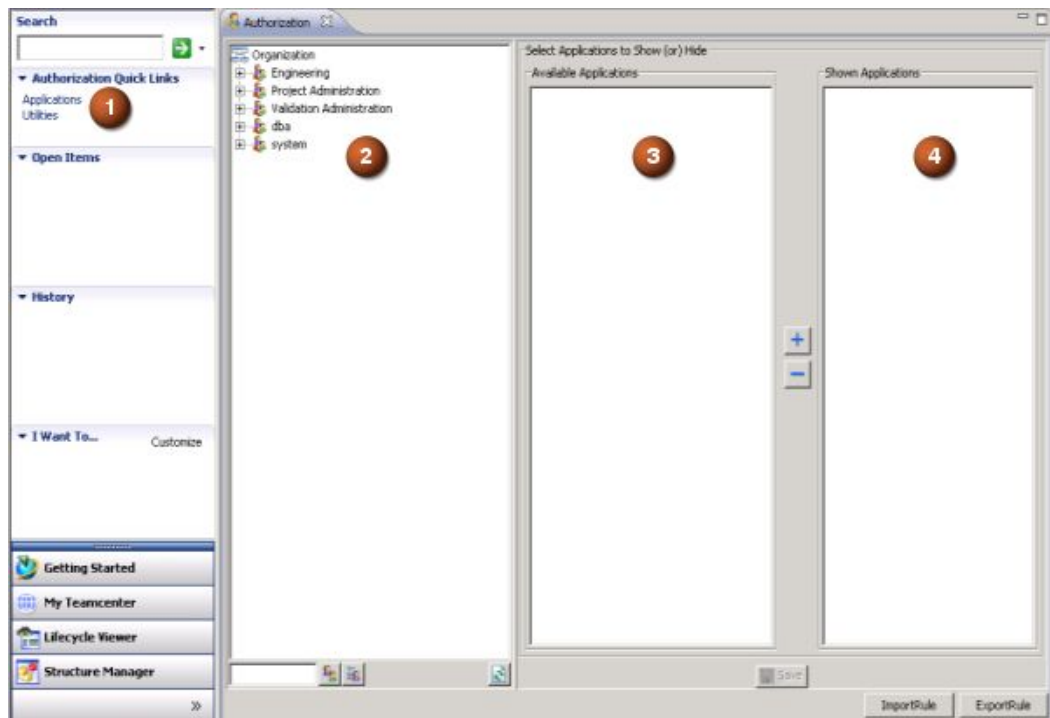
For example, you can:

- Grant all members of the **special admin** group access to the Organization application, regardless of their role within the group.
- Grant users who occupy the **Organization Admin** role in the **special admin** group access to the Organization, Workflow Designer and Access Manager applications.

Authorization works in conjunction with other Teamcenter applications to control access to product features and data, as follows:

- Access to product features is controlled using the Command Suppression application.
- Access to operations on objects, such as delete, copy, and change ownership, is controlled by configuring rules in Access Manager.

Authorization interface



- | | | |
|---|----------------------------------|--|
| 1 | Authorization Quick Links | Enables you to choose either utilities or applications for configuration. |
| 2 | Organization tree | Displays the groups and roles in your organization. You can choose which utilities or administrative applications are displayed in the interface for a selected group or for a selected role within a group. |
| 3 | Available Applications | Displays the list of administrative utilities or applications that can be shown or hidden. |
| 4 | Shown Applications | Displays the list of administrative utilities or applications that are shown in the interface for the selected group or role in group. |

Basic concepts of using Authorization

Authorization rules allow you to control access to Teamcenter administrative applications and utilities based on groups. System-level rules are delivered as part of your Teamcenter installation, and you can create additional rules to support your business processes using the Authorization application.

System-level authorization rules

System-level authorization rules are those rules delivered as part of your standard Teamcenter installation that govern access to administrative applications and utilities. By default, Teamcenter supplies two groups for administrative purposes, the **Project Administration** group and the **dba** group.

Project Administration group members only have access to the Project application, which allows them to create, delete, modify, and add users to or remove users from projects. **dba** group members are granted access to all Teamcenter administrative applications and utilities.

Often, administrative tasks are assigned at a functional level corresponding to your business practices. For example, responsibility for administering user data such as personal and organization information may be assigned to one group, while a different group may be responsible for designing workflow processes. In such cases, **dba** group privileges are more broad and powerful than is necessary or desirable. Authorization enables you to create authorization rules to model access to administrative tools to your business processes.

Basic tasks using Authorization

Use Authorization to perform the following tasks:

- Configure access to Teamcenter administrative applications.
- Configure access to Teamcenter utilities.
- Share administration authorization rules with other Teamcenter sites.

Import and export organization

The **dsa_util** utility allows you to distribute system administration data, such as organization data, from one site to another. This utility should be used only for the initial migration of system objects. Siemens PLM Software does not recommend to use it to maintain system objects.

The **dsa_util** utility propagates Teamcenter administration data among multiple sites and allows administrators to:

- Manage system data from a central site.
- Support nonnetworked sites.
- Create reports of the results of a distribution operation.

Export organization

To export, use the **dsa_util** utility with the following options.

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=User -filename=OrgExp.xml
```

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.
-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 -filename=OrgExp.xml
```

Activities

In the *Organization* section, do the following activities:

- Authorize access to the selected Teamcenter applications.
- Import an organization hierarchy using the dsa_util utility.
- Search for the organization.

Review questions

1. You can run the **make_user** utility script multiple times.
 - True
 - False

Instructor Note:

Answers to review questions

1. True

Summary

The following topics were taught in this lesson:

- The organization hierarchy includes persons, users, roles, and groups.
- Create groups and roles to represent your projects and functional responsibilities.
- Add users to groups to define their place in the organization.
- Use the search function to find groups, roles, or persons.
- Increase productivity by using a utility to load your organization.
- The **dsa_util** utility allows you to import and export organization data.

Lesson

16 More Teamcenter Options and Utilities

Purpose

The purpose of this lesson is to introduce using utilities to import data into .

Objectives

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

- Use `import_file`.
- Use `plmxml_import`.

Help topics

Additional information for this lesson can be found in:

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

Instructor Note:

Lesson instructor notes.

import_file

Imports files into the Teamcenter database according to a set of user-specified arguments. These arguments supply user identification information, dataset information, and (optionally) item information to be associated with the imported file. The arguments may be specified on the command line to import a single data file or in a file to import multiple data files (bulk import).

Depending on the arguments, each data file is copied (an **ImanFile** object is created), a dataset is created (or modified), and if specified, an item is created or modified to contain the dataset. In the absence of a specified item, the dataset is placed in the user's **Newstuff** folder.

Note

The **import_file** utility does not support the creation of custom item types.

SYNTAX

```
import_file -u=user-id -p=password -g=group  
-f=file-name | -i=file-name [-vb] [-log=file-name]  
-type=datasettype -d=dataset-name -ref=named-reference [-de={n  
| e | a | r}] [-item=item-id] [-revision=item-rev-num] [-ie={n  
| y}] [-desc=string] [-v=volume-name]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally **infodba** or another user with administration privileges. If this argument is used without a value, the operating system user name is used.

Note

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

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-id* value to be the password.

-g

Specifies the group associated with the user.

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

-f

Imports a single file into Teamcenter. The full path must be provided if the file does not reside in the current working directory. The **-f** and **-i** arguments are mutually exclusive. See example 1.

-i

Imports multiple files into Teamcenter using a specified import file. The full path must be provided if the file does not reside in the current working directory. The **-f** and **-i** arguments are mutually exclusive. See example 2.

-vb

Runs utility in verbose mode. Displays maximum amount of information. Nonverbose sessions only display error messages.

-log

Creates a log of items and datasets created.

-type

Defines the dataset type in Teamcenter, for example, **TEXT** or **UGPART** datasets.

-d

Specifies the name of the dataset into which the file is imported.

-ref

Specifies the type of named reference associated with the file. The value specified by this argument may or may not be identical to the value specified by the **-type** argument.

For example, **TEXT** or **UGPART** type datasets have named references of **TEXT** and **UGPART**, respectively. However, **DirectModel** type datasets have a **JTPART** named reference. Each dataset type defines one or more named references to be associated with it. See restriction #2.

For more information, see the *Rich Client Interface Guide*Type.

-de

Indicates that a dataset exists. Used when a dataset of the same name already exists.

=n

Specifies that a new dataset be added even if one with the same name exists. If it does exist, it is added to the same item folder. If it does not exist, it is placed in the new item folder or the user's **Newstuff** folder.

=e

Specifies that the dataset should be added if it already exists and that this dataset type supports multiple instances of the same dataset.

=a

Specifies that the imported file be added as a named reference to the existing dataset. When this is done, a new dataset version that contains the additional imported named reference file is created.

=r

Specifies that a new dataset revision be created and the existing named reference be replaced with the new one. This option generates an error if the dataset has no existing named reference.

-item

Specifies the name of the item containing the dataset that references the imported file.

-revision

Specifies the item revision number and revision ID. See restriction #3.

-ie

Specifies behavior if the item already exists.

=n

Specifies that the dataset will not be added if the item already exists.

=y

Specifies that the dataset may be added if the item already exists. If the item exists, but the item revision does not, an item revision is created.

-desc

Specifies a user-defined text description of an item that is created by the import function. If the **import_file** utility is creating a new revision of an existing item, this is the description of the item revision.

-v

Specifies the full path of the Teamcenter volume where the imported file is placed.

ENVIRONMENT

As specified in ***Unsatisfied xref title***.

FILES

As specified in *Basic concepts about Project*.

RESTRICTIONS

1. When importing a file, the file name cannot be longer than 31 characters.

2. To create a dataset in Teamcenter, the user must specify the dataset type and the named reference.
3. When importing a file as a dataset, you must specify the named reference using the **-ref** argument.
4. When importing a file into an item or item revision, you must specify the revision; otherwise, an error message displays indicating a missing revision.

EXAMPLES

- To import a single operating system file, **bike.dat**, into Teamcenter as a **UGPART** dataset named **my_bike_dataset**, enter the following on a single line:

```
$TC_ROOT/bin/import_file
-user=user-id -p=password -g=group -f=bike.dat
-type=UGPART -d=my_bike_dataset -ref=UGPART
```

- To import multiple operating system files into Teamcenter, first create an input file that contains the following information:

```
-f=bike1.dat -d=my_bike1_dataset -type=UGPART -ref=UGPART
-f=bike2.dat -d=my_bike2_dataset -type=UGPART -ref=UGPART
.
-f=binkeN.dat -d=my_bikeN_dataset -type=UGPART -ref=UGPART
```

- Run the **import_file** utility using the input file from example 2, entering the following command on a single line:

```
$TC_ROOT/bin/import_file
-user=user-id -password=password -group=group -i=input-file
```

- Import the **d:\some_file.jt** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\some_file.jt -type=DirectModel -d=my_jt_file_dataset -ref=JTPART
```

- Import the **d:\WordDoc.doc** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\WordDoc.doc -type=MSWord -d=my_word_dataset -ref=word
```

- Import the **d:\ExcelFileTest.xls** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\ExcelFileTest.xls -type=MSEExcel -d=my_excel_dataset -ref=excel
```

- Import the **d:\myfile.txt** file:

```
%TC_ROOT%\bin\import_file -user=user-id -p=password -group=group
-f=d:\myfile.txt -type=Text -d=my_text_file_dataset -ref=Text
```

plmxml_import

Imports objects to Teamcenter from a PLM XML file. In case there are files for import as determined by transfer mode, the utility looks for files in the path specified by the **xml_file** parameter. This utility is also used to import workflow templates.

The Business Modeler IDE application maintains most of the objects that affect the data model. To avoid data accuracy issues, any PLM XML file that may contain one or more of the following objects should not be imported using **plmxml_import** utility.

AliasTypeDef	GRMRule	ProcessTypeDef
ApplicationInterface	HideTypeRule (Type-Display rule)	PropBusinessOperation
AppearanceGroupTypeDef	IdContextRule	RelationTypeDef
ActivityTypeDef	ImanTypeDef	StatusTypeDef
CannedMethodRule	ItemTypeDef	StorageMediaTypeDef
ChangeTypeDef	ListOfValues	ToolTypeDef
CompoundPropDefRule	NamingRule	TypeBusinessOperation
DatasetTypeDef	NameFieldRule	UOMTypeDef
DeepCopyRule	NoteTypeDef	ViewTypeDef
Extension	OccurrenceTypeDef	WorkAreaTypeDef
FolderTypeDef	OperationTypeDef	
FormTypeDef	PropertyRule	

SYNTAX

```
plmxml_import [-u=user-id -p=password -g=group]  
-xml_file=name-of-xml-file -transfermode=transfermode-name  
[-type=object-type-to-import] [-log=log-file-name] [-s=TRUE | FALSE]  
[-import_mode=overwrite | ignore] [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally **infodba** or another user with administration privileges. If this argument is used without a value, the operating system user name is used.

Note

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

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-id* value to be the password.

-g

Specifies the group associated with the user.

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

-xml_file

Specifies the full path of the file name from which the data is imported.

-transfermode

Specifies the name of a transfer mode used to import the objects. This transfer mode specifies the traversal rules, filter rules, and property sets to be used for import. It determines what is imported from the system. If not specified, a default transfer mode is used. In addition the following transfer mode values can be applied to import workflow templates.

workflow_template_import

Use to create a new template. If the template already exists in the database, the command is ignored.

workflow_template_overwrite

Use to overwrite an existing template. A new version of the existing template is created in the database. If the template does not already exist, a new template is created.

-type

Specifies the name of the PLM XML element type. Only objects of this element type would be imported from the file.

-log

Specifies the name of a file to be used as the log file. If you do not specify a file name, a file is created using the name of the file specified by the **-xml_file** option. The log file name format is *file-name_log.txt*.

-s

Indicates that all imported objects are placed in a newly created folder within the user's **Newstuff** folder. The new folder is named using the right-most 32 characters of the XML file name. The default value for this argument is **FALSE**, and the behavior is not to save imported objects in the newly created folder unless this argument is set to **TRUE**.

Note

Imported objects are saved in the system regardless of this argument. This argument controls the placement of imported objects into a folder. Multiple imports of the same object with the **-s** flag set results in multiple folders containing references to the same object.

-import_mode

Specifies the mode in which import is handled for PLM XML Import/Export configuration objects. In **overwrite** mode, objects that already exist in the database are overwritten. In **ignore** mode, the imported object is ignored if the imported object already exists in the database. The PLM XML Import/Export configuration objects for which this option is applicable are as follows:

TransferMode
ClosureRule
PropertySet
FilterRule

-h

Displays help for this utility.

ENVIRONMENT

As specified in ***Unsatisfied xref title***.

FILES

As specified in *Basic concepts about Project*.

RESTRICTIONS

Siemens PLM Software recommends you do *not* use the following transfer modes with this utility:

JTDataImportDefault
TIEImportDefault

EXAMPLES

- To import all objects in the XML file using the default context, enter the following command on a single line:

```
plmxml_import -u=infodba -p=infodba -g=dba -xml_file=abc.xml
```

If errors are detected during the import operation, a log file named **abc_log.txt** is created.

- To import Organization elements from the XML file that correspond to groups in Teamcenter, enter the following command on a single line:

```
plmxml_import -u=infodba -p=infodba -g=dba -xml_file=abc.xml  
-type=Organization -log=mylog.txt
```

- To import all objects in the XML file using the default context and save them in a newly created folder within the user's **Newstuff** folder, enter the following command on a single line:

```
plmxml_import -u=bob  
-p=password -g=group-name -xml_file=myxml.xml -s=TRUE
```

- To import a new workflow template without overwriting existing templates, enter the following command on a single line:

```
plmxml_import -u=infodba -p=infodba -g=dba -xml_file=wkf_templates.xml  
-transfermode=workflow_template_import
```

- To import a workflow template and overwrite an existing template, creating a new version of the template in the database, enter the following command on a single line:

```
plmxml_import -u=infodba -p=infodba -g=dba -xml_file=wkf_templates.xml  
-transfermode=workflow_template_overwrite
```

plmxml_export

Exports objects from Teamcenter in PLM XML format. If there are files for export, as determined by transfer mode, a directory is created and the files are exported into that directory. The directory is named using the specified file name without the **.xml** extension.

This utility is also used to extract file information from the database into a PLM XML file to prepopulate FSC. For more information, see the **generate_loadfscache_tickets** and **fscadmin** utilities.

The Business Modeler IDE application maintains most of the objects that affect the data model. Any export operation that can result in one or more of the following objects is not recommended because the generated XML file should not be imported using **plmxml_import** utility to avoid data accuracy issues.

ActivityTypeDef	GRMRule	PropBusinessOperation
AliasTypeDef	HideTypeRule (Type-Display rule)	PropertyRule
AppearanceGroupTypeDef	IdContextRule	RelationTypeDef
ApplicationInterface	ImanTypeDef	StatusTypeDef
CannedMethodRule	ItemTypeDef	StorageMediaTypeDef
ChangeTypeDef	ListOfValues	ToolTypeDef
CompoundPropDefRule	NameFieldRule	TypeBusinessOperation
DatasetTypeDef	NamingRule	UOMTypeDef
DeepCopyRule	NoteTypeDef	ViewTypeDef
Extension	OccurrenceTypeDef	WorkAreaTypeDef
FolderTypeDef	OperationTypeDef	
FormTypeDef	ProcessTypeDef	

SYNTAX

```
plmxml_export [-u=user-id -p=password -g=group]
               -xml_file=xml-file-name -transfermode=transfermode-name
               {-item=item-id [-rev=item-revision-id] [-export_bom=yes | no]
               [-rev_rule=revison-rule] [-svrule=saved-variant-rule]
               | -class=class-name | -type=type-name -ics_class=ics-class-name
               |
               -imantypedef=iman-type -uid=uid-of-object -foldername=folder-to-export} [-h]
```

ARGUMENTS

-u

Specifies the user ID.

This is generally **infodba** or another user with administration privileges. If this argument is used without a value, the operating system user name is used.

Note

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

-p

Specifies the password.

If used without a value, the system assumes a null value.

If this argument is not used, the system assumes the *user-id* value to be the password.

-g

Specifies the group associated with the user.

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

-xml_file

Specifies the full path of the file to which the data is exported.

-transfermode

Specifies the name of the transfer mode used to export the objects. This transfer mode specifies the traversal rules, filter rules, and property sets to be used for export. It determines what is exported from the system. If not specified, a default transfer mode is used. If **-transfermode** is set to **justDatasetsOut**, you must specify a revision ID using the **-rev** argument.

-item

Specifies the ID of the item to be exported.

-rev

Specifies the revision ID of the item to be exported. If not specified, the configured revision (either specified or default) is exported for the item.

-rev_rule

Specifies the revision rule applied to export the BOM. This is also used to determine the configured revision if the **-rev** option is not specified.

If this option is not specified, the default revision rule is applied.

-svrule

Specifies the name of the saved variant rule to be applied for the BOM window configuration.

-class

Exports all instances of a given class.

Note

- You cannot use the **-class** argument to export scope rules (**TransferModes**, **ClosureRules**, **FilterRules**, **PropertySets**, **ActionRules**, and **TransferOptionSets**). Use the **tcplmxml_export** utility.
- A list of classes can be obtained from schema browsing utilities such as the Schema Editor in the Portal Admin group or the **sb** command line ITK utility.
- If **ListOfValues** and **BusinessRule** are specified, all instances of subclasses of specified class are also exported.

Options to export organization information using **-class** are shown in the following table:

Option	Description
Person	Export all persons
User	Export all users
Role	Export all roles
Group	Export all groups
GroupMember	Export all group members
POM_imc	Export all sites
ImanVolume	Export volumes
ListOfValues	Export all list of values
ListOfValuesString	Export string list of values
ListOfValuesDate	Export date list of values
ListOfValuesDouble	Export double list of values
ListOfValuesInteger	Export integer list of values
ListOfValuesChar	Export char list of values
ListOfValuesTag	Export reference list of values

Options to export workspace objects using **-class** are shown in the following table:

Option	Description
Item	Export all instances of item
ItemRevision	Export all instances of item revisions
Folder	Export all instances of folder
Form	Export all instances of forms
Dataset	Export all instances of datasets
Alias	Export all instances of alias
EPMJob	Export all instances of workflow jobs
PSBOMView	Export all instances of BOM view
PSBOMViewRevision	Export all instances of BOM view revision
Tool	Export all instances of tool

Options to export business rules using **-class** are shown in the following table:

Option	Description
BusinessRule	Export all business rules
NameRule	Export all naming rules
NameField	Export all naming field
HideTypeRule	Export all hide type rules of tool
ImanCompoundPropDef	Export all compound property rules of tool
ImanGRM	Export all GRM rules
TypeCannedMethod	Export all action rules

Other options using **-class** are shown in the following table:

Option	Description
FormTypeDef	Export all form type definition
ImanType	Export all instances of Teamcenter types
NoteType	Export all note types
PSViewType	Export all view types
UnitOfMeasure	Export all defined unit of measures
TaskType	Export all defined status
PSOccurrenceType	Export all occurrence types

-type

Exports all instances of a given type.

-ics_class

Exports the specified classification class, if it exists.

-imantypedef

Exports the definition of specified type.

Options and their results for **-imantypedef** are shown in the following table:

Option	Description
ListOfValues	Export list of values
ImanQuery	Export saved queries
Tool	Export tool definitions
TaskType	Export defined status
IdContext	Export identifier contexts
Status	Export defined status
StorageMedia	Export storage media
Note	Export PS occurrence note types
UnitOfMeasure	Export unit of measures defined
Occurrence	Export occurrence types
View	Export view types
RevisionRule	Export revision rules for PS configuration
Alias	Export alias and its types
Identifier	Export identifier and its types
MEWorkArea	Export workarea
MEOP	Export ME operation
ChangeTypeData	Export changeid/changetypen
Dataset	Export dataset type definition
ImanType	Export all Teamcenter types
ImanRelation	Export relations
AppearanceGroup	Export appearancegroup types

-export_bom

Specifies that the BOM is exported. This argument must be used in conjunction with the **-item** argument.

-uid

Exports the object specified by the UID.

-foldername

Exports the specified folder, if it exists.

-h

Displays help for this utility.

ENVIRONMENT

As specified in ***Unsatisfied xref title***.

FILES

As specified in *Basic concepts about Project*.

RESTRICTIONS

Siemens PLM Software recommends you do *not* use the following transfer modes with this utility:

BOMwriterExport
JTDataExportDefault
TIEPDExportDefault
TIEUnconfiguredExportDefault
PLMXMLAdminDataExport

EXAMPLES

- The following command exports item **ABC00001**, revision **A**, to a PLM XML file using the **toPrimeSupplier** transfer mode context. The default revision rule is applied.

```
plmxml_export -u=infodba -p=password -g=dba
-xml_file=abc00001_A.xml -item=ABC00001 -rev=A
-transfermode=toPrimeSupplier
```

- The following command exports item **ABC00002**, revision **A** to a PLM XML file using the **toEnterprise** transfer mode context by applying the specified revision rule and saved variant rule to apply to the BOM window:

```
plmxml_export -u=infodba -p=infodba -g=dba -xml_file=abc00002_A.xml
-item=ABC00002 -rev_rule="Latest Released" -svrule="AlphaRelease"
-transfermode=toEnterprise
```

- The following command exports all users to an XML file using the default context to export the data to PLM XML format:

```
plmxml_export -u=infodba -p=infodba -g=dba -xml_file=tcusers.xml
-class=User
```

- The following command exports an object specified by the UID and uses the default context to export the data to PLM XML format. The UID should be a unique identifier in Teamcenter:

```
plmxml_export -u=infodba -p=infodba -g=dba -xml_file=myobj.xml
-uid="QRw4LZ0g1YomJAAAAAAAAAAAAAAAA"
```

- The following command creates a PLM XML file containing all external file references associated with the top level item selected for cache prepopulation.

```
plmxml_export -u=infodba -p=infodba -g=dba -item=ITEM -rev=A  
-export_bom=yes -transfermode=justDatasetsOut -out=tickets.plmxml
```

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.

Lesson

17 *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](#)
- [Rich Client Interface Guide](#) (see *Navigating structures with queries and searches*)

Instructor Note:

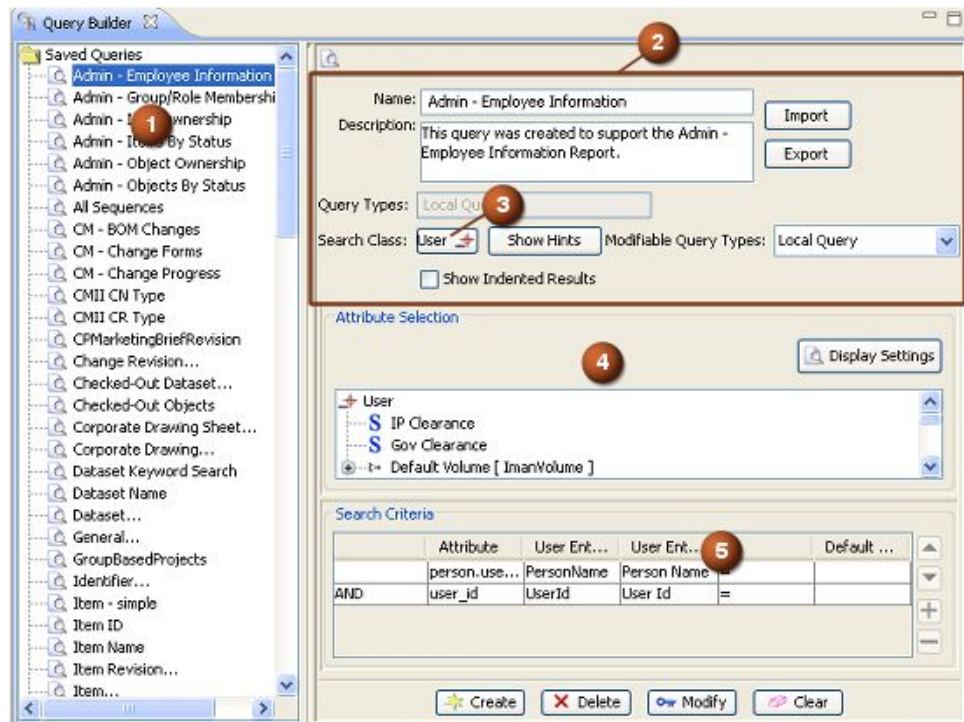
Lesson instructor notes.

Introduction to Query Builder

Query Builder allows you to create customized searches for objects in both local and remote Teamcenter databases. Building query definitions requires knowledge of the Teamcenter POM (persistent object manager) schema, which is an hierarchical arrangement of classes, subclasses, and attributes.

Query definitions, also called *saved queries*, identify search criteria used to find information in Teamcenter. Administrators define query definitions for end users. For example, you can create a saved query to find in the database all items that have been shipped.

Query Builder interface



- | | | |
|---|-------------------------------------|---|
| 1 | Saved Queries
tree pane | Displays all saved queries in the database. |
| 2 | Saved query
properties pane | Displays the name, description, query type, and search class of the saved query selected in the saved queries tree. |
| 3 | Search Class
button | Displays the Class/Attribute Selection dialog box that lists the query classes for selection. |
| 4 | Attribute
Selection pane | 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. |
| 5 | Search Criteria
pane | Defines the search criteria clauses using attributes, user entry keys, operators, and default values. Boolean operators are added for multiple search criteria clause processing. |

Basic concepts for using Query Builder

Depending on your needs, you can use an existing query from the saved queries tree. Saved queries are subject to standard object protection and are accessed by users through the search feature in My Teamcenter.

You can also use Query Builder to create queries based on the following features:

- Queries using the hints feature
- Queries that include a keyword search
- Queries using the **IS_NULL** or **IS_NOT_NULL** operators
- [Referenced-by queries](#)
- Queries based on classification attributes
- Subclass queries on a typed reference



Once created, 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.

Creating and managing queries

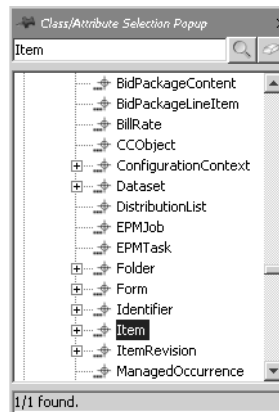
Query Builder allows you to create customized searches for objects in Teamcenter databases. When using Query Builder to create a query, you must provide the following information:

- The search class for the query
- At least one search criteria clause

Create queries

1. Click the **Clear** button  to remove existing information from the Query Builder boxes.
2. In the **Name** box, type a unique name for the query.
3. Optionally, type a description of the query in the **Description** box.
4. Select **Local Query** from the **Modifiable Query Types** list.
5. Click the **Search Class** button .


The **Class/Attribute 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 dialog box, 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.

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 the **Create** button .

The query is created, and the name appears in the saved queries tree of the Query Builder application and in the **System Defined Searches** list in My Teamcenter.

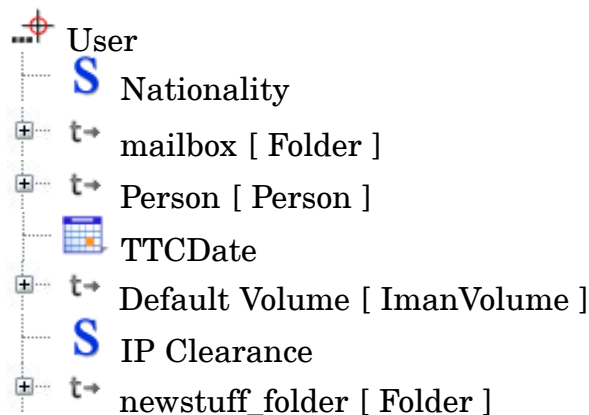
Using class attribute selections

To create query definitions, you select the class attributes to use in the search criteria. Class attributes can be found on the:

- Search class.

When a **Search Class** is selected, its attributes are displayed in the **Attribute Selection** box.

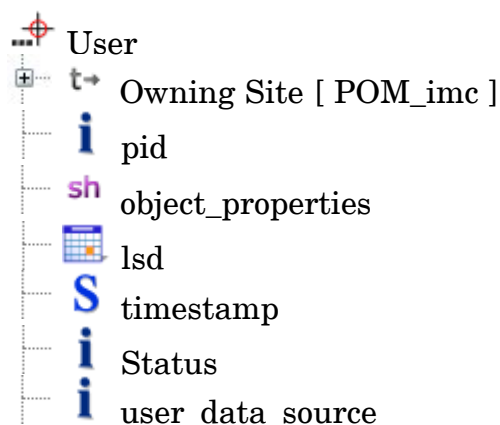
Attribute Selection box



- Parent classes.

When all attributes for the **Search Class** are displayed, inherited attributes are also displayed in the **Attribute Selection** box.

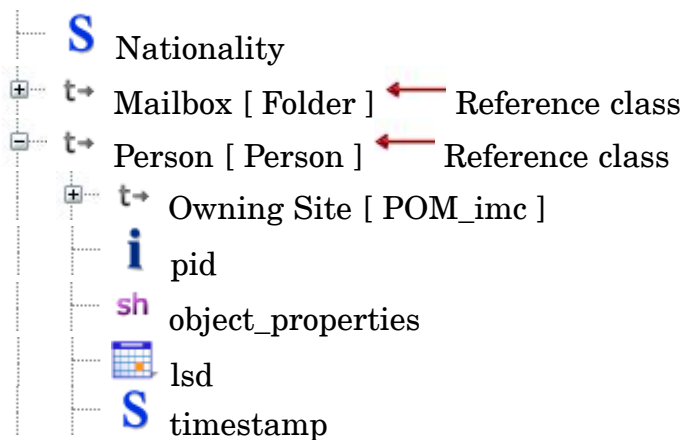
Attribute Selection box



- Reference classes.

When classes related to the **Search Class** are displayed, the reference class attributes are also displayed in the **Attribute Selection** box.

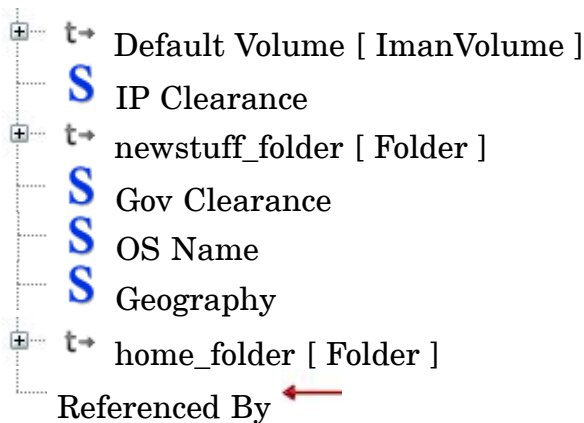
Attribute Selection box



- Referenced by classes.

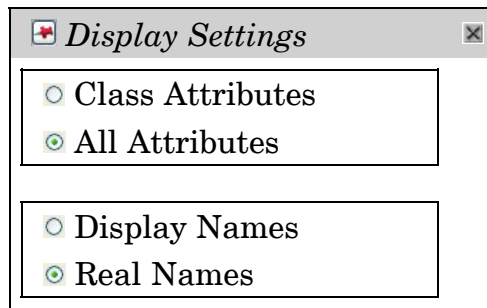
When classes related to the **Search Class** are not displayed, the **Referenced By** attribute is used to locate additional reference classes to display in the **Attribute Selection** box.

Attribute Selection box





Use the **Display Settings** box to display:

- Only attributes defined on the class or all attributes; those inherited from the parent.
- Attribute user interface display names or the attribute real database names.

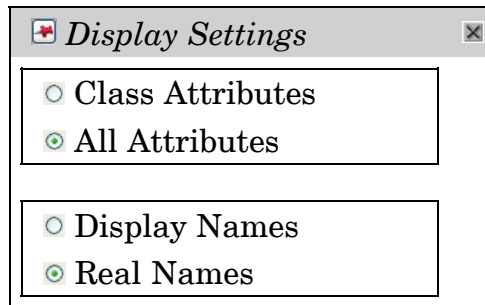



Double-click an attribute to add it to the **Search Criteria** table.

- The attributes without the plus symbol  can be directly set when searching for objects.
- The attributes with the plus symbol  refer to another class in the class structure, which may have direct attributes, or may have more attributes with plus symbols.

Add class attributes to search criteria


1. Select a class as the search class.
2. Set **Display Settings** to **All Attributes** and **Real Names**.



3. Populate the **Attribute Selection** list with the attributes required to build the search criteria. The attributes are either class attributes, parent class attributes, or reference class attributes. Reference classes are classes related to the search class. To list reference class attributes:
 - Double-click references with the plus symbol  to list more attributes or to open the **Class Selection Dialog**. Double-click a class from the **Class Selection Dialog** to add the class and its attributes to the **Attribute Selection** list.
 - Double-click **Referenced By** at the bottom of the **Attribute Selection** list to open the **Class Attribute Selection Dialog**. Double-click the **Search Class** box to search for a class. Select a reference and click **OK** to add the reference and its attributes to the **Attribute Selection** list.

Note

Navigating reference classes requires knowledge of the Teamcenter data model.

4. Double-click attributes without the plus symbol  to directly add them to the **Search Criteria** table.

Search Criteria pane

When you perform your search, Teamcenter examines the attribute specified in each of your search clauses and looks for values that match your search. Following are the search criteria elements.

Element	Description								
Boolean Rules	The Boolean rules (AND/OR) are used to combine clauses to create a custom query. When you use AND clauses together, both must be satisfied to return a match (both this clause and that clause). When you use OR clauses together, either can be satisfied to return a match (either this clause or that clause). Keyword clauses do not support the OR rule. 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 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. You must specify one of the following logical operators in each search clause. <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></table>	Logical operator	Description	=	Equal to.	!=	Not equal to.	>	Greater than.
Logical operator	Description								
=	Equal to.								
!=	Not equal to.								
>	Greater than.								

Element	Description
>=	Greater than or equal to.
<	Less than.
<=	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.

Default Value

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 **Enter** to save the default value. This box is required only when you do not specify the user entry name, unless the logical operator **IS_NULL** or **IS_NOT_NULL** is used.

The following keyword variables can be used to display default values in the query form:

- **\$USERID**
- **\$USERNAME**
- **\$GROUP**
- **\$TODAY**

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.

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. The **Search Criteria** pane defines the elements in the **Search Criteria** table.

	Attribute	User Entry L10N Key	User Entry Name		Default Value
	person.user_name	PersonName	Person Name	=	
AND	user_id	UserId	User Id	=	

The **Search Criteria** table example finds users that meet the person name and user ID criteria values entered by the user.

Double-click an attribute in the **Attribute Selection** box to add it to the **Search Criteria** table.

The combination of the **User Entry L10N Key** and **Default Value** elements dictate how the search criteria is presented to the user.

1. The **User Entry L10N Key** has a value and the **Default Value** is blank.

Result: The attribute displays in the saved query for the user to populate.

2. The **User Entry L10N Key** has a value and the **Default Value** has a value.

Result: The attribute displays in the saved query with the default value. The user can change the default value in the saved query pane.


3. The **User Entry L10N Key** is blank and the **Default Value** has a value.

Result: The attribute does not display in the saved query. The value is evaluated in the query.

Note

When the **User Entry L10N Key** is blank, the **Default Value** must have a value.

Create a new query based on an existing definition

1. Select an existing query from the saved queries tree.
2. In the **Name** box, type a unique name for the query.
3. Change the information in the **Description** box, **Search Class** box, and/or **Search Criteria** table columns.
4. Click the **Create** button .

The system adds the query to the saved queries tree. The query form is also available in the **System Defined Searches+** list in My Teamcenter.

Activities

In the *Query Builder definitions* section, do the following activities:

- insect COTS query definitions.
- Copy Item... query to create CCC_Item query.
- Create a limited access query to find Home folders.
- Test and add the queries to the Favorites search options.

Review questions

1. What are query definitions?

Select one.

- 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

6. What information is needed to create a query definition?

Select one.

- 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
7. 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
8. What do you need to specify a search criteria?
Select one.
- Selected keywords
 - Statements or clauses in the **Search Criteria** table for each attribute
 - A real name and a display name
9. You can use Boolean rules and logical operators to define the **Search Criteria**.
- True
 - False
10. What is the purpose of **IS_NULL** operator?
Select one.
- 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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Search criteria used to find information in Teamcenter.

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

2. Limit the number of query definition saved in the database.

Restrict the number of users allowed to create, modify and delete queries.

Help answer: 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.

3. True

Help answer: **Local Query** is the one that exist in local database and **Keyword Search Query** performs searches using a search engine.

Other types are: **User Query**, **Remote Query**, **User Exits Query**, and **BOM Structure Query**. These query types are created and managed using the **query_xml** utility only.

Finally, **elIntegrator Query** is a query type create through the Global Services application.

4. The **Class Attribute Selection** dialog box.

Help answer: The **Class Attribute Selection** dialog box allows you to navigate and search the Teamcenter POM schema and choose classes and attributes for use in query definitions.

You use the **Saved Queries** tree to select and review all saved queries in the database and the **Attribute Selection** table to define the search criteria clauses.

5. True

Help answer: You can select an existing query and change the name and attributes values, then click **Create** to save it as a new definition. The query is created, and the name appears in the **Saved Queries** tree of the Query Builder application and in the Find application in My Teamcenter.

6. A query name, a type, a search class, and a search criteria

Help answer:

Query definition	Example
Name	Find Home Folders
Modifiable Query Type	Local Query
Search Class	Folder
Search Criteria	Name = Home

7. True

8. Statements or clauses in the **Search Criteria** table for each attribute.

Help answer: 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).

9. True

Help answer: When you use Boolean **AND** clauses together, both must be satisfied in order to return a match. When you use **OR** clauses together, either can be satisfied in order to return a match (either this clause or that clause).

Logical operators provide 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.

10. To find objects with undefined attributes values.

Help answer: **IS_NULL** operator indicates that the reference attribute value must be blank (not set).

In programming terms, **IS_NULL** represents a special string termination character (null is ASCII character 0, not zero which is character 48) and implies that the string itself is empty or undefined.

Custom item type query definitions

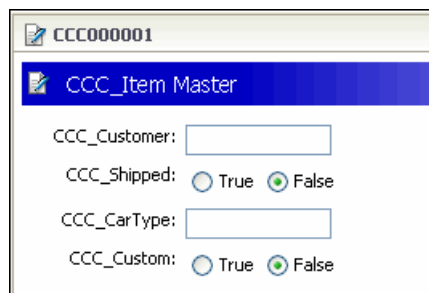
The general purpose of the query engine is to search the database for many kinds of data. It is often necessary to build queries to find items and item revisions based on the item master and item revision master form attributes and list of values.

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
	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.

Create a referenced-by query

You can create queries using clauses on a reversed-reference relationship.

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 unique name for the query.
2. Optionally, type a description of the query in the **Description** box.
3. Select **Local Query** from the **Modifiable Query Types** list.
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. Expand the **POM_application_object** class and locate the **WorkspaceObject** class.
6. 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** box.

Attribute Selection box

```

+ Dataset
  + Revision Anchor [ RevisionAnchor ]
    + [s] ref_names
  + Tool Used [ Tool ]
    + S Classification
    + [i] ref_types
    + S Format
    + [v] References
  
```

7. Double-click the **Referenced By** node in the **Attribute Selection** pane.

The **Class Attribute Selection** dialog box appears.

Note

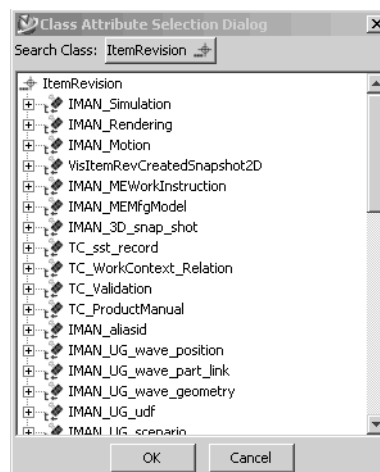
If you select the **Referenced By** attribute, the **Class Attribute Selection** dialog box appears. Use this dialog to select the class and through what attribute the given object is referenced in that class. You can add selected attributes to the search criteria table.

8. 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.

9. Expand the **POM_application_object** class and locate the **WorkspaceObject** class.
10. Expand the **WorkspaceObject** class and select the **ItemRevision** class by highlighting it and closing the dialog box.

The referencing class and its attributes are displayed in the **Class Attribute Selection** dialog box.

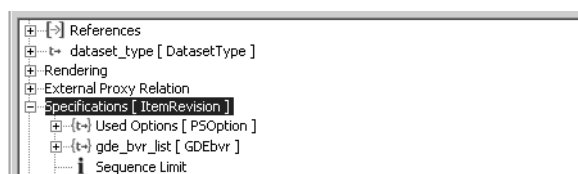


Note

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.


11. Select a referencing attribute, in this case **IMAN_specification**, by double-clicking the node.

The referencing attribute, **Specifications**, and class, **ItemRevision**, are displayed in the **Attribute Selection** area.



12. Select the attributes of the referencing class on which you want to build query clauses. In this case, locate and double-click the **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 ← symbol indicates a reversed-reference relationship.

13. Type a key name in the **User Entry Key** box. This makes the **User Entry Key** unique when you perform a query from the My Teamcenter search pane.
14. Click the **Create** button  to create the query.

The system adds the query to the saved queries tree. The query form is also available in the **System Defined Searches** list in My Teamcenter.

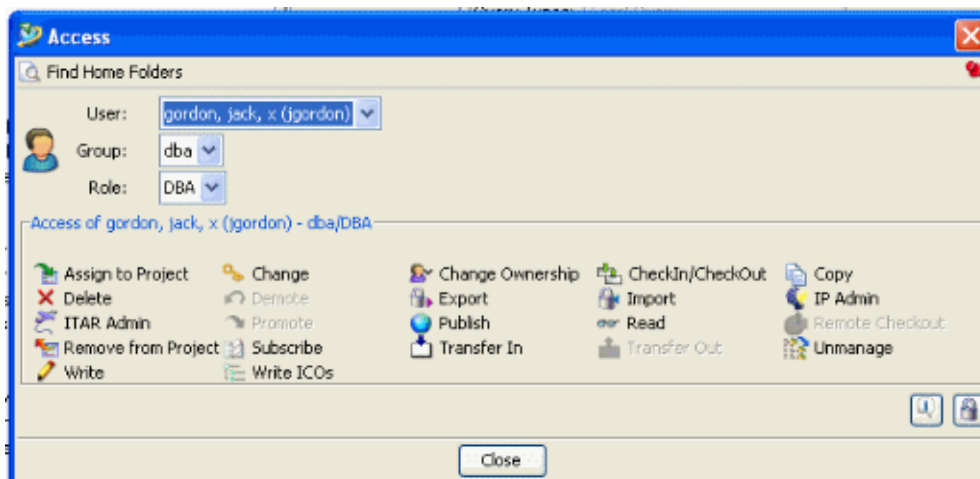
Limit access to query definitions

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.

Importing and exporting 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.

- 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. This results in errors when you attempt to create the query definition using the imported data.

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.

Export query definitions


Perform the following steps to export a query definition to an XML file:

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.

Activities

In the *Query Builder definitions* section, do the following activities:

- Create a query for all CCC items.
- Create a query with export, modify, and import.
- Import is_rejected and user_data3 query definitions.
- Create a query that uses form properties.
- Create a ItemRevision query using a relation to a form.
- Create a Item query using a relation to a form.
- Create a query from item to components.
- Create a query from a component to item revisions.
- Create a query definition using a relation to a form.
- Create a referenced-by query.
- Create a where-referenced from Hints.
- Create a query that finds Home folders in a Teamcenter site.

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.
 - 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
4. You can create queries using clauses on a reversed-reference relationship.
 - True
 - False
5. What is a referencing attribute?
Select one.
 - Any object attribute
 - An attribute that may reference the objects of the class being queried
 - An attribute with an **IS_NULL** value.
6. What is the **Show Indented Results** option used for?
Select one.
 - To display the results in an expanded tree format
 - To display object attributes
 - To save the search history
7. Saved queries are not subject to standard object protection.

- True
 - False
8. 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**
9. Query definitions can be shared with other Teamcenter sites.
- True
 - False
10. How are the query definitions exported?
Select one.
- Using XML file format
 - Creating a compressed file
 - It is not possible to export query definitions from Teamcenter
11. When importing query definitions, the XML files are parsed and verified before the data is imported.
- True
 - False

Instructor Note:

Activity instructor notes.

Answers to review questions

1. False

Help answer: The general purpose of the query engine is to search the database for *many* kinds of data.

2. Query forms are dialog boxes that provide user access to saved database searches.

Help answer: This is tested in the activity.

3. True

Help answer: This is tested in the activity.

4. True

Help answer: You can create queries using clauses on a reversed-reference relationship. For example, create a query to find dataset objects that are referenced, through an **IMAN_specification** relationship, by an item revision with a specific name.

5. An attribute that may reference the objects of the class being queried

Help answer: Only those attributes that may reference the objects of the class being queried. In the example, only the *reference by* **Dataset** class attributes are displayed in the dialog box.

6. To display the results in an expanded tree format


Help answer: 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.

7. False

Help answer: Saved queries are subject to standard object protection and are accessed by users through the search feature in My Teamcenter.

8. Adding accessors to grant or revoke access.

Create an **Object Named ACL**.

Help answer: You use the **Access Control List**  to modify the saved query access to add custom accessors to add or revoke privileges for groups and users.

9. True.

Help answer: Query definitions can be exported and saved as XML files that can be shared with other Teamcenter sites.

10. Using XML file format.

Help answer: Query definitions can be exported and saved as XML files that can be shared with other Teamcenter sites.

11. True.

Help answer: Yes, the XML files are parsed and verified before the data is imported.

The **Verify** button is used to validate that the POM class matches existing classes in the database before importing.

Summary

The following topics were taught in this lesson:

- Define saved queries.
- Create saved queries.
- Limit access to saved queries.
- Import and export query definitions.

Lesson

18 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:

- Identify the different types of report definitions.
- Create and manage summary and item report definitions.
- Import and export report definitions throughout the enterprise.

Help topics

Additional information for this lesson can be found in:

- [*Report Builder Guide*](#)
- [*PLM XML Export Import Administration*](#)

Instructor Note:

Creating report definitions is an interactive process where the administrator first creates a report, executes the report, validates the results, and then publishes the report to the users. To use Report Builder, the administrator must understand the data model, PLM XML translation, and style sheet basics.

Introduction to Report Builder

Report Builder allows you to create and manage your own report definitions. These report definitions provide end users with reports to run in the rich client or the thin client.

End users of the rich client use Report Builder to create new reports to meet their organization's needs. Use Report Builder to create the following report definitions:

- **Summary**

Reports that summarize similar information, for example, reports that show all the employees, the items belonging to a user, or the release status of items.

- **Item**

Reports that can be run on a particular item, for example, reports that show the BOM list for an item or the workflow signoff for an item.

- **Custom**

Reports that use custom processing.

Report Builder works with the Query Builder application and the PLM XML Export Import Administration application. To assemble reports, you can use saved queries created in Query Builder and property sets created in PLM XML Export Import Administration.

Before you begin

Prerequisites

To use Report Builder, you should know:

- Query Builder

Query Builder is a Teamcenter application that enables you to create and maintain customized searches for objects in the Teamcenter databases.

- PLM XML Export Import Administration

You use PLM XML Export Import Administration to create transfer mode objects that contain rules to configure import or export operations.

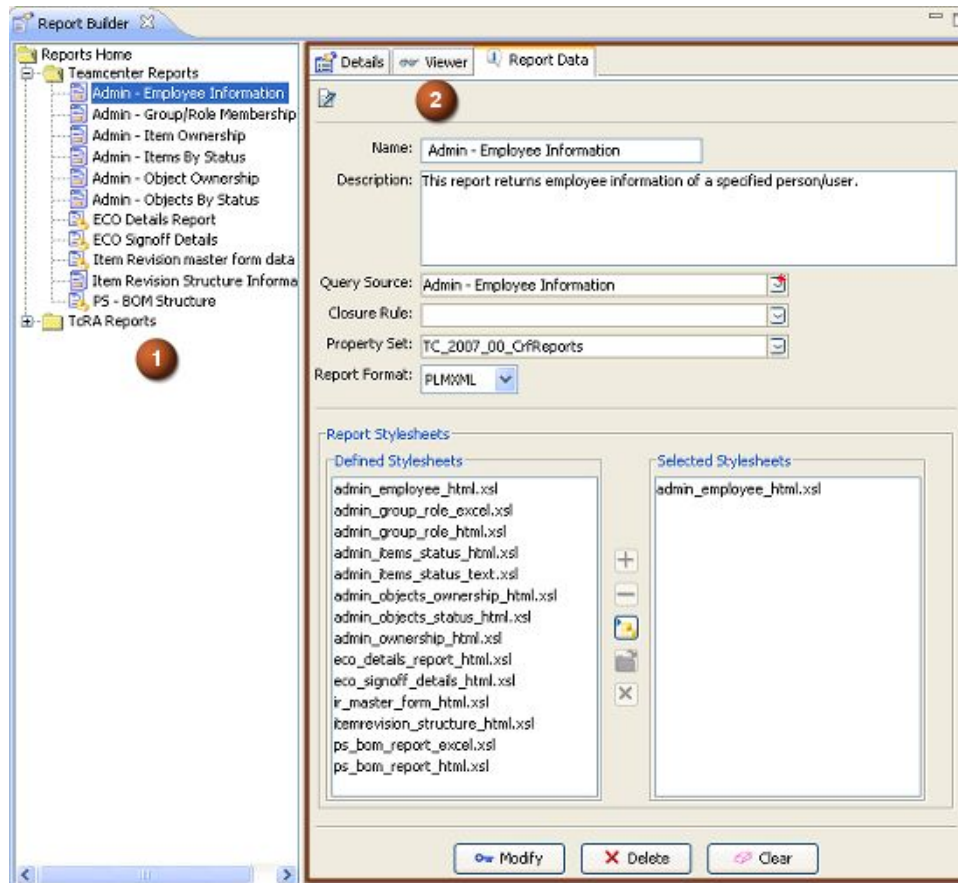
- PLM XML and TcPLMXML schema

You must understand the two schemas that Report Builder uses. PLM XML is an open schema based on standard W3C XML schemas. TcPLMXML schema is a Siemens PLM Software format that uses Teamcenter XML. Understanding these schema allows you to work with the XML-formatted organization of the data for import and export to Teamcenter.

- Teamcenter data model

You must understand the Teamcenter data model, including the overall data that is stored. For example, some of the data stored are items, item revisions, and forms, along with the constructs holding the data.

Report Builder interface



- 1 **Reports Home tree** The **Reports Home** tree pane displays the available report definitions. The **Teamcenter Reports** folder contains report definitions created using Report Builder. The **TcRA Reports** folder contains report definitions created using Teamcenter's reporting and analytics.
- 2 **Report Builder tabs** The Report Builder tabs display file details, properties, and configurable data for the selected report.

Note

For information about general interface topics such as the navigation pane, tabs, general menu commands, data display, data objects, favorites, the clipboard, and printing object properties, see *Getting Started with Teamcenter*. For information about user interface components that are specific to My Teamcenter in the rich client, see the *Rich Client Interface Guide*.

Basic concepts for using Report Builder

Report Builder lets administrators create and manage Teamcenter report definitions. After administrators create the report definitions, users can use these definitions to generate reports from My Teamcenter in the rich client or from the thin client.

Administrators can create several different kinds of reports:

- Summary reports present information based on a saved query definition.
- Item reports are run on one or more selected items based on a class of items.
- Custom reports show results based on custom processing.

All report definitions have a similar structure and contain a source for the data (such as a query) and a style sheet to format the output.

Report Builder definition types

Report Builder allows you to create three types of reports:

- 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 one or more objects, such as item revisions. 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 an object to be selected and summary reports depend on a saved query. 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.

- 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 you select a custom report 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.

Common structure in all reports include:

- A unique report ID that can be assigned automatically by the system.
- A report name and description.
- A style sheet dataset to reformat the report data for output.

Report properties	Description	Report type		
		Summary	Item	Custom
Query source	Specifies the saved query used to find items to report.	✓		
Closure rule	Specifies structure processing for related items.	✓		
Report format	PLM XML or TcPLMXML report output properties.	✓	✓	
Property set	Specifies additional output properties.	✓	✓	
Class	Specifies the type of items allowed by the report.		✓	
Transfer mode	Specifies structure processing and filtering for related items.		✓	
Process	Specifies the program for the custom process path.			✓
Output	Specifies the output file name for the report.			✓
Parameters	Specifies variable input for the custom process.			✓

The end user has the option to attach different style sheets to produce different types of reports in HTML or Microsoft Excel format. For examples of how to format summary and item reports, refer to the Report Builder sample style sheets. The sample style sheets are available on the corporate server in the *tc_data/crf/Resources* directory.

Closure rules, transfer modes, and property sets are defined using PLM XML

Standard Teamcenter report definitions

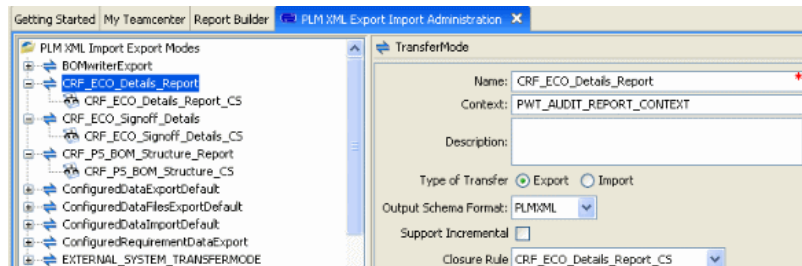
The report definitions shown in the following table are delivered with the standard Teamcenter installation.

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.

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.

Primary Object Class Type	Primary Object	Relation Type	Related Property Or Object	Property Action
CLASS	ItemRevision	PROPERTY	last_mod_date	DO
CLASS	ItemRevision	PROPERTY	object_string	DO

- 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.

Importing and exporting report definitions

The utility allows report definitions, their dependent data (for example, saved query definitions and property set definitions), and their associated style sheets to be exported from one Teamcenter server and imported to another.

```
import_export_reports {-import | -export | -execute}  
[-u=user-id -p=password -g=group]  
-stageDir=directory -reportId=report-identifier -f=output-filename.xml  
[-h]
```

Key arguments for the **import_export_reports** utility:

Argument	Definition
import	Specifies the report definition is imported.
export	Specifies the report definition is exported.
stageDir	Specifies the fully qualified name of the directory that contains all of the report definitions and its associated data in predefined format. This directory must exist prior to import and export.
reportId	Specifies the ID of the report definition. On export, this is the name of the directory that is created where the report definition and style sheets are written. On import, this is the name of the directory where the report definition and style sheets are located.

Activities

In the *Report Builder definitions* section, do the following activities:

- Create a summary report definition.
- Attach a style sheet.
- Define a closure rule and a transfer mode.
- Create an item report definition.
- Attach a style sheet.
- Import a report definition.
- Export a report definition.

Review questions

1. What are report definitions?
Select one.
 - 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.
 - Custom reports
 - Item reports
 - Summary reports
4. What do you need to create an item report definition?
Select one.
 - A custom application
 - A property set and a saved query
 - A report ID, a report name, a class, and a transfer mode
5. The Report Builder application leverages on the PLM XML schema to produce report data.
Select one answer.
 - True
 - False
6. 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
7. 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
8. What do you control with property sets?
- Select one.
- Query sources
 - Transfer mode objects
 - **UserData** elements
9. What do you specify with the **SKIP** action?
- Select one.
- That the action is executed
 - That the first action is not executed, but then can be processed
 - That the relationship should not be followed
10. What do you use to deploy report definitions to other Teamcenter sites?
- Select one.
- The **import_export_reports** utility
 - The **File® Export** menu in Report Builder
 - The PLM XML Export Import Administration window
11. When you export and import report definitions, the saved queries and associated style sheets are also processed.
- True
 - False

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Report specification templates

Help answer: Report definitions are report specification templates that contain the name, description, query sources, query description, property sets, and style sheets to output report information.

2. True

Help answer: Summary reports are generated from saved or dynamic 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.

3. Item reports

Help answer: Item reports require a context and summary reports are context free.

4. A report ID, a report name, a class, and a transfer mode

Help answer: 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, and a transfer mode object.

5. True

Help answer: The Report Builder application leverages on the PLM XML schema to produce report data. You use the PLM XML Export Import Administration to create transfer mode objects to define the report data export context, closure rules to define the scope of the report translation, and to define the report property set that adds the Teamcenter data needed in PLM XML format if the data does not exist.

6. To import and export objects or system data

To contain the rules that configure the export operation

Help answer: Transfer mode objects allow you to import and export objects or system data. Transfer mode objects 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.

7. closure rules

Help answer: Closure rules specify how to process the data structure to obtain the information. Closure rules are defined as a set of rules that are evaluated in order for each target object.

8. **UserData** elements

Help answer: You use property sets to control 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 operation.

9. That the relationship should not be followed

Help answer: Use the **SKIP** to specify that the relationship should not be followed. This directive is useful to eliminate special cases before a more general rule is reached, but the **SKIP** action cannot be combined with any other action.

10. The **import_export_reports** utility

Help answer: Once reports are defined, you can export and deploy the reports to other Teamcenter sites using the **import_export_reports** command line utility.

11. False

Help answer: The **import_export_reports** exports and imports report definitions in files that include associated style sheets. Saved queries must be imported and exported separately.

Summary

The following topics were taught in this lesson:

- Different types of report definitions.
- Create summary and item report definitions.
- Define PLM XML report data.
- Import and export report definitions.

Lesson

19 *Using item revision configuration*

You can create and apply revision rules to select the appropriate revision of components in the product structure. A revision rule sets the criteria for selecting the revision, for example, whether to load working revisions or determining the released revisions that are loaded. This allows you to configure a structure as it was or will be on a particular date, for example, by utilizing effectivity data on each item revision (release status). For additional information, see [Understanding revision rules](#).

Users can do the following:

- Create and set date and optional end item revision effectivity.
- Create and set unit number and end item revision effectivity.
- Share an existing effectivity between structures.
- Create and share a named effectivity.
- Protect an effectivity so that only certain users can apply it.

Both unit number and date ranges may be closed or open-ended. For open-ended ranges, **UP** and **SO** (stock out) values may be specified. Ranges may be discontinuous, for example, unit numbers **1, 5, 10–UP**.

A privileged user (typically, the system administrator) creates the revision rules that are then made available to other users. Consequently, nontechnical users do not have to understand the full complexity of revision configuration, but simply apply the appropriate revision rule.

Before creating or implementing item revision configuration, you should read *Getting Started with Product Structure*, which includes additional information.

Understanding revision rules

Item revision configuration allows you to create and apply revision rules that select the appropriate revision of parts and assemblies in the product structure. A revision rule:

- Selects working revisions and (optionally) specifies the owning user or group.
- Selects revisions by status (according to status precedence) or the latest revision with any status using release date.
- Optionally, specifies the effectivity against which the revisions are configured. Effectivity may be specified by date or unit number; the unit number may optionally be qualified by end item.

Note

Do not confuse the effectivity in the revision rule with the actual effectivity data visible on item revisions, as described in [Displaying and editing revision effectivity](#).

- Select revisions in a specified override folder.
- Select the latest revisions according to the revision ID by alphanumeric, numeric order, or creation date order. This selection does not depend on whether revisions are working or released.

You define each of the above criteria with a revision rule entry. A revision rule may contain any number of rule entries, each of which attempts to select a revision according to the specified criteria, for example, the status that the revision should have or the user or group that owns the revision.

Teamcenter evaluates rule entries in order of precedence until a revision is successfully configured. You can include some entries more than once to define an order of precedence, for example, status, as shown:

```
Working (Owning Group = Project Y)
Has Status (Production, Effective Date)
Has Status (Pre-Production, Effective Date)
```

You can modify the order of the rule entries to change the precedence Teamcenter uses when evaluating the revision rule. Certain rule entries can also be grouped so they are evaluated with equal precedence. For more information, see [Defining a Working entry](#).

Teamcenter always enforces a revision rule when you open a structure in a Structure Manager window.

Elements of a revision configured product structure

The following elements may appear in a product structure that is controlled by revision configuration.

Element	Purpose
End item	A product, system or module with respect to which you can configure the structure by effectivity. For example, you can configure the structure of unit number 110 in product X400 , where X400 is the end item.
Imprecise assembly	A single-level assembly that has items (<i>not</i> item revisions) as the components. Teamcenter determines the applicable revision from the revision rule settings.
Override list	A mechanism that allows a user to override the revision that would normally be loaded by the revision rule. The user places item revisions in a workspace folder, and the revision rule is overridden by the rule specified in the override list.
Precise assembly	A single-level assembly that has specific item revisions as the components. When Teamcenter applies the revision rule, the precisely specified item revision is configured by a precise entry in a revision rule.
Release status	An object assigned to an item revision after it is successfully released. Item revisions can be configured according to their status. The status may optionally contain effectivity data for revision configuration (<i>not</i> occurrence effectivity configuration).
Revision rule	The parameters set by a user that determine which revision of an item at a particular time. You can also save a revision rule as a workspace object.
Rule entry	A revision rule comprises an ordered list of rule entries. Each type of rule entry is concerned with a particular type of configuration.
Snapshot	You can save a configured product structure as a <i>snapshot</i> . The snapshot folder contains all revisions of the structure, and you can use it to redisplay a saved structure at any time.

Element	Purpose
Baseline	A dataset with a copy of the currently configured structure attached to it. Baselining configures a completely released structure and thereby guarantees that the models are always the same as when the baseline was created.
Working revision	An unreleased version of the structure. Any user with write privileges can freely change this revision. Teamcenter maintains no record of intermediate states of a working revision.

Configuring privileged and unprivileged users

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

Unprivileged users have access to the following commands:

- **View/Set Revision Rules**
- **Set Date, Set Unit, Set Override**, where appropriate

Privileged users have access to the following commands:

- **Modify Current Rule**

Modifications may be applied to the current structure but only saved if the user has write access to the original rule.

- **Create/Edit Revision Rules**

The user may save new or changed revision rules.

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

For more details, see ***Unsatisfied xref title***.

REVIEW NOTE

Issue: This was added. Where should it go?

This rule provides parameters to locate the assemblies in which the item or item revision is used. Revision rules are made up of sequential lists of entries, each of which is evaluated to obtain a configured revision of the item. For example, if you select the Latest Working rule, only the latest working revision of the assembly is retrieved. Released assemblies and earlier versions of the assembly are not retrieved when this rule is selected.

Using revision rules from My Teamcenter

You can create revision rules and save them as workspace objects. These workspace objects can be referenced by folders in other applications, for example, My Teamcenter. You can copy a revision rule into My Teamcenter. You can also search in My Teamcenter to locate a particular rule. You can then reference a rule within a particular item or revision for configuration

purposes. You can view the contents of a revision rule in My Teamcenter by clicking the **Viewer** tab. This displays the list of entries in the rule.

Creating a revision rule

If you are a privileged user, you can use two dialog boxes to create or edit a revision rule, the **Modify Current Rule** dialog box, and the **Create/Edit Rules** dialog box. Both dialog boxes contain the Revision Rule Editor, which comprises two main panes.

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

Create a rule entry

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

Syntax for revision rule entries is given in [Defining revision rule entries](#).

Modify a rule entry

1. Select an entry in the list of entries.
2. Click **Copy**. Teamcenter copies the entry into the editing area and displays the appropriate entry type and boxes.
3. Edit the values of the entry and click **Replace** with the original entry still selected. Teamcenter replaces the old entry with the new entry.

Syntax for revision rule entries is given in [Defining revision rule entries](#).

Delete a rule entry

1. Select an entry in the list of entries.
2. Click **Remove** and Teamcenter removes it from the rule.

Edit the entries within a rule

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

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

Modify the current revision rule

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

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

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

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

Syntax for revision rule entries is given in [Defining revision rule entries](#).

Defining revision rule entries

A revision rule is made up of a sequential list of entries. Evaluation of the rule involves evaluating each of the entries, in order, until a configured revision of the item is successfully obtained. A rule may be made up of entries of the following types.

Defining a Working entry

Use a *working entry* to select working item revisions; that is, those items revisions without any release status. By default, Teamcenter selects the latest working revision of the item according to the date it was created. You can select a more specific revision with one of the following settings:

- Owning user

If you specify an owning user within a working entry, Teamcenter configures the latest revision owned by the specified user, if there is such a revision. You can also set the owning user to **Current**, and Teamcenter configures the latest revision owned by the current user.

- Owning group

If you specify an owning group within a working entry, Teamcenter configures the latest revision owned by the specified group, if there is such a revision. You can also set the owning group to **Current**, and Teamcenter configures the latest revision owned by the current user's group.

Tip

There may be more than one working entry within a revision rule. For example, a rule may configure the current user's working revision and, if none is found, configure the current group's working revision instead. If a user changes group, it is necessary to reapply the revision rule to configure the appropriate revisions for the new group.

However, in many circumstances, it is good practice to limit the number of working revisions, typically to a single revision. Do this in Business Modeler IDE, as follows:

1. Right-click the item revision business object on which you want to limit the working revisions and choose **Open**.
2. Click the **Operations** tab, open the **Operations** folder, scroll down to the bottom of the **Operations** folder, and open the **Legacy Operations** folder.
3. Select the **ITEM_copy_rev** operation.
4. In the **Pre-Condition** pane, choose **Add** and select **checkLatestReleased**.
5. Add an argument that specifies the number of working revisions allowed.
6. Save and deploy the modified configuration.

For more information, see the *Business Modeler IDE Guide*

Example of a Working entry

An example of **Working** entries in a revision rule follows:

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

Assume you configure the following two items with this rule:

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

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

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

Note

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

Example of a Working entry with current user/group

An example of a current user/group entry in a revision rule follows:

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

Assume you configure the following item with this rule:

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

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

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

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

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

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

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

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

Defining a Status entry

Use a status entry to select item revisions that are released with a particular status. The following settings are available for status entries:

- Any release status

Teamcenter configures the latest item revision with a released status, regardless of the actual status.

- Selected status

Teamcenter configures the latest item revision with a selected status type. This setting allows you to configure a structure that contains only item revisions with a specified status.

For details about configuring release statuses for your site, see the *Business Modeler IDE Guide*.

- Released date

Teamcenter selects the latest item revision according to the date the revision was released (that is, the date the particular status was added).

- Effective date

Teamcenter selects the latest item revision according to effectivity dates defined on the release status. Privileged users define effective dates as described in [Displaying and editing revision effectivity](#).

- Effective unit number

Teamcenter selects the latest item revision according to unit numbers defined on the release status.

Example of status hierarchy in a revision rule

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

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

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

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

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

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

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

Note

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

Example of effective date configuration in a revision rule

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

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

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

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

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

Defining a Latest entry

You can specify a *latest* entry to select the latest item revisions regardless of whether they are released. For this entry, Teamcenter does not differentiate between working revisions and revisions with status. The following settings are available for **Latest** entries:

- Creation date

Teamcenter selects the latest item revision according to the date the revision was created.

- Alphanumeric revision ID

Teamcenter selects the latest item revision in alphanumeric order by revision ID. It selects any numeric revision IDs in numeric order by their first digit, for example, 1, 10, 2, 21, and so on. It selects any alphabetic revision IDs in alphabetic order, for example, a, aa, b, and z.

- Numeric revision ID

Teamcenter selects the latest item revision in numeric order by revision ID. It does not configure revisions with nonnumeric IDs.

- Alpha and number revision ID

Teamcenter selects the latest item revision first in order of any initial alphabetic characters, then any following numeric digits, for example, a, a2, a5, b2, b23, aa12, aa123, bc22, aaa0 and aab.

Example of a revision rule with latest by creation date entry

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

```
Latest( Creation Date )
```

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

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

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

Defining a date entry

Use a date entry to specify a date to configure the structure against. You can only use this entry type with other entries. These types of entry find the latest entry before the specified date:

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

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

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

Note

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

You can qualify date effectivity with an [end item entry](#).

Defining a unit number entry

A unit number entry specifies a unit number to match when configuring item revisions with status using unit number effectivity. You can only use this entry type with other entries. If no unit number entry is present in a revision rule, Teamcenter configures all status entries that configure by effective unit number.

Typically, a unit number is a property of the end product or a major module of a product. As Teamcenter may manage many units, you typically qualify a unit number entry with an [end item entry](#).

Defining an end item entry

Use an end item entry to qualify the unit number or the effective date specified in the revision rule.

You must use the end item entry with a unit number entry. The unit number you specify refers to a unit or serial number of the item identified by the end item entry. This item is typically the end product, a major module or subsystem of a product, if it has its own unit number serialization.

You can also use an end item to qualify date effectivity, so you can make an item revision effective in different products at different times.

If you put an end item entry in a rule, you force the application of the end item in all uses of the rule. If you omit the end item entry, users can specify an end item to configure the structure at runtime by choosing **Tools→Revision Rule→Set Date/Unit/End Item**. You can also add a button to the toolbar to implement the **Set Date/Unit/End Item** menu command.

Teamcenter interprets revisions with date effectivity and no qualifying end item as effective for all possible end items. Hence, when Teamcenter processes a rule with date effectivity and an end item entry, it considers only revisions having date effectivity with no end item, and those having date effectivity qualified by the end item specified in the rule. Only those revisions with date effectivity qualified by a different end item are excluded. If the rule does not specify an end item, Teamcenter only considers revisions with date effectivity and no end item qualification.

Defining grouped entries

You can group status entries and working entries for equal precedence. In the case of status entries, you can group two or more different statuses with equal precedence. In the case of working entries, you can group different owners with equal precedence. You can also group entries according to item type.

Example of revision rule with grouped entries

An example of a revision rule that uses grouped entries to obtain equal precedence of status types follows:

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

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

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

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

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

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

For details about grouping revision rules by item type, see [Grouping revision rule entries by item type](#).

Grouping revision rule entries by item type

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

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

Creating Has Item Type entries

Create a **Has Item Type** entry in a revision rule in the following format:

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

  <Entry 1>

  <Entry 2>
  etc.}
```

For example:

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

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

Grouping entries for multiple item types

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

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

Grouping combinations of item type entries

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

- Single revision rule entry grouped for item type:

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

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

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

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

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

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

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

Create and group Has Item Type revision rule entries

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

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

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

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

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

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

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

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

Note

You should only group **Working** entries with other **Working** entries and **Status** entries with other **Status** entries. Do not group other combinations of entries.

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

Group existing revision rule entries by item type

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

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

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

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

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

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

Note

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

Ungroup combinations of item type entries

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

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

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

Teamcenter removes the selected entries and updates the dialog box.

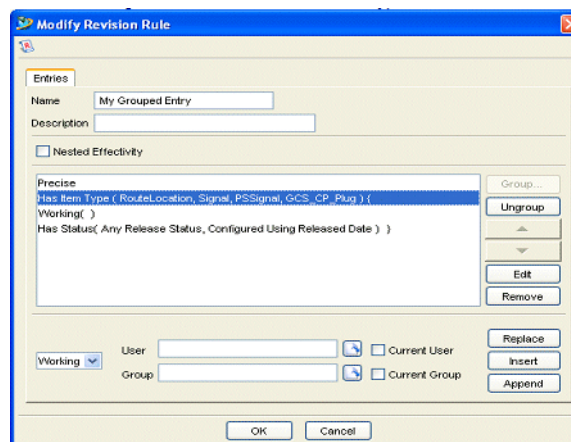
Note

To remove an equal precedence entry from a revision rule, you must highlight *all* the lines of the entry in the dialog box.

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

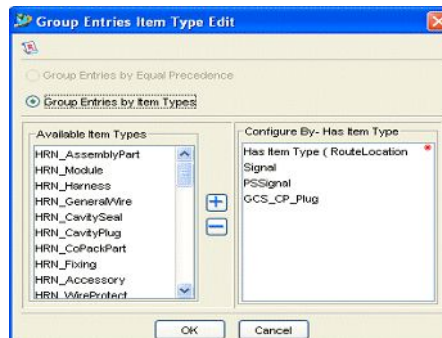
1. Choose **Tools**→**Revision Rules**→**Create/Edit**.

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



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

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




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

Note

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

Applying a revision rule to a Structure Manager window

Each Structure Manager window has an associated revision rule that Teamcenter applies to the displayed structure. The name of the revision rule appears in the banner at the top of the window. Setting a revision rule in one window does not affect the revision rule that is applied in another window. When you open a new Structure Manager window, Teamcenter applies your default revision rule to the structure.

If you do not want to use the default rule, you can [set a revision rule](#) for your current Structure Manager session by clicking the **Revision Rule** button  on the toolbar or choosing **Tools**→**Revision Rule**→**View/Set Current**.

A revision rule is always in force when a Structure Manager window is open. The revision rule allows Teamcenter to determine the specific revision of each part and assembly to configure in the structure. You set your default revision rule by choosing the **Edit**→**Preferences** menu command.

Note

Unlike a revision rule, a variant rule is optional and no default is applied.

Set a revision rule

- Set a revision rule by clicking the **Revision Rule** button on the toolbar or choosing **Tools→Revision Rule→View/Set Current**.

Teamcenter displays the **Set Revision Rule** dialog box containing a list of revision rules that were configured by a privileged user. The right side of the dialog box shows the details of the selected revision rule. To set a revision rule, you select the required rule and click **OK** or **Apply**. If you are a nonprivileged user, you do not need to understand the complexity of the rule syntax, but should understand the appropriate rule to apply in a particular circumstance.

Tip

You can save a default revision rule by choosing the **Edit→Options** menu command.

Set date/unit/end item

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

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

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

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


Tip

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

Set an override folder

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

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

- A **Name** box into which you can enter a case sensitive search string to find the required override folder. Click the **Find**  button next to this box to perform the search.
- The search results box. Double-click the folder you want to set.
- A **Folder** box that displays the folder currently used as the override folder in the revision rule.

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

Note

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

Viewing revision rule information

Use the **Rule Configured By** column in the product structure window to interpret how a particular line is configured. The values that may appear are:

- **Working (Owning User = *user*, Owning Group = *group*)**

This appears if the item revision is working and thus configured by a **Working** entry in the revision rule. The owning user and owning group arguments only appear if they are specified in the revision rule.

- **Has Status (*status*, Configured Using <... >)**

Shows the status specified in the revision rule that configured the item revision. If **Any Release Status** is specified, **Any Release Status** is shown. The **Release Status** column shows the status of the configured item revision.

The method of configuration (**Configured Using**) is also displayed and may be **Released Date**, **Effective Date**, or **Effective Unit No.**

Additionally, this column shows the date or unit number used in the configuration, as follows:

- **Date (Today) or Date (1-Jan-2007)**

The date specified in the revision rule. Used to configure historical revisions with status.

- **Unit No.(4)**

The unit number specified in the revision rule.

- **End Item (*item object id*)**

Identifies the end item specified in the revision rule.

- **Override Folder (*folder*)**

The override list folder name is shown if the override list caused the item revision to be configured.

- **Precise**

This appears if the occurrence is precise and is configured as such.

- **Latest (...)**

This appears if the item revision is configured by a **Latest** entry in a revision rule. The method of latest configuration can be **Creation Date**, **Rev ID Numeric**, or **Rev ID Alphanumeric**.

- **Not Found**

This is shown if no revision meets the revision rule criteria. Teamcenter displays **???** to represent the revision.

Displaying and editing revision effectivity

By default, revision effectivity data is not displayed in the Structure Manager properties table. Before editing revision effectivity, you must first display the data.

If you are working with revision effectivity, you may want to display the **Revision Effectivity** property, as described in ***Unsatisfied xref title***.

Display revision effectivity data

1. Select the line whose item revision effectivity data (ranges of dates or unit numbers for which the revision is effective) you want to view.
2. Choose **Tools**→**Effectivity**→**Revision Effectivity**.

Teamcenter displays the **Revision Effectivity** dialog box. This dialog box lists all the effectivity ranges for each status attached to the item revision.

Note

While you can set date effectivity without an end item, unit effectivity must always have an end item.

Edit revision effectivity data

Note

For easier access to editing functions, consider adding an **Edit Revision Effectivity** button to the toolbar by right-clicking the toolbar, choosing **Customize**, and selecting the required button.

If you previously released two or more items or item revisions simultaneously, when you edit the effectivity of one such item or item revision, you also edit the effectivity of any associated items or item revisions. You can identify any such affected items by sending the structure line to the Workflow Viewer to see the other targets of the process that share the same release status and whose effectivity will also change.

1. In the **Revision Effectivity** dialog box, select the appropriate revision effectivity line and click **Edit**, or double-click the line.

Teamcenter displays the **Edit Revision Effectivity** dialog box.

Note

You can also edit revision effectivity in My Teamcenter. Double-click the item status and change the displayed value.

2. In the **Create or Edit Effectivity** dialog box, choose **Units** or **Dates** effectivity, as appropriate, and define the effectivity data.

If defining unit effectivity, type the desired effectivity range in the **Units** box. Use the - character within a continuous range, and the , character to separate discontinuous ranges. For example, the unit range **1-5,7-9** defines effectivity for units 1 through 5, and 7 through 9 (but not effective for unit 6).




If defining date effectivity, select a cell in the **From** or **To** column, select a date from the calendar (and optionally enter a time), and click **Set Date** to place that date in the selected cell. Click the **Clear Date** button to remove the date from the currently selected cell. Repeat these steps for additional date ranges until you have entered all the desired date ranges.

- Click the **UP** button to add the **and up** (open-ended effectivity) condition to the end of the unit or date effectivity range.
- Click the **SO** button to add the **stock out** condition to the end of the unit or date effectivity range.
- If you use effectivity mapping, select the **Shared Effectivity** check box and see [Managing nested effectivity](#) for further information.


- Select the **Apply Access Manager effectivity protection** check box to apply the predefined Access Manager rules to this effectivity.

Note

Teamcenter interprets **UP** and **SO** conditions as open-ended for revision configuration purposes. The revision is considered effective for any value greater than or equal to the unit or date value immediately preceding the **UP** or **SO**. *Stock out* indicates that existing stocks of a component revision should be used up before the next revision.

3. Optionally, for date effectivity, use the **Edit Revision Effectivity** dialog box to define an end item to qualify the effectivity range. You *must* use this with unit effectivity to specify a product, module, or subsystem that carries the unit number to which this effectivity refers. You can select an end item in one of the following ways:
 - Clicking **Find**  adjacent to the **End Item** box and searching for an item by identifier and/or name.
 - Copying an item to the clipboard before opening the **Revision Effectivity** dialog box and clicking **Paste**  adjacent to the **End Item** box.
 - Clicking **MRU**  adjacent to the **End Item** box.

Note

If you want to remove the entered end item, click **Clear**  adjacent to the **End Item** box.

4. Click **OK** to save the effectivity data you entered.

Reuse components in different products

To support reuse of the same components in different products, you can define multiple effectivities on the same status of an item revision, if each effectivity specifies a different end item. To create an additional effectivity on a status that already has effectivity:

1. In the **Revision Effectivity** dialog box, select the appropriate line and click **Create**. Teamcenter displays the **Edit Revision Effectivity** dialog boxes, with blank effectivity ranges.
2. Define the new effectivity, as described in steps 2 and 3 of the previous procedure.

Managing nested effectivity

A product structure may include a product that has its own effectivity that is separate from that of the product structure. For example, the product structure of a car may include an engine that is obtained from an external supplier. In this case, the effectivity of the engine differs from the effectivity of the car. You can use nested effectivity to change the end item context when you configure the car, as only one effectivity can be set in a revision rule; in this example, you must set the effectivity from car to engine at the engine.

Note

You can use nested effectivity with date or unit number effectivity.

To accommodate variations of effectivity within a product, you create a *configuration item* to attach to each assembly that is configured by a different end item to the top-level item. In the previous example, you would create a configuration item for the engine and the effectivity of the configuration unit defines the engine assemblies. A configuration item is where the effectivity context of the structure changes; it defines a new end item for the affected substructure.

You can create a mapping table to define the ranges of dates or unit numbers in the top-level product that configure a particular unit number or effective date in the lower level assembly. When you expand the product structure and apply a revision rule, the unit number or date set at the top level is converted to the scheme defined in the mapping specified for the assembly.

For more information about nested effectivity, see *Getting Started with Product Structure*.

Configuring nested effectivity

To configure nested effectivity, your Teamcenter administrator must set the **EffectivityConfigurationItemProperties** preference. This preference contains the item properties that identify configuration items. For example, the following entry defines the name of the type of item as the identifying attribute:

isConfigurationItem
object_type:object_name

Each entry in the list must be followed by the corresponding values. For example, the following example states that a configuration item is of the **Product** or **Vehicle** item type:

EffectivityConfigurationItemProperty.isConfigurationItem=true
EffectivityConfigurationItemProperty.object_type.object_name=
Product
Vehicle

Creating a configuration item

You do not create a configuration item separately, but identify an item as a configuration item when you create it. To do this, ensure the **Configuration Item** check box is selected on the **New Item** wizard before you click **Finish**.

Creating a revision rule for nested effectivity

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

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

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

Item	Status, effectivity	Note
NE-A2000/A	Released, EI: NE-A5000, 15 July-UP	Rev Effect wrt NE-A5000 – Product Y

The resulting configuration item effectivity mappings are:

- Mapping on NE-A2000 (CI):
 - NE-A1000 (CI) – Product X : NE-A2000 (CI)
 - NE-A5000 (CI) – Product Y : NE-A2000 (CI)
- Mapping on NE-A3100 (CI):
 - NE-A2000 (CI) – Product Y : NE-A3100 (CI)

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

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

Item	Revision	Revision Effectivity	Rule configured by	Status
NE-A5000A-Product Y (view)	A	Released 15-Jul-2005 00:00 to UP (NE-A5000)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A5000A-Product Y)	Released
NE-A2000A-Module K (CI) (view)	A	Released 20-Jul-2005 00:00 to UP (NE-A2000)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A2000A-Module K (CI))	Released
NE-A1000A-Allen Assy (view)	A	Released 20-Jul-2005 00:00 to UP (NE-A2000)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A2000A-Module K (CI))	Released
NE-PP100A-Piece Part 10	A	Released 20-Jul-2005 00:00 to UP (NE-A2000)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A2000A-Module K (CI))	Released
NE-PP200A-Piece Part 20	A	Released 16-Jul-2005 00:00 to UP (NE-A5000)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A5000A-Product Y)	Released
NE-A3100A-Module P (CI) (view)	A	Released 23-Jul-2005 00:00 to UP (NE-A3100)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A3100A-Module P (CI))	Released
NE-A4000A-General Assy (view)	A	Released 25-Jul-2005 00:00 to UP (NE-A3100)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A3100A-Module P (CI))	Released
NE-PP700A-Piece Part 70	A	Released 25-Jul-2005 00:00 to UP (NE-A3100)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A3100A-Module P (CI))	Released
NE-PP800A-Piece Part 80	A	Released 25-Jul-2005 00:00 to UP (NE-A3100)	Has Status(Released, Configured Using Effective Date), Date(27-Jul-2005 17:12), End Item(NE-A3100A-Module P (CI))	Released

Nested effectivity 1

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

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

Item	Revision	Revision Effectivity	Rule configured by	Status
NE-A5000A-Product Y (view)	A	Released 15-Jul-2005 00:00 to UP (NE-A5000)	Has Status(Released, Configured Using Effective Date), Date(16-Jul-2005 18:51), End Item(NE-A5000A-Product Y)	Released
NE-A2000A-Module K (CI) (view)	A	No configured Revision	No configured Revision	Released
NE-A1000A-Allen Assy (view)	A	No configured Revision	No configured Revision	Released
NE-PP100A-Piece Part 10	A	No configured Revision	No configured Revision	Released
NE-PP200A-Piece Part 20	A	Released 16-Jul-2005 00:00 to UP (NE-A5000)	Has Status(Released, Configured Using Effective Date), Date(16-Jul-2005 18:51), End Item(NE-A5000A-Product Y)	Released
NE-A3100A-Module P (CI) (view)	A	No configured Revision	No configured Revision	Released
NE-A4000A-General Assy (view)	A	No configured Revision	No configured Revision	Released
NE-PP700A-Piece Part 70	A	No configured Revision	No configured Revision	Released
NE-PP800A-Piece Part 80	A	No configured Revision	No configured Revision	Released

Nested effectivity 2

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

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

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

The screenshot shows a configuration tree on the left with items like NE-A5000-Product Y, NE-A2000-Module N, NE-A1000-Base Assy, NE-PP100-Place Part 10, NE-PP200-Place Part 20, NE-A2100A-Module P, NE-A5000-Central Assy, NE-PP1001-Place Part 10, and NE-PP60173-Place Part 60. On the right, a table titled 'Revision Effectivity' lists rules for these items, including release dates and effective dates. A status bar at the bottom indicates 'No configured Revision'.

Item	Revision	Revision Effectivity	Rule configured by	Status
NE-A5000-Product Y (view)	A	Released 15-Jul-2005 00:00 to LP (NE-A5000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A5000-Product Y)	Released
NE-A2000-Module N (CI)	A	Released 20-Jul-2005 00:00 to LP (NE-A2000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A2000-Module N (CI))	Released
NE-A1000-Base Assy (view)	A	Released 20-Jul-2005 00:00 to LP (NE-A1000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A1000-Base Assy (view))	Released
NE-PP100-Place Part 10	A	Released 20-Jul-2005 00:00 to LP (NE-A2000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A2000-Module N (CI))	Released
NE-PP200-Place Part 20	A	Released 20-Jul-2005 00:00 to LP (NE-A2000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A2000-Module N (CI))	Released
NE-A2100A-Module P (CI)	A	Released 16-Jul-2005 00:00 to LP (NE-A5000)	Has Status(Released), Configured Using Effective Date 3, Date(23-Jul-2005 17:12), End Item(NE-A5000-Product Y)	Released
NE-A5000-Central Assy (view)	A	Released 23-Jul-2005 00:00 to LP (NE-A3100)	No configured Revision	???
NE-PP1001-Place Part 10	A		No configured Revision	???
NE-PP60173-Place Part 60	A		No configured Revision	???

Nested effectivity 3

Create effectivity mapping on a configuration item

Effectivity mapping allows you to modify the parameters of the revision rule (for example, the end item, date and unit number) as you cross the borders of configuration items. Teamcenter attaches a mapping table to the revisions of the configuration item, and each entry in this table defines an optional end item and an optional effectivity range that defines the effectivity of the mapping itself. A mandatory subeffectivity entry defines the effectivity parameters used in the configuration item's substructure.

To view or define the effectivity map for an item, select the item corresponding to the configuration item in the product structure tree and choose **Tools→Effectivity→Effectivity Mapping**. Teamcenter displays the **Effectivity Mapping** dialog box allowing you to view any defined mappings for the configuration item or create new mappings.

To create a new effectivity map in the **Effectivity Mapping** dialog box:

1. Select an end item to which the effectivity map applies. The end item is *not* normally the configuration item.
2. Optionally, select the **Use Shared Effectivity** check box to use an effectivity map and share it with another end item. You can also select the **Create New** check box to create a new effectivity map and make it available to share with other end items.

Note

You can share any effectivity type, not just effectivity maps.

3. Define a date or unit effectivity in the usual format.
4. Define the subeffectivity.

Optionally, you can select the **Use last release date** check box to set the subeffectivity date to that one which the configuration item is released. This allows you to configure the released substructure without needing to specify the exact date.

Note

A blank entry in the mapping table indicates **ALL**.

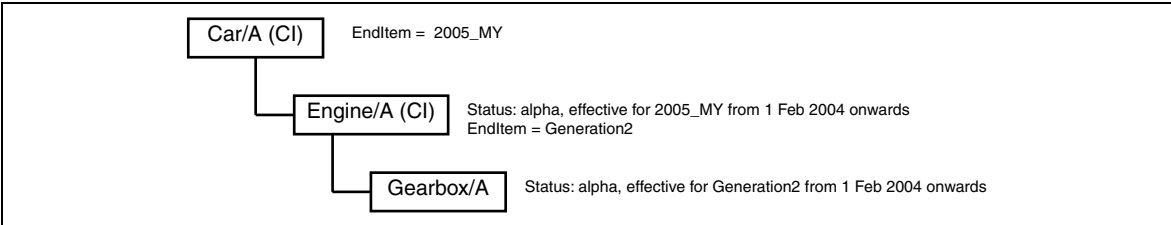
To create a new effectivity map, select an item revision in the **Effectivity Mapping** dialog box and click the **Create** button. Teamcenter displays the **Create Effectivity Mapping** dialog box, allowing you to define the parent product effectivity and the subeffectivity of the item revision.

Similarly, to edit an existing effectivity map, select an item revision in the **Effectivity Mapping** dialog box and click the **Edit** button. Teamcenter displays

the **Edit Effectivity Mapping** dialog box, allowing you to modify the parent product effectivity and the subeffectivity of the item revision.

Managing product generations

Instead of referencing a specific configuration item when you define an effectivity map, you can reference a separate entity—the generation item. The following figure shows a simplified product structure in which the effectivity statements referenced separate generation items (**2005-MY** and **Generation2**), rather than a specific configuration item. These generation items are items that can exist independently of a product structure.



Product generations

The effectivity map of this structure is as follows:

End item/revision	Effectivity range	Subeffectivity
2005_MY		Gen2
2006_MY		Gen3
002585-Truck	Jan 2007 – Apr 2007	Jan 2007
002997-Truck	May-2007 —	Gen2, Apr 2007

In this example, the **2005_MY** parent end item configures the **Gen2** end item when you enter the **Engine/A** assembly. Similarly, **2006_MY** configures **Gen3**.

The final line of the example shows how you can transform multiple parameters (**End Item** and **Date**) in a single mapping.

Defining end item revisions

To implement product generations, you can create *end item revisions*. You define end item revisions in the same way as end items, but additionally specify a revision of the end item. For example, you define the following effectivity:

Release status	Unit/date range	End item/revision
Production	10-Feb-2007 00:00 to UP	002988–Car/A
Production	01-Apr-2007 00:00 to UP	002988–Car/B

In this example, the effectivity of a component may have a different effectivity, depending on the revision of the car that is configured. The component is effective from 10–Feb in **Car/A** and from 01–Apr in **Car/B**. If you create a revision rule that defines an end item revision and Teamcenter encounters an object that has effectivity based on the end item (not the revision), it successfully matches if the end item revision is a revision of the end item.

You can also *pack* or collapse all effectivity entries that share the same release status and unit or date range into a single line by selecting the **Packed** check box in the **Effectivity** dialog box.

Using revision rules from My Teamcenter

You can create revision rules and save them as workspace objects. These workspace objects can be referenced by folders in other applications, for example, My Teamcenter. You can copy a revision rule into My Teamcenter.

You can also search in My Teamcenter to locate a particular rule. You can then reference a rule within a particular item or revision for configuration purposes.

You can view the contents of a revision rule in My Teamcenter by clicking the **Viewer** tab. This displays the list of entries in the rule.


Managing occurrence to part relationships

By default, an occurrence automatically references a particular item revision, depending on the revision rule currently in effect. Optionally, you can set the occurrence to unconditionally reference a precise revision of the part. You can then detect any precise occurrences that reference an out-of-date revision and manually upgrade each occurrence to an approved revision of the referenced part. This option is available only with a precise structure.

Note

The administrator defines the valid approved status of an item revision in the **TC_ValidApprovedStatus** site preference.

Show occurrences of superseded item revision

- When you create a new item revision and it is approved, the occurrences of older revisions are not automatically updated with the new revision if the structure is precise. To identify superseded item revisions, right-click a line with children and choose **Edit**→**Show Superseded**. Teamcenter places the superseded item revision symbol () against each child with an approved later item revision.

Update precise occurrences of superseded item revision

- To update an occurrence with an old revision to the new approved revision, right-click the occurrence and choose **Update Superseded**. Teamcenter shows the **Update** dialog box listing all the approved revision of the item, and you should select a revision and click **OK**. You can update all the applicable occurrences in the parent assembly or just the currently selected component.

Capturing configurations

Teamcenter provides the following formats for persistently capturing the configuration of a structure:

- [Snapshots](#) containing only the configuration (item revision), not the associated data. Consequently they require relatively little disk space.
- [Baselines](#) containing a copy of the complete working data for the structure. Baselines occupy a significant amount of disk space, as you create a copy of the structure each time you baseline it.
- [Intermediate data capture \(IDCs\)](#). IDCs are similar in purpose, but save the configuration in PLM XML format. You can create IDCs in Structure Manager and view them in the Multi-Structure Manager application.
- Product view. When you click the floppy disk button on the Viewer menu bar, Teamcenter saves a PLM XML file that contains a visual representation of your current viewer session. The session file may subsequently be reopened to show the saved product view.

Using snapshots

A *snapshot* is a folder that stores all the item revisions contained in a configured product structure. You can use the snapshot to redisplay the structure *as saved*.

Note

You cannot create snapshot folders for precise baselines, only imprecise baselines.

Create a snapshot

1. Configure the structure you want to save by applying the appropriate revision rule.

2. Choose **File→New→Snapshot**.

Teamcenter displays the **Create Snapshot** dialog box.

3. In the dialog box, type a name for the snapshot folder and (optionally) a description.

To keep track of snapshots and what they refer to, name them carefully. Use the **Name** and **Description** boxes to do this. You may want to record the name of the revision rule used to construct the snapshot in the **Description** box.

4. Click **OK** to create the snapshot folder and attach it to the top level of the structure by a **TC_snapshot** relation.

This action performs a full expansion of the structure, which may take some time for a large structure.

Open an existing snapshot

1. Select the snapshot folder in My Teamcenter.
2. Drag-and-drop the snapshot folder into the Structure Manager application. The structure is displayed as you saved it in the snapshot.

The ad-hoc revision rule applied in this Structure Manager window uses the snapshot folder as an override list. It also configures precise references.

View a snapshot

- You can view the item revisions stored in a snapshot by expanding the snapshot folder in My Teamcenter.

Note

You should protect snapshots against modification of their contents using the Access Manager rule tree as described in ***Unsatisfied xref title***.

Using baselines

During the development of a product design, you may want to share your working design with other users. You may also want to save an interim version of your design for future reference. To do this, you can create a *baseline* of the work-in-progress (WIP) design. When you request a baseline, Teamcenter creates a new revision for each unreleased revision in the structure and releases it with a predefined status, for example, **Baseline**. This method configures a completely released structure and thereby guarantees that the models are the same as when the baseline was created. This approach may be expensive as many new revisions might be created and (with them) copies of the associated data and CAD designs.

Note

The **Baseline_precise_bvr** preference controls whether a precise or imprecise baseline is created.

A baseline contains a deep snapshot of a selected item revision. When you create a baseline, you can also copy datasets, forms and other attachments associated with the item revision. Teamcenter observes deep copy rules when creating a baseline.

For more information about defining deep copy rules, see the *Business Modeler IDE Guide*.

Before creating baselines, you must use the Business Modeler IDE to create the appropriate baseline naming rules for your site and attach them to the appropriate item revision types, as described in [Define baseline naming rules](#). You must also create a **Baseline** status type in Business Modeler IDE.

To display a baseline in Structure Manager, load the top line after applying a revision rule that contains a **Has Status=Baseline** entry and release date corresponding to the baseline's creation date. The baseline may also include any new revisions created by the baseline.

You may not want to create a baseline for all item types, for example, you may not want to baseline engineering changes and documents. Such exclusions are determined by the setting of the **Baseline_restricted_item_types** preference, as described in the *Preferences and Environment Variables Reference*.

If your company shares baselines between different sites with Multi-Site Collaboration, you may have to manually checkout replicated items owned by other sites when you create the baseline. Similarly, you may have to manually check in the replicated items when the baselining process is complete. Alternatively, Teamcenter may automatically check out and check in replicate items. Your Teamcenter administrator sets the **Baseline_auto_remote_checkout_allowed** preference to determine if the checkin and checkout process is automatic or must be completed manually.

Optionally, your site may use *smart* baselines. If so, a new baseline includes only item revisions that have changed since the previous baseline was created, and unchanged item revisions are referenced. If your administrator sets the **ITEM_smart_baseline** preference to **ON**, this option is automatically selected each time you create a new baseline.

If you want to release the baseline immediately after creation, you must create the necessary workflow and add the workflow name to the **Baseline_release_procedures** preference. A baseline release process must adhere to a *quick release* template. Quick release templates are process templates that define a zero-step release procedure, allowing the baseline to become a released object that cannot be modified.

For more information, see the *Workflow Designer Guide*.

Note

Set the **Baseline_allow_edits** preference to **ON** to allow users to edit the item ID of items that have been baselined. Once the item is released, the item ID can no longer be modified.

If you are working with NX, Teamcenter may synchronize (refile) the assembly to NX before creating the baseline. This action is optional and depends on the setting of the **Baseline_nxmanager_refile** preference. You may also set the **Baseline_refile_required_dtypes** and **Baseline_refile_not_required_item_types** preferences to refine the refile operation.

Define baseline naming rules

Use the Business Modeler IDE to create the appropriate baseline naming rules for your site. The following example shows how to format a naming rule in the Business Modeler IDE for baselines of items:

```
Item Rule
A Released Rev
A.001 PDI
A.002 PDI
A.003 PDI
B Released Rev
B.001 PDI
B.002 PDI
C Working Rev
```

To establish this rule:

1. Create an item business object (type) called **ItemPDR**, if it does not already exist.
2. Create a naming rule called **Baseline rev rule** (any name can be used) with the following pattern:

```
" . "NNN
```
3. Attach the **Baseline rev rule** to the baseline suffix property on the **ItemPDR** item business object.

For information about using the Business Modeler IDE, see the *Business Modeler IDE Guide*.

Create a baseline

1. In the product structure tree, select the top line item revision of the structure you want to capture and choose **Tools**→**Baseline**.

Teamcenter displays the **Baseline** dialog box.

Note

You can only baseline work in process item revisions, not release item revisions.

2. In the dialog box, fill in the following boxes:

Baseline Revision

This is generated according to site preferences and naming rules defined in the Business Modeler IDE. You cannot change the displayed value.

Description

Enter a description that is stored with the baseline.

Release Procedure

Select a release procedure from the dropdown list. The available release procedures are defined by the administrator with the **Baseline_release_procedures** preference. The Teamcenter administrator predefines this Workflow process and it must not include any signoffs.

Job name

Teamcenter generates this name in the format **Baseline_ItemID_BaselineRevID**. You cannot change the displayed value; it is truncated to 32 characters, if necessary.

Job Description

Enter a description of the baseline job.

Baseline Label (optional)

Enter an alphanumeric string that represents the baseline label. Teamcenter uses the label you enter as the name of the baseline folder. This box is only displayed if your site uses baseline labels.

3. In the dialog box, select one or more of the following check boxes and click **OK** to create a baseline or dry run:

Open on Create

Opens the baseline automatically after it is created.

Dry Run Creation

Creates a report, as described in [Creating a dry run](#).

Precise Baseline

Selects the type of baseline to create—precise or imprecise. Your site may be configured to allow only precise baselines, depending on the setting of the **Baseline_precise_bvr** preference.

Finding the baseline folder

After creating a baseline, Teamcenter optionally places all the item revisions in the structure in a baseline folder. The creation of this folder is optional and depends on the value of the **Baseline_create_snapshot_folder** site preference. It attaches this folder to the top level of the structure.

If the value of this preference is **0**, the baseline folder name is in the format **Baseline_ItemID_BaselineRevID** and may be truncated to 32 characters, if necessary.

If the value of this preference is **1**, the baseline folder name is the alphanumeric string you entered in the **Baseline Label** box in the **Baseline** dialog box.

Note

In addition to setting this preference, you must set the **Snapshot** relationship on the item revision to make the snapshot folder visible. Do this by choosing **Edit**→**Options**→**General**→**Item Revision** and add **Snapshot** to the **Shown Relations** column.

Creating a dry run

Performing a dry run of a baseline allows you to validate the data associated with the base item revision or structure before you create the actual baseline. This avoids the necessity to roll back the baseline, if Teamcenter encounters errors during its creation.

The dry run generates a report that contains error messages, if Teamcenter encountered any problems while traversing the structure.

Your administrator can configure Teamcenter to always make a dry run, by setting the value of the **baselineDryRun** and **Baseline_dryrun_always** preferences.

Using intermediate data captures

An IDC is a PLM XML file that contains the definitions of all the objects in the captured structure. You can manage the PLM XML file in the same way as any other workspace object, including assigning it to a workflow, assigning it a release status and controlling access privileges with Access Manager.

You can create intermediate data captures (IDC) that contain the configuration of a structure at the time the IDC is created. An IDC may contain any configured structure including a collaboration context, structure context or group of structure lines. You can view IDCs with the Multi-Structure Manager application.

You can create an IDC that stores the current state of the selected root object and any related objects, such as structure lines or attachments. You cannot capture workspace objects that are not in the structure.

Capture structure lines into IDC

1. Select the root object in the structure and choose **Tools→Intermediate Data Capture**.

Teamcenter displays the **New Intermediate Data Capture** dialog box.

2. Select **IntermediateDataCapture** from the list of IDC types at the left of the dialog box.
3. Enter the name and optional description of the intermediate data capture, choose a transfer mode name from the dropdown list, and then click **OK** or **Apply**.

Teamcenter validates the objects in the structure you selected. If any of the objects cannot be captured, it displays an error message, otherwise it creates the PLM XML file containing the IDC.

Note

Ensure you select a transfer mode that is appropriate for the data you want to capture, for example, **ConfiguredDataExportDefault** or **BOMwriterExport**. The displayed list shows all transfer modes that are available in the system.

Lesson

20 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:

- Identify the key components to rule-based protection.
- Evaluate the rule tree.
- Create a new rule in the rule tree.
- Create a new access control list (ACL).
- Import and export the Access Manager rule tree.
- Verify the effect of access rules.
- Configure group security.

Help topics

Additional information for this lesson can be found in:

- [*Access Manager Guide*](#)
- [*Security Administration Guide*](#)

Introduction to Access Manager

Access Manager enables you to control user access to data objects stored in Teamcenter by:

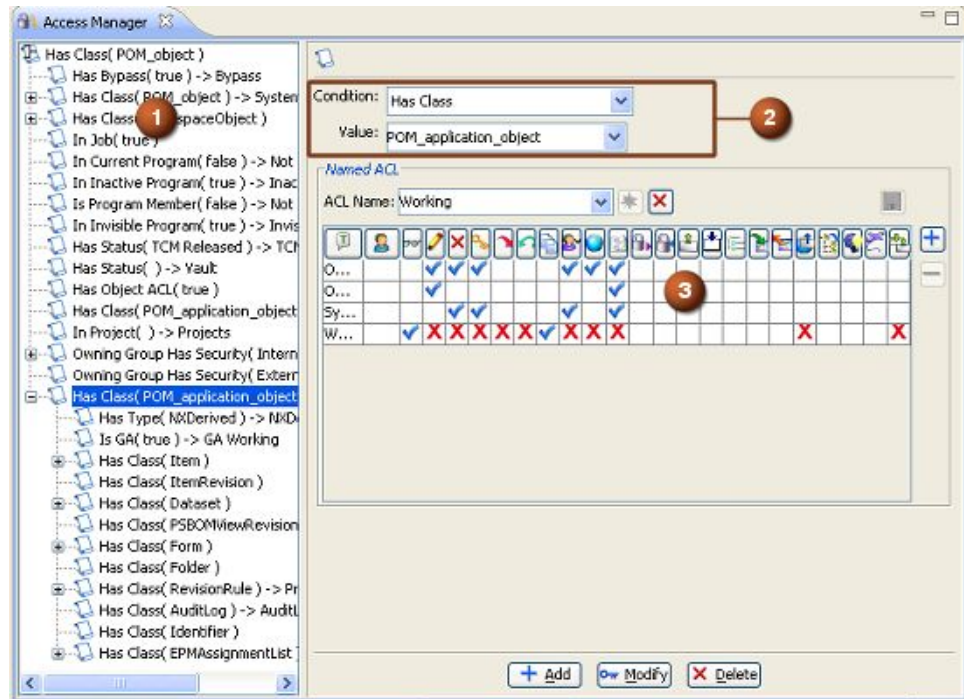
- Defining rules.
- Defining access control lists (ACLs).

Rules and ACLs are used in combination with information about the user, such as group membership, project membership, nationality, and clearance level, which together determine the user's authorization to interact with data.

Access Manager is an administrative application that leverages:

- User information maintained in the Organization application.
- Project information created using the Project application.
- Object metadata and business rules that are defined and maintained using the Business Modeler IDE.

Access Manager interface



1 Rule tree pane

Enables you to view the structure of your access rules by expanding and collapsing branches. Select a rule in the tree to see the rule properties and named ACLs in the rule properties pane.

2 Rule properties

Displays the condition and value for the rule selected in the rule tree. You can modify these properties and then create or modify a rule. You can delete the selected rule.

3 Named ACL table

Displays the ACL name and accessor entries for the rule selected in the rule tree. You can create, modify, and delete named ACLs.

Basic concepts for using Access Manager

Use Access Manager to define various conditions or rules that control who can or cannot access various objects.

- Administrators define global, rules-based protection, which affects your entire Teamcenter site.
- Administrators or users can grant exceptions to these rules using access control lists (ACLs) for object-based protection.

To take full advantage of Access Manager, you should be familiar with the data access methodologies, rules, accessors, and privileges that are used to implement data access protections.

Basic tasks using Access Manager

Using Access Manager, you can:

- Create, modify, and delete rules.
- Create, modify, and delete access control lists (ACLs).
- Export and import the rule tree.

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 is the primary security mechanism.
- Object-based protection is a secondary security mechanism that allows you to grant exceptions to rules.

Rules-based protection

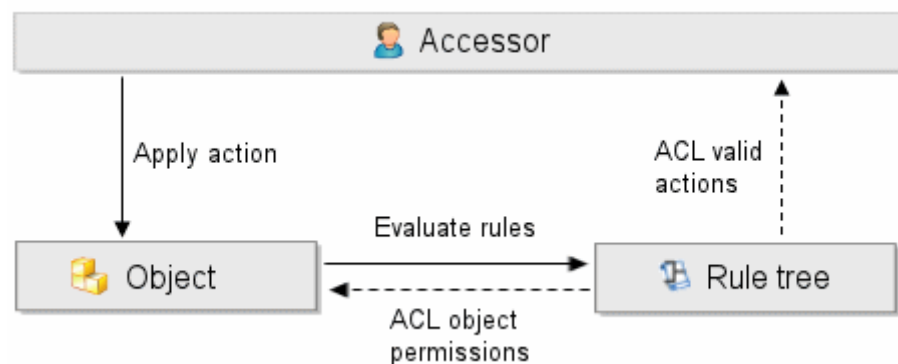
Rules provide security for your Teamcenter data by:

- Controlling access to data on a global basis.
- Determining whether a user has permission to view or perform an action on an object.
- Filtering data according to the attributes of the data.
- Granting privileges to the data according to the users' IDs and their session context (the group and role they used to log on).

Rules are defined by a combination of:

- A condition.
- A value for the condition.
- An access control list (ACL) that grants privileges to accessors.

The condition and value identify the set of objects to which the rule applies; the ACL defines the privileges granted to users (accessors).



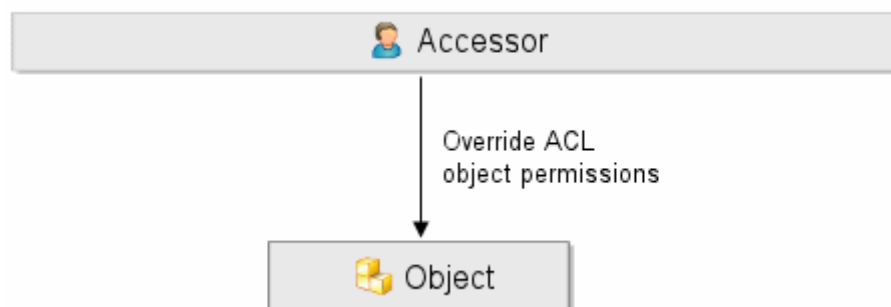
User actions against objects cause the rule tree to be evaluated to dynamically build an access control list for the object. The ACL controls permissions for the object and determines who (accessors) can do what (actions) to the object.

Object-based protection

Object-based protection uses access control lists (ACLs) to create exceptions to rules-based protection on an object-by-object basis.

Object ACLs are most useful when you need to:

- Grant wider access to a specific object.
- Limit access to a specific object.




























Teamcenter uses ACLs to determine access to an object. Users with proper permissions can override the ACL for an object to grant or deny permissions for certain users but only when the rule tree allows.

For example, the rule tree does not allow object-based access rules to override the rules-based protection when:

- An object has an assigned status.
- The object access rule is granted in a workflow.

Access control lists

										
System Administrator										
World										

Access control lists (ACLs) contain a list of accessors and the privileges granted, denied, or not set for each accessor.

Access control lists display the current protections for an object.

Accessors

Accessors are collections of users who share certain common traits, such as membership in the group that owns the object or membership in the project team. Just as rules have a precedence weighting in the rule tree, accessor precedence weighting is considered when the ACL is evaluated.

Each pairing of an accessor with corresponding privileges in the list is referred to as an *access control entry (ACE)*. An ACL can be comprised of one or many ACEs.

ACLs are associated with conditions in the rule tree as part of a rules-based security model, and they can be used in more than one rule.

In addition, object ACLs grant exceptions to rules-based protection and are created by users with change privileges.

Life cycle of data

All data in an enterprise typically passes through three basic phases, **Released**, **In-Process**, and **Working**.

Data state	Description
Released	Data is formalized and must be protected from modification. Released data is often consumed by users outside the authoring group; whereas, in-process and working data is consumed by authors and generally requires more restrictive read access.
In-Process	Data is semiformalized and because it is in the process of being released, it is assumed to be accurate and in its final form. However, allowances must be made for last-minute changes. The primary objective for protecting in-process data is to ensure that it is tightly controlled while it is being released.
Working	Data is not very firm and is expected to undergo many changes before it is released. The objective for protecting working data is to ensure that only the proper persons have permission to view, modify, or manipulate the data.

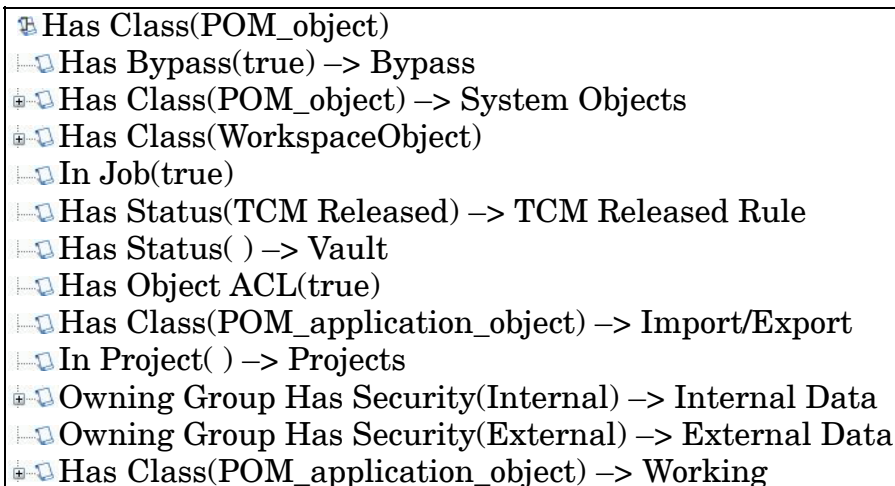
Access Manager rule tree

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 privileges unless explicitly denied.

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 ACL are applied.

- The rules are evaluated from the top to the bottom of the tree.
- Rules at the top take precedence over rules at the bottom of the tree.
- Subbranches always take precedence over parent branches in the tree.

The rule tree appears to the left of the Access Manager window.



```

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
  
```

Instructor Note:

For a list of default rule conditions, see the *Access Manager Guide*.

















How rules work

Rules are defined by a combination of a condition, a value for that condition, and an access control list (ACL) that grants privileges to accessors.

- The condition and value identify the set of objects to which the rule applies.
- The ACL defines the privileges that are granted to users (accessors) specified in the ACL.

IF *condition = value* is **TRUE**, **THEN** apply ACL to object.

Example ACL

 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy
World								

Rule syntax

The following syntax applies to rules:

Condition {Value} -> ACL

The parts of the rule can be thought of as an **IF** clause and a **THEN** clause.

- The condition and value supply the **IF** part of the rule and examine the object with Boolean logic.
- The access control list (ACL) supplies the **THEN** part of the rule by describing the access permission.

For example:

Has Type {UGMASTER} -> UG Model

In this example, **Has Type** is the condition, **UGMASTER** is the value, and **UG Model** is the name of the ACL.

Evaluating the rule tree for the effective ACL

The rule tree evaluation results in an *effective ACL*. The effective ACL represents the cumulative compilation of all the named ACLs that apply to the object the user is trying to access.

The rule tree is evaluated as follows:

- Trim rules that do not apply to the object because their conditions are false.

Note

The rules are not removed from the tree, but they are ignored during evaluation.




















- Evaluate rules in order of precedence, from top to bottom.
- Evaluate the subbranch of a rule before evaluating the parent rule.
- Evaluate subbranch rules in order of precedence, from top to bottom, in the event that there are multiple subbranch rules.

The *effective ACL* is determined by compiling the ACLs in the order that the tree is traversed.

Example rule tree evaluation by order of precedence

This example rule tree shows the order of precedence in the left column, assuming all conditions are met.

- The first two rows are the first two rules evaluated because they are highest in the tree and have no subbranch.
- The third row only gets evaluated after all its subbranches are evaluated.

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

Activity

In the *Access Manager* section, do the following activity:

- Evaluate the rule tree.

Determine the order of precedence by writing a numbered next to each row corresponding to the order the rules are evaluated.

▢

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

Instructor Note:

10

▢

Condition {Value}@ Named ACL

3

▢

Condition {Value}@ Named ACL

1

▢

Condition {Value}@ Named ACL

2

▢

Condition {Value}@ Named ACL

4

▢

Condition {Value}@ Named ACL

9

▢

Condition {Value}@ Named ACL

5

▢

Condition {Value}@ Named ACL

8

▢

Condition {Value}@ Named ACL

6

▢

Condition {Value}@ Named ACL

7

▢

Condition {Value}@ Named ACL

Example of compiling an effective ACL

When the user attempts to access a **UGMASTER** dataset, the **rule tree** is trimmed to reflect only those rules that apply to the object.

Has Class(POM_object)

Has Class(POM_app_object) -> Working





Has Class(Dataset)










Has Type(UGMASTER) -> UGMASTER

Based on the trimmed rule tree, the effective ACL is compiled by evaluating the tree (from bottom to top) as follows:










1. Find the topmost leaf node in the tree, in this case, **Has Type(UGMASTER) -> UGMASTER**. Add the **UGMASTER** ACL to the effective ACL.
2. Find the next node, **Has Class(Dataset)**. This node has no associated ACL, so it does not contribute to the effective ACL.
3. Find the next node, **Has Class(POM_app_object) -> Working**. Add the **Working** ACL to the effective ACL.
4. Find the next node, **Has Class(POM_object)**. This node has no associated ACL, so it does not contribute to the effective ACL.

The rule tree evaluation results in the following effective ACL.

 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy	ACL
Role in Owning Group	Designer		✓					✓	UGMASTER
World			✗		✗			✗	UGMASTER
Owning User			✓	✓	✓				Working
Group Administrator				✓	✓				Working
Owning Group			✓						Working

 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy	ACL
System Administrator				✓	✓				Working
World		✓	✗	✗	✗	✗	✗	✓	Working

The effective ACL is evaluated when a user attempts to access a **UGMASTER** dataset. The lines that do not apply to the user are ignored. For example, if you are a designer in the owning group of the **UGMASTER** dataset, but you are not the owning user, system administrator, or group administrator, the following entries in the ACL are applied when you try to access a **UGMASTER** dataset.















 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy
Role in Owning Group	Designer		✓					✓
World			✗		✗			✗
World		✓		✗		✗	✗	








After the effective ACL is trimmed to include only the entries that apply to the user attempting to access the dataset, the privileges in the remaining ACL entries are evaluated. This is done by working down each privilege column until you encounter a granted ✓ or denied ✗ symbol.


In this example, the privilege evaluation grants the accessor read, write, and copy privileges and denies the accessor delete, change, promote, and demote privileges.

Access privileges

Following are some commonly used access privileges.

Symbol	Privilege	Description
	Read	Controls the privilege to open and view an object.
	Write	Controls the privilege to check the object out of the database and modify it.
	Delete	Controls the privilege to delete the object.
	Change	Controls the privilege to modify object protections that override the rules-based protection for the object. You must have change privileges to apply object-based protection (object ACLs).
	Promote	Controls the privilege to move a task forward in a workflow process.
	Demote	Controls the privilege to move a task backward in a workflow process.
	Copy	Controls the privilege to copy an object.
	Change ownership	Controls the privilege required to grant, change, or restrict ownership rights to an object.
	Publish	Controls the publish privilege to users or groups.
	Subscribe	Controls the privilege to subscribe to an event on a specified workspace object.
	Export	Controls the privilege to export objects from the database.
	Import	Controls the privilege to import objects in to the database.
	Transfer out	Controls the privilege to transfer ownership of objects when they are exported from the database.
	Transfer in	Controls the privilege to assign ownership of objects when they are imported in to the database.

Symbol	Privilege	Description
	Write Classification ICO	Controls the privilege to write Classification objects (ICOs).
	Assign to project	Controls the privilege to assign an object to a project. This applies to users who are not designated as privileged project team members. Note The validation of the Assign to project privilege in conjunction with privileged project membership is evaluated based on the value of the TC_project_validate_conditions preference.
	Remove from project	Controls the privilege to remove an object from a project. This applies to users who are not designated as privileged project team members. Note The validation of the Assign to project privilege in conjunction with privileged project membership is evaluated based on the value of the TC_project_validate_conditions preference.
	Remote checkout	Controls the privilege to remotely check out an object.
	Unmanage	Enables users to circumvent the blocking implemented using the TC_session_clearance preference. For more information about session clearance, see the <i>Security Administration Guide</i> .
	IP Admin	Enables users to add users to manage IP licenses.
	ITAR Admin	Enables users to add users to manage ITAR licenses.

Symbol	Privilege	Description
	CICO	Controls administrative override privileges for the checkin, checkout, transfer checkout, and cancel checkout features.

Categories of accessors

Category	Accessor	Description
General	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.
	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.
	Group Administrator	User who has special maintenance privileges for the group.
	World	Any user, regardless of group or role.

For a complete list of accessors, see the *Access Manager Guide*.

Complex rule tree example

This view of the default rule tree is used in the example that follows.

```

Has Class(POM_object)
├── Has Bypass(true) -> Bypass
├── In Job(true)
├── Has Status( ) -> Vault
├── Has Object ACL(true)
├── Has Class(POM_application_object) -> Working
│   ├── Has Class(Item) -> Items
│   ├── Has Class(Item Revision) -> Item Revs
│   └── Has Class(Dataset)
│       └── Has Type(UGMASTER) -> UGMASTER

```

A user, Jim Smith (**jsmith**), a designer in the engineering group, attempts to modify the **MyPart UGMASTER** dataset with working status. To perform this action, Jim Smith needs write privileges on the dataset.

The following ACLs are considered when the sample rule tree is evaluated:

1. The **Has Bypass(true) -> Bypass** rule is evaluated. This high-level rule grants system administration privileges to users.

Result: Jim does not have bypass set, nor is he a system administrator, therefore, this rule condition is false and the **Bypass** ACL is not applied. The evaluation moves down the tree to the next branch.

2. The **In Job(true)** rule is evaluated. This rule evaluates whether the object is in a workflow.

Result: No ACL is defined. Therefore, the condition being true has no effect. The evaluation moves down the tree to the next branch.

3. The **Has Status() -> Vault** rule is evaluated. This rule evaluates whether the object has an attached status type. If yes, the **Vault** ACL is applied.

Result: The **MyPart** dataset is in working status; therefore, the rule condition is false and the **Vault** ACL is not applied.

4. The **Has Object ACL(true)** rule is evaluated. This rule evaluates whether an ACL exists for the object.

Result: No object ACL is defined by a user, therefore the condition is false and has no effect. The evaluation moves down the tree to the next branch.

5. The **Has Class(Item) -> Items** rule is evaluated. This rule evaluates whether the object is of class item. If yes, the **Items** ACL is applied.

Result: The **MyPart** is of class dataset not item; therefore, the rule condition is false and the **Items** ACL is not applied.

6. The **Has Class(Item Revision) -> Item Revs** rule is evaluated. This rule evaluates whether the object is of class item revision. If yes, the **Items** ACL is applied.






Result: The **MyPart** dataset is of class dataset not item revision; therefore, the rule condition is false and the **Item Revs** ACL is not applied.

7. The **Has Type(UGMASTER) -> UGMASTER** rule is evaluated. This rule evaluates whether the object is of class **UGMASTER**. If yes, the **Items** ACL is applied.

Result: The **MyPart** dataset is of class **UGMASTER**; therefore, the rule condition is true and the **UGMASTER** ACL is applied.

UGMASTER ACL

The **UGMASTER** ACL explicitly grants write access to users who fill the **Designer** role in the owning group and explicitly denies write access to all other users in the owning group.

 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy
Role in Owning Group	Designer							
Owning Group								

8. The **Has Class(Dataset)** rule is evaluated. This rule evaluates whether the object is of class dataset.










Result: The **MyPart** dataset is of class dataset; therefore, the rule condition is true. No ACL is defined, therefore the condition being true has no effect.

9. The **Has Class(POM_application_object) -> Working** rule is evaluated. This rule evaluates whether the object is of the **POM_application_object** class. If yes, the **Working** ACL is applied to the object.

Result: All workspace objects, including datasets, are subclasses of the **POM_application_object** class; therefore, the rule condition is true and the **Working** ACL is applied.

Working ACL

The **Working** ACL explicitly grants write, delete, and change privileges to owning users and write privileges to the owning group. It also grants delete and change privileges to the group administrator and the system administrator. All other users are granted read and copy privileges and explicitly denied write, delete, change, promote, and demote privileges.

 Accessor	 User	 Read	 Write	 Delete	 Change	 Promote	 Demote	 Copy
Role in Owning Group	Designer		✓					
Owning User			✓	✓	✓			
Group Administrator				✓	✓			
System Administrator				✓	✓			
World		✓	✗	✗	✗	✗	✗	✓

Activity

In the *Access Manager* section, do the following activity:

- Effective ACL evaluation

Review questions

1. If the rule condition evaluates to false, what happens?

Select all that apply.

- The ACL is applied.
- The rule is ignored.
- Privileges are denied.

Instructor Note:

Answers to review questions

1. The rule is ignored.

Understanding the rule creation process

The basic process used to create rules is:

1. [Add a rule to the tree.](#)
2. [Create and save the access control list \(ACL\).](#)
3. [Attach the new ACL to the rule by modifying the rule.](#)

Tip



You must always save the rule or ACL after making modifications.

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

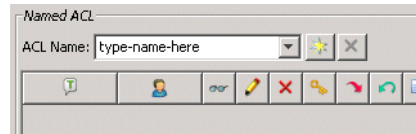
ACLs can be referenced in more than one rule.

3. Click the **Add** button  located below the ACL table.
4. Click the **Save** button  in the toolbar.




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 an access control list (ACL)

1. Enter the ACL name in the **ACL Name** box.





Named ACL box

2. Click the **Create** button  to the right of the **ACL Name** box.
3. Click the **Save** button  to the right of the **ACL Name** box.
4. Click **Add**  to add a new row to the access control entry (ACE) table.
5. Double-click the cell in the **Type of Accessor** column to select an accessor.
6. Double-click the cell in the **ID of Accessor** column to select an accessor ID.

Note

Some accessor types, such as **User**, **Group**, and **Role**, require you to select an accessor ID to define a specific instance of the accessor type. Other accessor types, such as **World** and **Owning Group**, are either singular or are relative to the object being accessed; therefore, no ID is required.



7. Set privileges by double-clicking the cell corresponding to the privilege you want to set, and choose  to grant privileges or choose  to deny privileges.

Note


Whenever possible, do not explicitly set privileges. Leaving privileges unset allows rules to accomplish focused objectives by allowing objects and accessors to filter through rules that do not apply to them.

8. Click **Save** .

Modify an Access Manager rule

1. Select the rule you want to modify.
2. Modify the condition or value in the rule pane.
3. To attach an ACL to the rule, select an ACL from the **ACL Name** list.
4. Click the **Modify** button  located below the ACL table.
5. Click the **Save** button  in the toolbar.

Note

When you make changes to a rule, the changes are not saved until you choose **File**→**Save** or click the **Save** button  on the toolbar.

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, you want to restore the original rule tree, 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 on the toolbar.

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.

Activities

In the *Access Manager* section, do the following activities:

- Export the Access Manager rule tree.
- Add an Access Manager rule to the rule tree.
- Import the Access Manager rule tree.

Review questions

1. When is the best time to export the rule tree?

Select all that apply.

- After making changes
- Never
- Prior to making changes

Instructor Note:

Answers to review questions

1. After making changes.
Prior to making changes.

Best practices

- **Understand your organization's business rules.**

A thorough understanding of your organization's business rules enables you to model access rules that support your business processes and are transparent to users. When modeled correctly, Access Manager rules grant users the privileges required to perform the tasks associated with their jobs while denying them access to data that is released or out of the scope of their functional role.

- **Document the business rules and the rule tree developed to meet them.**

Every rule in the rule tree and the named ACLs associated with the rules are included for a purpose. For maintenance purposes, Siemens PLM Software strongly recommends that you document the purpose of the rules, how they are populated, and why they have been populated. Future versions of Teamcenter add new rules and accessors. Merging new rules and accessors is a manual process, which is simplified if you have thoroughly documented the Access Manager rule tree.

- **Export the rule tree before and after making changes.**

When new rules do not work as expected, you must be able to restore an earlier, working version of the rule tree. A backup copy is essential to restoring rules back to their original state.

- **Add new rules for working data in the Working data branch of the tree.**

The proper location to add new rules for working data is under the **Working** data branch in the rule tree. This helps you customize your rule tree and identify working data.

```
Has Class(POM_application_object) -> Working
```

- **Whenever possible, leave privileges unset.**

Leaving privileges unset in ACLs allows rules to accomplish focused objectives, and it also allows objects and accessors to filter through rules that do not apply to them.

- **Populate access control lists (ACLs) sparingly.**

Explicitly grant privileges, and only deny privileges when you must block users from access that would otherwise be implicitly granted.

- **Use the Has Attribute condition to create custom rules based on any attribute of an object of a given class.**

For example:

```
WorkspaceObject:object_name=*x
PublicationRecord:security=suppliers
```

The class and attribute names are not case sensitive. The attribute type can be **string**, **double**, **integer**, **logical**, or **reference**.

This rule supports custom attributes.

- **Set security precedence.**

You can embed type-level security rules under project-level security rules to give the type-level security rules higher precedence than the project-level security rules. For example, the project administrator can add a subbranch under the **Has Class (Form)** rule entry to control access to certain form types that contain sensitive data. The rule for the form type is written as follows:

```
Has Class(Form)
  Has Type(Finance) -> finance_acl
```

If your site requires that project-level security rules take precedence over type-level security rules, you must embed project-level security rules under the type-level security rules. However, Siemens PLM Software does not recommend this practice.

- **Define relevant ACL names.**

ACL names are displayed in the rule tree and in dialog boxes throughout the Teamcenter interface. You can significantly enhance overall usability by defining these names carefully. For example, when creating an ACL for working data, name it according to the data type (for example, item, item revision, or **UGMASTER**) rather than a role name or some other description.

Note

ACLs can be referenced in more than one rule.

- **Use discretion in applying the Bypass ACL.**

The **Bypass** ACL grants all privileges to system administrators who have selected the user **Bypass** setting. Use discretion in applying this ACL.

Cautionary statements

- **Do not modify access control lists (ACLs) referenced by rules on the System Objects branch.**

Adding new rules, deleting rules, or in any way modifying existing rules on the **Systems Objects** branch of the rule tree may result in unpredictable behavior or loss of data.

```
Has Class(POM_object) -> System Objects
```

- **Do not modify the upper area of the rule tree.**

Deleting or changing the order of the branches in this area of the rule tree may result in unpredictable behavior or loss of data.

- **Do not use a text editor to modify rule tree files.**

Rule tree files are simple ASCII files and conform to a particular format. You can read rule tree files using any text editor; however, modifying them with a text editor can easily corrupt the file.

- **Do not use the infodba account to change object ACLs.**

It is assumed that objects owned by **infodba** are seed parts or other special-case objects.

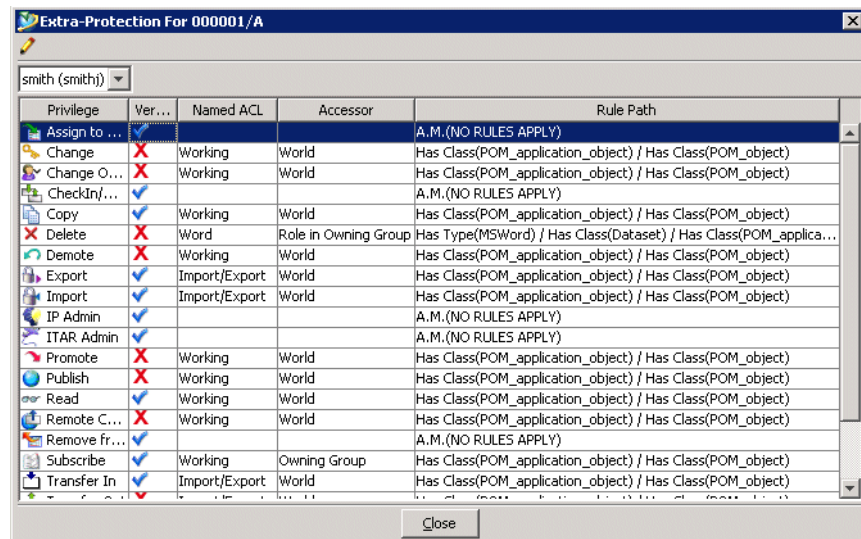
Verifying the effect of access rules

After you implement access rules, verify that the rules produce the desired privileges for different types of accessors. You can do this by viewing the access privileges in My Teamcenter. You can also determine which rules result in a privilege being granted or denied by viewing the verdicts in the **Extra Protection** dialog box.

View the rules from which privileges are derived

- In the **Access** dialog box, click .

The **Extra Protection** dialog box appears, showing the rules that apply to a privilege being granted or denied.



Extra Protection dialog box

Activities

In the *Access Manager* section, do the following activities:

- View default protections.
- Modify a named ACL.
- Export the Access manager rule tree.
- Add an Access manager rule to the rule tree.
- Import the Access manager rule tree.

Configuring group security

Group security works with access rules to control access to data based on a group's internal and external group security setting. It is important to understand how to configure security to limit access to data 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, access to data owned by internal groups is granted to users in internal groups but denied to external suppliers.

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 rules-based protection).

			
Owning Group			
Groups with Security	Internal		
Groups with Security	External		









Configuring security for data owned by a supplier (external data)

In this example, read access to data owned by a specific external supplier is granted to users in internal groups but denied to external groups other than the owning group. The external owning group, supplier, has read and write access to their data.

Use the following rule that defines privileges for all data owned by users who are members of external 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		

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Company group

2. Supplier group

Summary

The following topics were taught in this lesson:

- Rules-based protection is the primary security mechanism.
- Object-based protection allows you to grant exceptions to rules.
- Rules are evaluated based on their placement within the tree structure.
- The ACL defines the privileges that are granted to users (accessors) specified in the ACL.
- Export the rule tree before and after making changes.
- Group security protects data from being accessed by outside suppliers.

Instructor Note:

Summary instructor notes.

Lesson

21 *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*](#)

Instructor Note:

Lesson instructor notes.

Introduction to Project

The Project application provides a mechanism for organizing data and implementing access control based on project membership.

Project is used by Teamcenter administrators to:

- Set up projects.
- Assign users as members of projects.
- Implement access control based on project membership.
- Apply filters to determine the subset of project data that users view in My Teamcenter.







Project works with Access Manager to control access to data by project members.

Project administration window



- | | |
|---|---|
| <p>1 Quick Links</p> <p>2 Projects tree</p> <p>3 Project definition and rules</p> | <p>Displays links to navigate between the Project Administration and Smart Folder Administration panes.</p> <p>Displays the list of projects in Teamcenter. New projects created appear in this list.</p> <p>Displays the properties or project rule of the selected project. Use this pane to create the project, to assign project team members, to designate privileged team members and to apply access rules to a project.</p> |
|---|---|

Project administration buttons

Button	Description
 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 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 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.

Basic concepts about Project

Projects organize data and are the basis for granting data access to project team members. The following concepts apply to projects:

- Only privileged team members can assign data to projects.
- Data can be assigned to or removed from projects manually or when the data item is created, and items can be assigned to more than one project.
- Propagation rules define the associated data that is implicitly assigned to a project when a primary item is assigned to the project.
- All items in a complete product structure can be assigned to a project using the **update_project_bom** utility.

Project administrators and team members

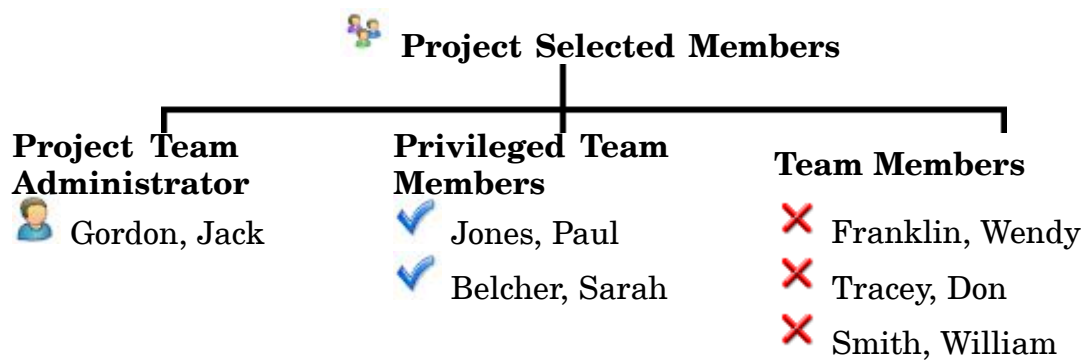
The following table describes the project administrators and members that can create, manage, and use projects.

Project administrators and members	Definition
Project administrator	<p>Teamcenter user with privileges to create and administer projects.</p> <p>Project administrators can:</p> <ul style="list-style-type: none">• Modify projects.• Delete projects.• Add team members to projects.• Assign privileges to team members.• Remove team members from projects. <p>Note</p> <p>These privileges only apply to projects that the project administrator has created.</p>
Project team administrator	<p>Project team member with privileges to modify project information. These privileges apply to the project metatdata, not to the data assigned to projects.</p> <p>Project team administrators can:</p> <ul style="list-style-type: none">• Add team members to projects in which the team administrator is also a member.• Remove team members from projects in which the team administrator is also a member. <p>Note</p> <p>There can be only one project team administrator per project.</p>

Project administrators and members

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.

The following diagram illustrates a typical project hierarchy.



Using 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, privileged team members, and project team administrator.
3. Set a user's default project.
4. As database objects are created, they can be assigned to the project automatically or manually by privileged team members.

Applying project security (Access Manager) rules

Project administrators can extend the default security rules, which grant read access to project data to members of the project team, on a project-by-project basis.

Note

Project administrators only have access to the  **In Project()** -> **Projects** branch of the rule tree.

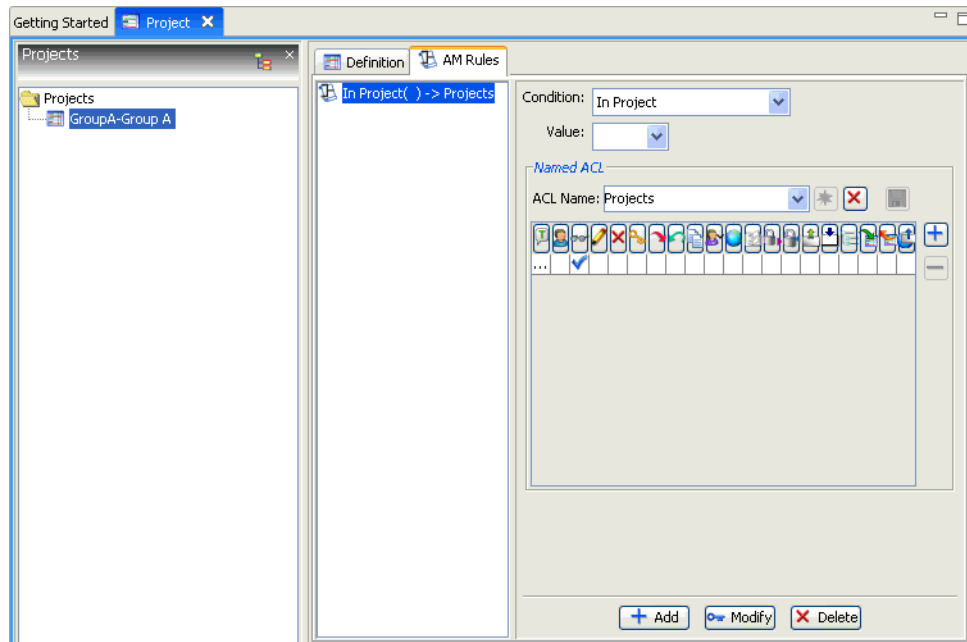
Using the Project branch in the rule tree, you can:

- Grant or deny access to a particular group of users by applying the **Owning Group** condition.
- Grant or deny access to users based on their membership in a project by applying the **Is Project Member** condition.
- Grant or deny access to groups of users based on the group's categorization as internal (OEM) or external (supplier) by applying the **Owning Group Has Security** condition.

For more information about configuring project-level security rules, see the *Security Administration Guide*.

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.

Modifying existing projects

1. Select an existing project from the **Project** tree.
2. Modify project settings by:
 - Adding or removing team members.
 - Assigning a new project team administrator.
 - Assigning or removing privileged team member status.
 - Changing security rules.
3. Click **Modify**.

Your modifications to the project information are saved to the database.

Note

Siemens PLM Software does not recommend changing the project name or ID once data has been assigned to an active project.

Assigning data to projects

End users can manually assign data objects to projects. In addition, you can configure Teamcenter to automatically assign related objects to a project when the primary object is assigned to the project, and you can use the **update_project_bom** utility to assign or remove objects in a product or manufacturing structure to or from a project. Teamcenter can also be configured to automatically assign data objects to projects when the object is created.

Data objects can be assigned to projects in one of three ways:

- Manually assigned to projects by users who are designated as privileged team members. Privileged team members include:
 - Project administrators
 - Project team administrator
 - Privileged team members

Note

Additional project security can be achieved using the **ASSIGN_TO_PROJECT** and **REMOVE_FROM_PROJECT** privileges. The exact behavior of these privileges is controlled by the **TC_project_validate_conditions** site preference.

- Automatically assigned to projects when the object is created. 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**.

- Automatically assigned to projects when the primary object is assigned to the project. Project propagation rules determine which secondary objects are assigned to a project when a primary object type is assigned.

You can use the **update_project_bom** utility to assign or remove objects in a product or manufacturing structure to or from a project.

Automatically assigning objects to projects

Teamcenter administrators can configure Teamcenter to automatically assign certain types of objects to a project when the specified objects are created by privileged team members.

The following object types can be configured for automatic assignment:

- Item and item revision subtypes, such as engineering changes and documents
- Forms
- Datasets

For example, Teamcenter can be configured to assign new item revisions to the *current project* of the user who creates the new item revision.

Note

Your current project is defined in the **User Settings** dialog box. You can choose the **Edit→User Setting** command to access the **User Settings** dialog box.

Use the Business Modeler IDE to configure the **autoAssignToProject** extension. This extension defines the type of objects that are automatically assigned to the user's current project when the specified object is created.

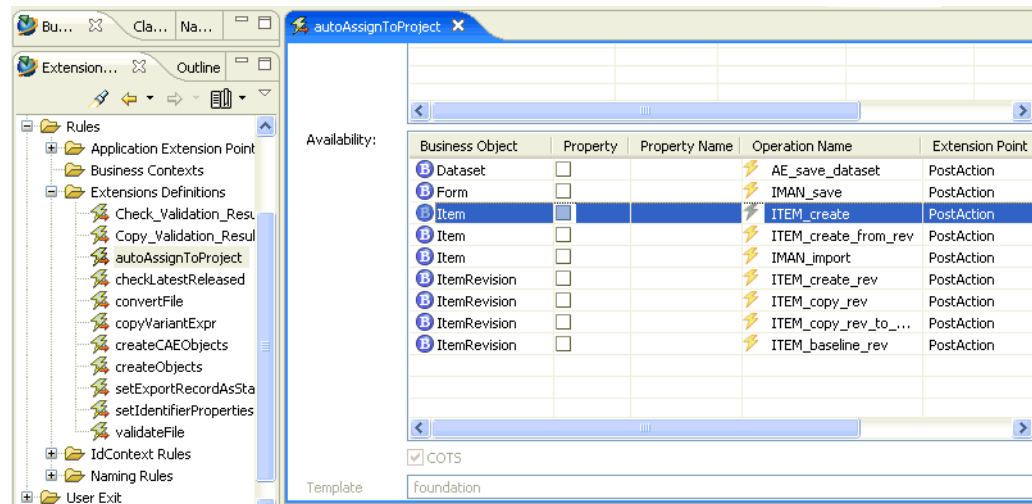
Considerations for automatic assignment of objects to projects

- When there is a conflict between a propagation rule and the execution of the **autoAssignToProject** extension, the extension takes precedence over the rule.
- Automatic project assignment applies only to object creation; whereas, propagation of related objects (level-one propagation only) occurs any time a relation between two objects is created, modified, or deleted.
- Automatic assignment of objects to projects is explicit; therefore, the object can only be removed from the project explicitly using the **Project→Remove** command.
- Propagation rules implicitly assign secondary objects to projects; therefore, when the primary object is removed from the project, the secondary object is also removed.

autoAssignToProject availability

From the **Business Modeler IDE** window, look up the availability of the **autoAssignToProject** extension on business objects.

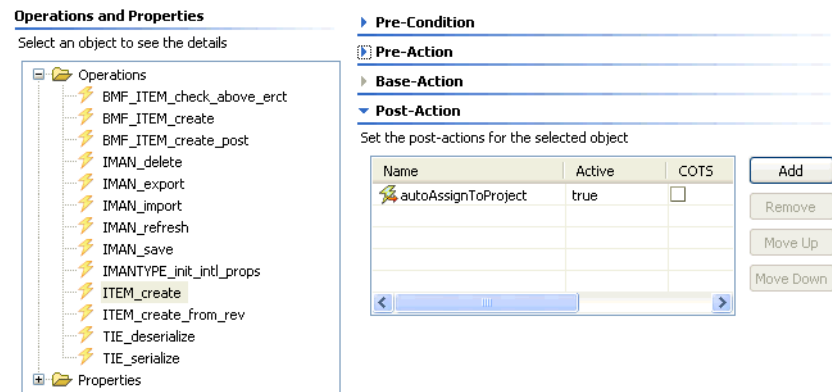
In the **Availability** table, view the business object, operation, and extension point for which the extension can be used. Decide which business object and operation for which you want to use the extension, and observe the extension point where it can be used.



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.

Activities

In the *Projects to control access* section, do the following activities:

- Configure automatic assign to project.
- Create a project with team members.
- Test automatic project assignment.

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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Project administrators
2. False

The project property is automatically filled in upon creation with the value of the current project for the user.

Summary

The following topics were taught in this lesson:

- 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).
- The **autoAssignToProject** extension allows the current project to be automatically assigned to newly created objects.

Instructor Note:

Summary instructor notes.

Lesson

22 *Teamcenter security*

Purpose

The purpose of this lesson is to understand basic security administration concepts and tasks.

Objectives

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

- Define authentication, authorization and authorized data access.
- Control access to working data.
- Control access to in-process data.

Help topics

Additional information for this lesson can be found in:

- [*My Teamcenter Guide*](#)
- [*Security Administration Guide*](#)
- [*Authorization Guide*](#)

Instructor Note:

Lesson instructor notes.

Introduction to Teamcenter security administration

Security administration is the process of establishing and maintaining control of access to objects in database. This is accomplished by using access control lists (ACLs) and rule conditions.

Teamcenter administrators are responsible for defining and implementing security controls, including but not limited to:

- Creating access rules and maintaining the rule tree.
- Creating access control lists (ACLs) corresponding to rules.
- Configuring project-level security to control access to data in specific projects.
- Configuring group security to control access for specific groups of users.
- Configuring authorized data access for International Traffic in Arms Regulations (ITAR) compliance and intellectual property (IP) protection.

You must have administrative privileges to perform most security-related tasks. In addition, some tasks require you to have administrative privileges related to the type of security being implemented. For example, you must have project administrator or project team administrator privileges to perform project-related security tasks.

Teamcenter security applications

The following Teamcenter applications are used to implement security solutions:

- Access Manager
- Project
- Workflow Designer

Note

If you have trouble accessing Teamcenter security applications, see your system administrator; it may be a licensing issue.

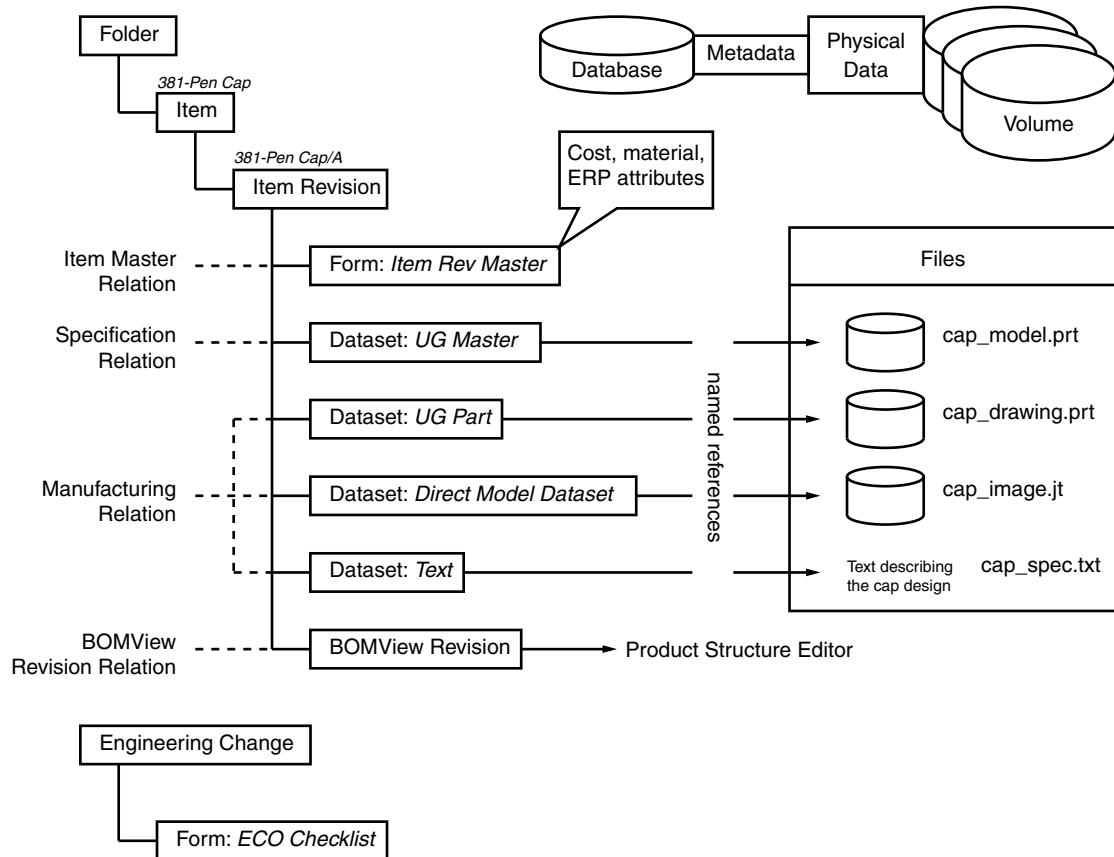
Basic concepts

Teamcenter security administration requires an understanding of:

- Object model hierarchy
- Authentication
- Authorization

Teamcenter object model hierarchy

The following is a simplified illustration of the major concepts of the Teamcenter object model hierarchy. It is important to understand the object model, as some of the security implementations described in this guide propagate information down the hierarchy structure.



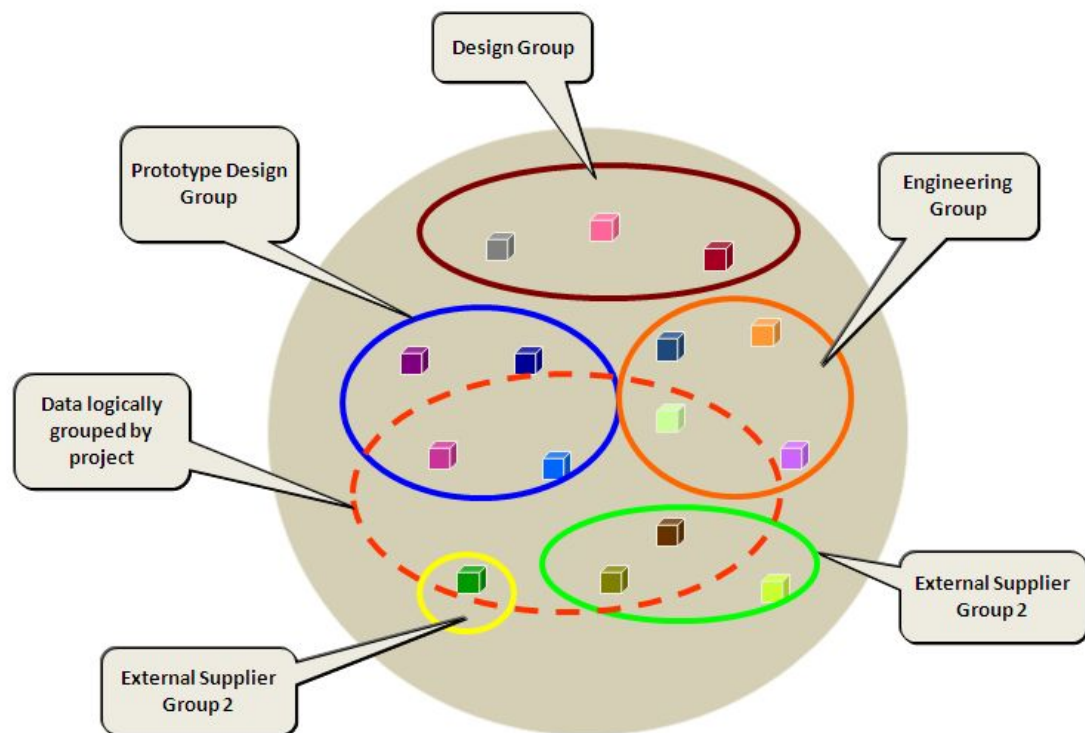
Authentication

Authentication is a process of determining whether someone or something is, in fact, who or what it is declared to be.

Authentication refers to gaining access to a Teamcenter application or product solution.

Authentication to load applications, such as Structure Manager or Change Viewer, in your Teamcenter session is provided by the Siemens PLM Software Common Licensing Server daemon.

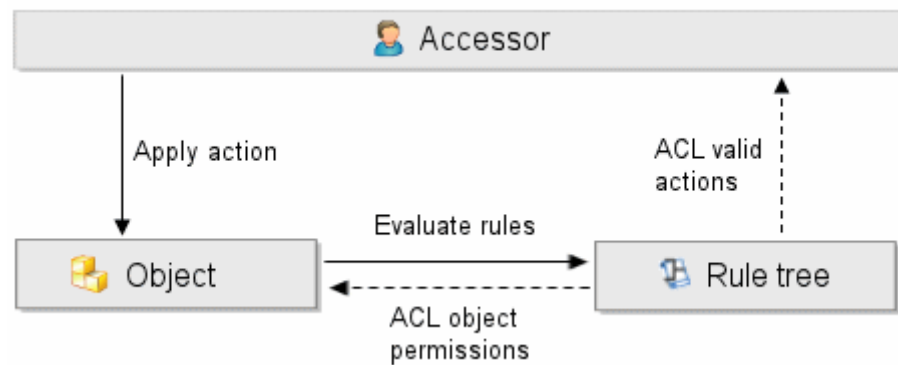
Authorization



Authorization refers to the implementation of rules that control access to specific data stored in the Teamcenter database. Using rules and ACLs in combination with information about users, such as their group and project membership, nationality, and clearance level, enables you to design and implement sophisticated security models to protect your data.

- Authorization to interact with data is controlled by a combination of global rules (*rules-based protection*) and object access control lists (ACLs) applied to specific objects that allow for exceptions to the global rules (*object-based protection*).
- Authorization designates user access to various system resources based on the user's identity. It is typically defined through the use of categories such as groups, roles, and teams.

Rules-based protection

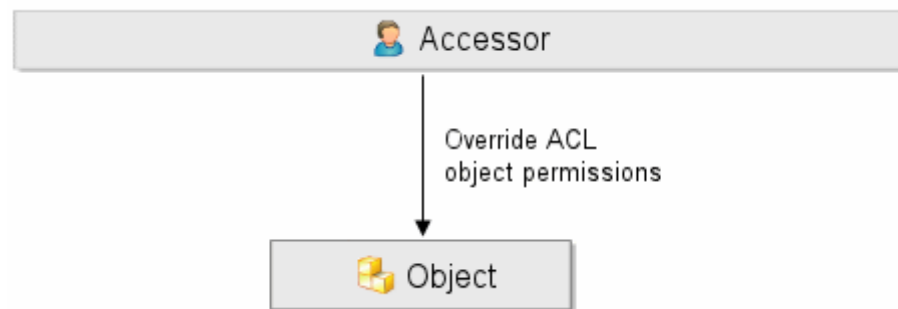


Rules provide security for your Teamcenter data by:

- Controlling access to data on a global basis.
- Determining whether a user has permission to view or perform an action on an object.
- Filtering data according to the properties of the data.
- Granting privileges to the data according to the users' IDs and their session context.

Privileges are directly attached and stored with the objects. Subsequent modifications of privileges may cause intensive processing.

Object-based protection

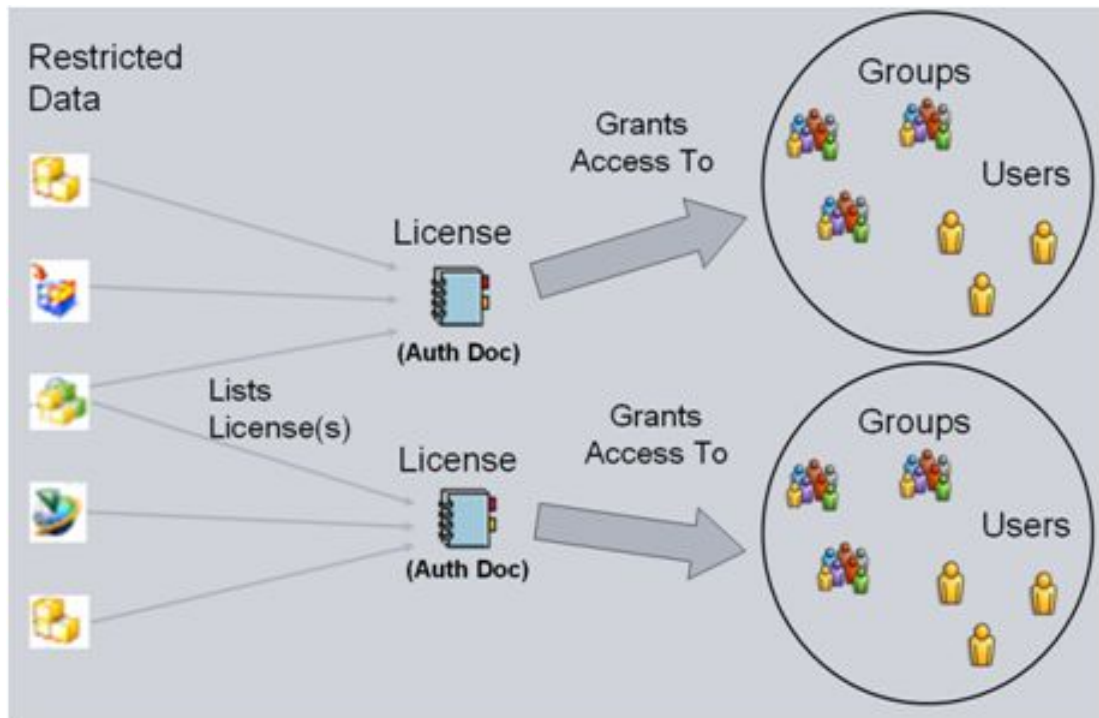


Object-based protection uses access control lists (ACLs) to create exceptions to rules-based protection on an object-by-object basis. The system administrator defines privileges according to a given access strategy of a company.

Teamcenter uses ACLs to determine access to an object. Users with proper permissions can override the rules-based ACL for an object to grant or deny permissions for certain users but only when the rule tree allows.

- Privileges are defined independently within global ACL objects.
- The rule tree allows the assignment of the globally defined ACLs (for runtime evaluation).
- Object-based protection allows high flexibility for subsequent modifications.

Authorized data access



Authorized data access (ADA) is a generic term that applies to the configuration of Teamcenter security for intellectual property (IP) data and for data that is deemed military in nature and is, therefore, subject to International Traffic in Arms Regulations (ITAR) policies.

ADA controls access to classified data using user clearance and authorizing documents (licenses) that grant limited-time access to specific users or groups of users.

ADA facilitates data access control by:

- Identification of users as restricted.
- Identification of data as restricted.
- Authorization of access to restricted data by restricted users using authorizing documents, such as licenses, Technical Assistance Agreement (TAA), non-disclosure agreements, and contracts.

ADA assesses validation during any access attempt by restricted user.

Controlling access to working data

Determining who should have privileges to access working data, and which privileges they should be granted, is an essential component of any Teamcenter security implementation. Privileges are assigned based on your company's business practices and are commonly implemented by determining who has responsibility for the data.

For example, when determining who should have write access, you can grant it to users in the following categories:

- **Owning users**

Granting write access to the owning user indicates that they are ultimately responsible for the content and handling of the data.

- **Owning groups**

Granting write access to the owning group enables a teamwork approach to creating and maintaining data..

- **Project members**

Granting write access based on the projects to which data is assigned enables a teamwork approach to creating and maintaining data and allows the data to be easily assigned to projects, upon which access is then defined. It also provides a mechanism to control access for suppliers.

- **Authorized users for classified data**

Granting read access based on authorized data access concepts enables you to protect intellectual property (IP) and data subject to International Traffic in Arms Regulations (ITAR) policies using combinations of user authorization, object classification, and authorizing documents (IP and ITAR licenses).

Note

Although multiple users can be granted write privileges to data, the data can be modified only by one user at any given time. This behavior is ensured by implicit and explicit checkout.

Controlling access to in-process data

Access privileges to data that is in process (data that is the target in a workflow process) are controlled using the **In Job** rule condition.

Unlike other rule conditions, you do not associate an ACL directly with the **In Job** condition. 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.

Note

ACLs associated directly with the **In Job** condition are ignored when the rule tree is evaluated. Only ACLs associated with the workflow process are used to grant access.

The **EPM-set-rule-based-protection** handler passes information to Access Manager to determine which ACL to use when the task with which the handler is associated is in a current or started state. Workflow ACLs are created in the Workflow Designer application within the context of a specific task and are considered an attribute of the task.

Example

If the handler is associated with the start of a **Review** task, when the task is started, the ACL specified by the handler's **Named_ACL** argument determines the access privileges for the target objects associated with the process.

Activities

In the *Teamcenter security* section, do the following activities:

- Set up ADA and Access Manager rules for ITAR.
- Import new user accounts and add ADA attributes.
- Assign government classification values and associate a license to data objects.
- Prepare to test user access to data objects.
- Test user access to data objects.

Instructor Note:

In these activities, students will practice Teamcenter Unified Security functionality related to Access Manager/ITAR/IP/ADA.

Due to time limitation or students need, you may need to make these optional or let students work on them after the last section is completed.

Note

Limited to most basic ITAR configuration due to time constraints; concepts are similar for IP, etc.

The data a user is allowed to view varies based on their nationality, clearance level, licenses granted, etc.

We will show how the same data would appear differently for different users depending on the security rules in place and their access allowed to these objects.

Appropriate users see appropriate data based on given sample ACLs.

- **usperson** – can view all three objects as expected, but does not have privileges to attach licenses to data objects.
- **foreignpersonwithlicense** – can only view 2 out of 3 objects as expected (access blocked to **ITAR controlled** item)
- **foreignpersonwithnolicense** – can only view 1 out of 3 objects as expected (access blocked to **ITAR controlled** and **ITAR controlled WITH license**) items.

There are a number of basic steps that must be performed when configuring and administering authorized data access for ITAR data.

We will review all these steps, however due to time constraints not all steps will be executed.

Review questions

1. Which of the following is a process of determining whether someone or something is, in fact, who or what it is declared to be?

Select one.

- Authentication
- Authorization
- Object model hierarchy

2. Which of the following refers to the implementation of rules that control access to specific data stored in the Teamcenter database?

Select one.

- Authentication
- Authorization
- Object model hierarchy

3. Authorization to interact with data is controlled by which of the following?

Select all that apply.

- Password protection
- Global rules (rules-based protection) only
- Object access control lists (ACLs) only
- A combination of global rules (rules-based protection) and object access control lists (ACLs) applied to specific objects that allow for exceptions to the global rules (object-based protection)

4. How does Authorized data access (ADA) control access to classified data?

Select all that apply.

- Using user clearance
- By authorizing documents (licenses) that grant limited-time access to specific users or groups of users
- By identification of users and data as restricted

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Authentication
2. Authorization
3. Combination of global rules (rules-based protection) and object access control lists (ACLs) applied to specific objects that allow for exceptions to the global rules (object-based protection).
4. All answers are correct.
 - Using user clearance
 - By authorizing documents (licenses) that grant limited-time access to specific users or groups of users.
 - By identification of users and data as restricted.

Summary

The following topics were taught in this lesson:

- Teamcenter security administration
- Basic Teamcenter security concepts
- Authentication, authorization and authorized data access
- Controlling access to working and in-process data

Instructor Note:

Summary instructor notes.

Lesson

23 *Workflow process modeling*

Purpose

The purpose of this lesson is to define the workflow process.

Objectives

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

- Describe the workflow process.
- Describe Enterprise Process Modeling (EPM).
- Define a process model.
- Import and export process templates.
- Create a quick-release process.
- Release data using the **release_man** utility.
- Configure a baseline process.

Help topics

Additional information for this lesson can be found in:

- [*Workflow Designer Guide*](#)

Instructor Note:

Lesson instructor notes.

Introduction to 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.

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. Additional process requirements, such as quorums and duration times are defined in the template using workflow handlers.

- Workflow Viewer

An end user can use this application to initiate workflow processes.

- 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**.

Workflow process terminology

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.

Basic concepts for using Workflow Designer

There are two types of templates that you can use to create a workflow process:

- Process templates
- Task templates

Workflow process template

A process describes the individual tasks and the task sequence required to model the workflow process. Process templates define a blueprint of a process or task to be performed at your site.

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.

The graphic-oriented Workflow Designer display allows you to easily browse through the process templates and view:

- Task flow
- Task hierarchy
- Task attributes
- Task 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.

Basic tasks using Workflow Designer

To design and maintain workflow tasks, you can perform the following actions:

- Create templates.
- View templates.
- Add tasks to templates.
- Link tasks.
- Modify task behavior.
- Import and export workflow templates.

Working with Workflow Designer

1. To start Workflow Designer, click **Workflow Designer**  in the navigation pane.
2. Click **Browse**  to view process data and the details of the process. You cannot make any modifications in this mode.

Browse mode is the default mode when you first access the Workflow Designer.

3. Click **Edit Mode**  to edit templates.

To use the Workflow Designer in edit mode, you must be a member of the system administration group.

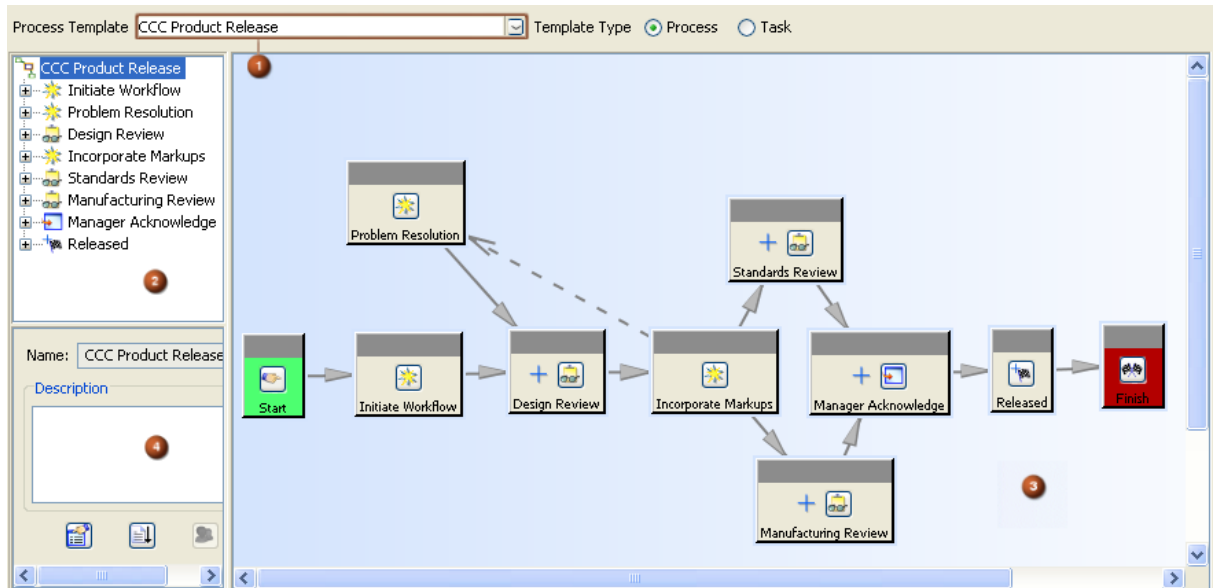
Note

Access may be restricted even if you have administrator privileges.

The **infodba** account is an administrative super-user account and should not be used for production work.

Workflow Designer interface

Workflow Designer uses the standard Teamcenter rich client interface.



- 1 **Process Template box** Lists either all *process* templates or all *task* templates, depending on whether you click the **Process** or **Task** button for the **Template Type**.
- 2 **Process Template tree** Displays all tasks in the selected process template or all subtasks contained within the selected task template.
- 3 **Process Flow pane** Displays a graphical representation of all tasks in the selected process template or of all subtasks within a selected task template.
- 4 **Template Manager pane** Contains elements related to managing the selected process template or task template. The elements displayed depend on the status and configuration of the selected template.

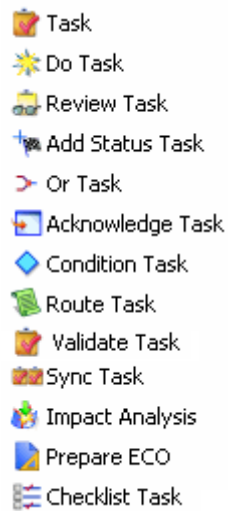
In this example, the template stage is set to **Under Construction**; the template is visible only to users with administrative privileges. When you select this process template, the **Set Stage to Available** check box displays. This check box does not display when the template stage is set to **Available**.

Workflow 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.

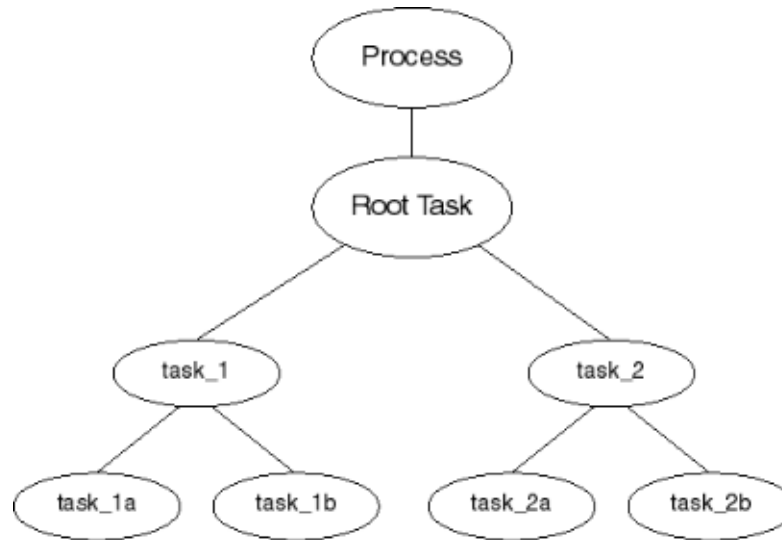
You can create, view, add tasks, modify, and link templates. Additional process requirements, such as quorums and duration times, are defined in the template using *workflow handlers*. *Handlers* are the lowest level building blocks in workflow.

Example tasks include **Task Do**, **Review**, **Add Status**, **Route**, and **Checklist**.



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.

In EPM, each instance of a process uses a process template. This allows each process template to be used as a blueprint for creating multiple processes.

In EPM, tasks have both temporal (time) and hierarchical (structure) relationships. It allows individual tasks to complete sequentially (serially) or asynchronously (in parallel).

You can define:

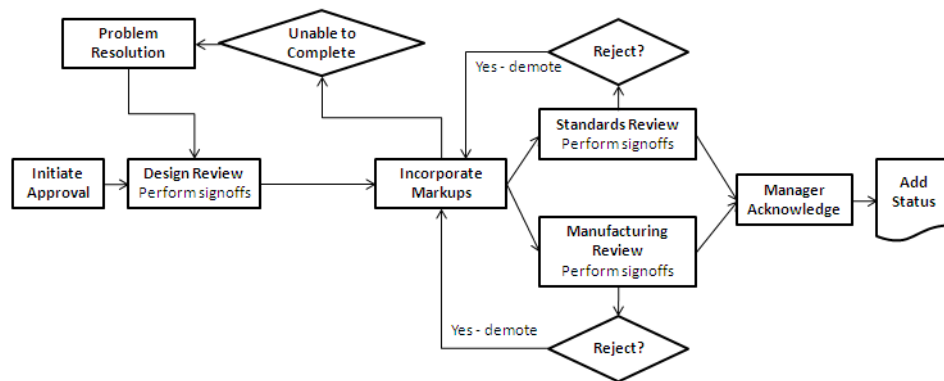
- A specific process by placing workflow and/or change management tasks (**do**, **perform signoff**, **route**, **checklist**, and so on) in the required order of performance.
- Additional process requirements (such as quorums and duration times) in the template, using workflow handlers.

Workflow Designer allows you to create both serial and parallel process templates, and provides you with nine core templates on which you can build new process templates.

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.

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.

Sharing process templates

Importing and exporting process and task templates from the Teamcenter database is useful for transferring workflow templates between different Teamcenter sites.

You can export process and task templates from the Teamcenter database, store them in a single export file you define, in a directory you define. 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.

Note

The AM rules are not exported with the workflow. Export the AM rules file if you modified the rules for the workflow process.

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.

Note

Because the AM rules are not imported with the workflow, import the additional file containing the AM rules modified for the workflow process, if available.

Activity

In the *Workflow process modeling* section, do the following activity:

- Import process templates.

Review questions

1. How do you transfer workflow between different Teamcenter sites?
Select one.
 - 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?
Select one.
 - Importing existing templates into the system is not allowed.
 - Nothing, because the system automatically overwrites the existing templates.
 - Select the **Overwrite Duplicates** option.

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Using workflow import/export functionality.
Help answer: 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.
2. False
Help answer: Choosing **Tools® Export** displays the **Export Workflow Templates** dialog box. You can select one or more template for export.
3. Select the **Overwrite Duplicates** option.
Help answer: If you want the system to overwrite any template of the same name that already exists in the database, choose the **Overwrite Duplicates** option. If you do not choose 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 errors in any case.

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:

- 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
- Create a process based on the **Empty Process** template and add the **Add Status Task** that contains the **create-status** and **add-status** handlers.
- Create a process based on the **Empty Process** template and add the **create-status** and **add-status** handlers.

Create a quick-release process template


1. Choose **File**→**New Root Template**.

The **New Root Template** dialog box is displayed.

2. In the **New Process Template Name** box, type a name for the process template.

Note

The name can be a maximum of 30 characters. Do not use periods or commas.

3. Choose **Process** as the template type.
4. From the **Based On Root Template** list, select **Empty template** to use as the base for the quick-release template.
5. Click **OK** to close the dialog box.
6. Select **Add Status Task** template  on the toolbar.
7. Double-click in the process flow, immediately to the right of the **Start** task.
A **New Add Status Task 1** is created.

Note

Because your new quick-release template includes the **Add Status Task** template, it contains the required **create-status** and **add-status** handlers.

8. Type **Released** in the **Name** box and press Enter.
9. Link the tasks from **Start** to **Finish**.
 - ☐ Place the mouse cursor in the **Start** task and press and hold the left mouse button.
 - ☐ Drag the cursor to the **Released** task.
An arrow appears between the two tasks indicating they are linked.
 - ☐ Repeat the process until all tasks are linked.



Note

A task can be repositioned by clicking the top part of the task and dragging it to a new location.

10. Make the quick-release workflow available to the users.

- ☐ Select the **Set Stage to Available** check box.

The **Stage Change** dialog box appears.

- ☐ Click **Yes**.

The process template is displayed in the **Process Template** list within Workflow Designer and in the **Based On Root Template** list in the **New Root Template** dialog box.

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.

Activities

In the *Workflow process modeling* section, do the following activities:

- Create a quick-release process.
- Test the quick-release process.
- Add release status using `release_man`.

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 becomes read-only.
 - Users cannot access the released data.
2. What is used to release data in a process template?
Select one.
 - 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.
 - 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

Instructor Note:

Answers to review questions

1. The release status and date attributes get values.
The released data become read only.

Help answer: Other facts on releasing data:

- Releasing flat objects, such as datasets or forms, just release the object
- Releasing a folder, just release the folder, but not its contents
- Releasing an item revision, releases the item revision, BOMView revision, and all of the specification relation objects

2. A handler.

Help answer: Releasing data is done through the use of the **add-status** handler in the release process template. You can remove or change the handler to get other desired functionality using action handlers.

add-status handler finds all the status types attached to the root task and attaches them to each target object.

create-status handler attaches the specify status to the root task.

set-status handler applies the appropriate status objects to the processes target objects.

3. True

Help answer: To create a quick-release process you could either create a process based on the **Review Process** template as it has the create/add status handlers already built in, although this template also contains unnecessary handlers. Or you could create a process based on the **Empty Process** template and add the create-status and add-status handlers.

4. A workflow utility to release data in batch mode.

Help answer: The **release_man** utility is used to release objects in batch mode without creating jobs and audit files.

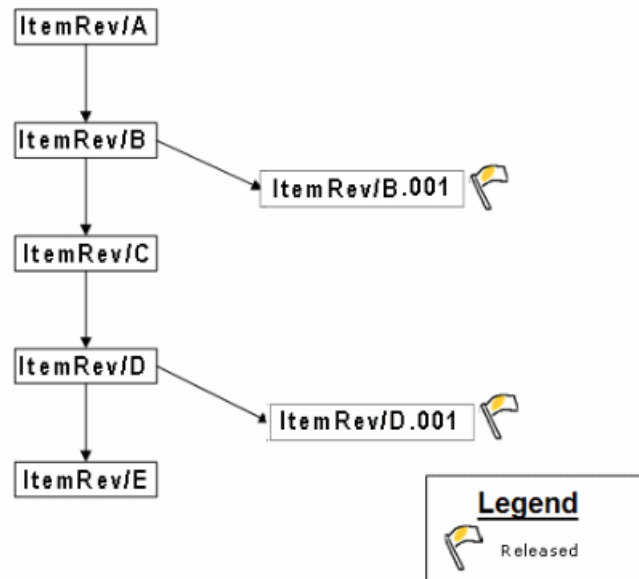
You need to belong to the DBA group to execute the **release_man** utility, a valid status, and a release folder inside the **Home** folder.

5. True

Help answer: The **release_man** utility can also be used to remove or delete the status from objects using the **-unrelease** argument.

Creating a baseline process

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 a 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 activities

In the *Workflow process modeling* section, do the following activities:

- Create and attach a naming rule for a baseline release.
- Create a status option for a baseline process.
- Create an empty process and add handlers.
- Adjust the Teamcenter environment.
- 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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. A snapshot of a work-in-process data.

Product design at a particular stage.

Help answer: The **baseline** function 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.

2. A naming rule.

A quick-release process.

A status option.

Help answer: And you also need a value for the **Baseline_release_procedures** preference to execute the process baseline.

Change Management process model

Change Management is the process of controlling changes to a product's definition and configuration.

Using Change Manager, you can:

- Propose solutions.
- Manage problem items.
- Track affected items and references.
- Edit change attributes.

You need a Teamcenter Change Management author license to use all Change Manager functionality. If you have only an access license, you are limited to the following functionality:

- Creating a problem report (PR).
- Checking in a PR and submitting it to a workflow process.
- Attaching supporting information to the PR.
- Adding business items to the PR **Problem Items** and **Reference Items** folders.
- Searching for and view change objects (PRs, ECRs, and ECNs).
- Receiving and acting on workflow assignments and tasks.

Schedule Manager must be available and licensed to use the work breakdown structure functionality.

Workflow Designer must be available to process a change. Status changes can only occur within a workflow process. Business model state transitions should also be made using a workflow handler.

Change management process

Workflow processes must be created to support change management. Siemens PLM Software does not provide change workflows—you must create your own.

You should define a change process that is flexible enough to impose the appropriate level of rigor and control, based on the level of risk, cost, and the business items impacted by the change. Teamcenter provides a property that directs changes on one of two tracks:

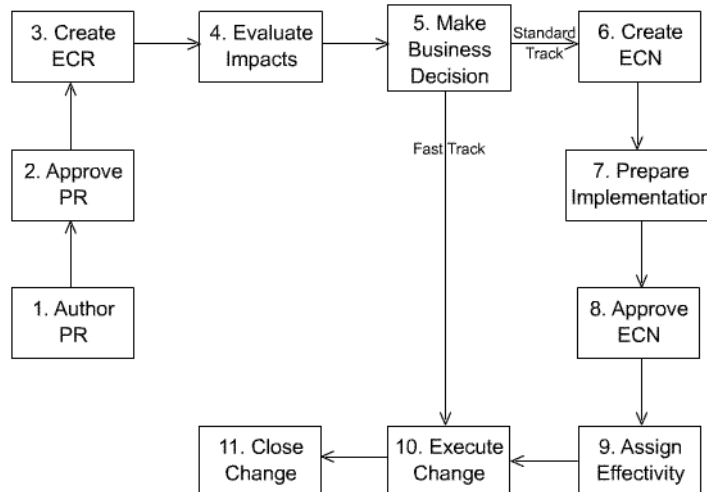
- **Fast track**

The user can plan a change, approve it, and begin execution without the need to be put before a change review board. Fast track changes meet prescribed criteria such as low risk or low cost.

- **Standard track**

A detailed change process that supports problem reports, change requests, change notices, and implementation plans to manage the entire cycle of review, approval, and implementation of a change.

You can manage your changes in the way that works best for you and your processes. This example shows a typical standard track process and notes which steps are not used in a fast track process.



1. Author a problem report.

A requestor creates a problem report to identify a problem or enhancement, provide a preliminary assessment, and show the steps necessary to reproduce the problem.

2. Approve a problem report.

A change specialist or administrator assigns a priority to the problem report and assigns it to an analyst for technical review. The specialist or analyst recommends a disposition, such as **Approved**.

3. Create a change request.

A requestor (who may be the analyst associated with the problem report) creates a change request to address the problem report.

Note

The change request can address more than one problem report.

4. Evaluate the impacts.

Optionally, if a change specialist determines that the impact of the change requires additional evaluation, the administrator assigns the change request to an analyst. The analyst identifies the items impacted by the change, prepares supporting documentation, and prepares a high-level proposal for the actions required to implement the change.

5. Make a business decision.

A change specialist submits the change request to a change review board who decides if the change will be made. The change review board either approves the change request, rejects it, or requires additional investigation. If this is a fast track change, the review board is the owner of the change and the process moves to the execute change step.

6. Create a change notice. (Not used in fast track processes.)

The requestor (who may be the analyst of the change request) either creates a new change notice to address the approved change request or associates the change request with an existing change notice. The change notice addresses the implementation details of the change. It may address multiple change requests. The requestor can delegate responsibility for elaborating the details of the implementation plan.

7. Prepare an implementation plan. (Not used in fast track processes.)

The analyst develops a detailed plan to address the set of approved change requests addressed by the change notice.

8. Approve the change notice. (Not used in fast track processes.)

The change implementation board reviews and approves the plan to address the change.

9. Assign an effectivity. (Not used in fast track processes.)

A change specialist assigns effectivities to the change notice. The effectivities specify the timing of when the change takes effect.

10. Execute the change.

The analyst implements and tracks the detailed plan for addressing the change. A change administrator tracks the implementation progress at a high level.

11. Close the change.

The analyst closes the associated levels of the implementation plan. When all the actions associated with each level of the implementation plan are complete, a change administrator closes the change.

Introduction

Change Manager helps you track changes to a product throughout its life cycle. You propose a change to a product and then manage the entire cycle of review, approval, and implementation of the change. You can articulate the work required to:

- Implement a change.
- Assess its impact on any managed business items such as parts or documents.
- Notify life cycle participants about proposed and authorized changes.
- Track progress and completion of work.
- Compare before and after product configurations.

Change Manager is used by a wide range of people. Anyone who uses Teamcenter can create a problem report (PR) to identify and formally track an issue with your product information. Others who are involved in the change process can review and confirm the problem and provide input into what business objects must be updated to resolve the issue. Members of a change review or change implementation board can review and approve or disapprove the changes.

Change Manager enables an organization to continuously improve its products by building them according to released documents, and then creating, changing, and approving the documents to build in continuous improvement.

Change Manager can be easily configured to use your change process.

Use Change Manager with Workflow Designer to track the evolution of changes through your organization according to a controlled, repeatable process. Schedule Manager provides work breakdown structures that you use to plan and schedule the changes you are making to your product.

Basic concepts for using Change Manager

Change Manager provides a change management foundation as well as a preconfigured change management state model that enforces default business rules. It is a formal change process that lets you manage a consistent, controlled closed-loop process to continuously improve your products. Change Manager is based on the precepts of the CMII closed-loop change model, but it does not fully conform to CMII nor is it CMII certified.

For more information about CMII, see:

<http://www.icmhq.com>

You can propose a change to a product and then manage the entire cycle of investigation, elaboration, review, approval, and implementation of the change. You articulate the work required to implement a change, assess its impact on any managed business items such as parts or documents, notify life cycle participants about proposed or authorized changes, and track progress and completion of the change. This is typically controlled by workflows that flexibly guide the change through the four phases of the change process: authoring, review and approval, execution, and closure.

The change management solution is supported using the following change objects:

- **Problem report (PR)**
- **Enterprise change request (ECR)**
- **Enterprise change notice (ECN)**

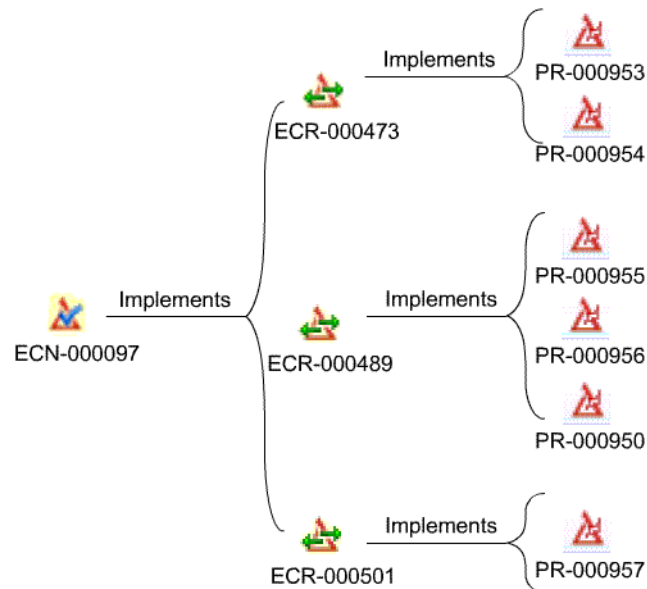
Change objects capture the necessary change information either as part of their properties or through establishing relations with other objects.

Changes are classified to determine processing.

- *Fast track* classification empowers a user to plan a change, approve it, and begin execution without the change being reviewed and approved by a separate review board. Typically, the majority of changes are processed through fast track.
- *Standard track* classification follows a more formal change process including change and implementation review boards.

Change objects

Problem reports (PRs), enterprise change requests (ECRs), and enterprise change notices (ECNs) are objects that address problems or enhancements. The following figure illustrates the relationship between these objects.



Issues identified in one or more PRs can be implemented by a single ECR. Similarly, issues identified in one or more ECRs can be implemented by single ECN. Likewise, a single ECR can be split and implemented by two or more ECNs.

Change object	Description
Problem report (PR)	Initiates a change. A PR clearly defines a problem or an enhancement. The creation of a PR sometimes leads to the creation of an enterprise change request. Creating a PR is an optional step in the change process. Depending on the conventions at your site, you may first identify a problem or enhancement with an enterprise change request, not a PR. A PR may be addressed by multiple ECRs.
Enterprise change request (ECR)	Initiates a proposal that recommends a change and captures business decisions associated with the change. An ECR states the cost estimate and benefits of making a change and provides a recommended action. An ECR is sometimes a response to a PR. A single ECR may logically group and address issues identified in multiple PRs.

Change object	Description
Enterprise change notice (ECN)	Provides a detailed work plan to resolve one or more ECRs or a portion of one ECR. An ECN identifies all items and documents affected by a change and authorizes the actions that address a change.

Workflow handlers designed for the Change Management process

Special workflow handlers are provided with Teamcenter for use in process models designed for the Change Manager 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 Engineering Change (EC) process. This handler may be used several times in a process, depending on the requirements of the EC process.

- **ECM-att-new-status-for-aff-revs**

This handler attaches a separate release status object for each affected item revision of the targeted EC revision. Based on the design of the workflow, all targets of a process share the same release status object, and therefore, share the same effectivity.

- **ECM-copy-end-item-effectivity**

This handler copies effectivity from one affected revision of the target EC revision to all other affected revisions that do not share the same release status. You only need to set the effectivity in one affected revision, then use this handler to copy the effectivity to the release statuses of other revisions. If more than one affected revision contains effectivity information, this handler does not take effect.

- **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.

A revision is considered to be going through competing change if that revision is an affected revision for a non-released engineering change (EC) revision. The body of the e-mail is different for each competing change found. If no subject argument is given, the e-mail is sent with the default subject: **Competing change is being released**.

If no recipient list is supplied, the e-mail is sent to the owner of the competing EC.

- **EPM-attach-related-objects**

This handler attaches the specified related objects of the target objects as target/reference attachments to the process. This handler searches all

target objects, finds the secondary objects with the specified relation and type (if specified), and then adds them as target/reference attachments.

- **EPM-set-property**

Accepts a list of properties and a list of associated values, and uses those values to set the properties on the specified objects.

Review questions

1. Which of the following change objects support the change management solution?

Select all that apply.

- **Problem report** (PR)
 - **Enterprise change request** (ECR)
 - **Engineering Change** (EC)
 - A special workflow template
 - **Enterprise change notice** (ECN)
2. You need a Teamcenter Change Management access license to use all Change Manager functionality.
 - True
 - False
 3. Why is a **EPM-attached-related-objects** handler added to a process template?

Select one.

 - 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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. The change management solution is supported using the following change objects:
 - **Problem report** (PR)

- **Enterprise change request (ECR)**
- **Enterprise change notice (ECN)**

2. False

Help answer: You need a Teamcenter Change Management author license to use all Change Manager functionality.

3. To attach a separate release status object for each affected item revision of the targeted change revision.

Help answer: The **EPM-attached-related-objects** handler attaches the specified related objects of the target objects as target/reference attachments to the process.

Summary

The following topics were taught in this lesson:

- Basic concepts of Workflow
- Basic concepts and tasks for using Workflow Designer
- Import and export process templates
- Release data using a quick-release process
- How to configure a baseline process

Instructor Note:

Summary instructor notes.

Lesson

24 *Workflow process templates*

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.
- Deploy Workflow templates to users and groups.
- Troubleshoot a workflow process.

Help topics

Additional information for this lesson can be found in:

- [*Workflow Designer Guide*](#)

Instructor Note:

Lesson instructor notes.

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.

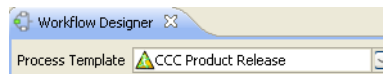
Examples

- Create a process template outlining the process required for a final design review, name the process template to reflect the 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 an **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.





Workflow task templates









The **Task** template is the default template. Use it as a starting point for creating your own custom tasks, such as:


- Tasks to carry your custom forms.
- Other site-specific tasks for users to complete.

The **Task** template is synonymous with the **EPMTask** template.

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	Has two options, if at least one failure path is configured. Complete confirms the completion of a task and triggers the branching to a success path. Unable to Complete indicates the task is unable to complete, for various reasons.
	Review Task	Uses the EPM-hold handler, which stops the task from automatically completing when started. Uses the select-signoff-team and perform-signoff subtasks, each of which has its own dialog box. Wait for Undecided Reviewers is an option that allows the workflow designer user to set the Review task to wait for some, all, or a percentage of reviewers to submit their decisions before completing and following the appropriate path.
	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.

Symbol	Task template	Definition
	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.
	Validate Task	Gives you the ability to respond to errors by providing an alternate path which the process traverses when an error occurs.
	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.
	Impact Analysis Task	Provides an impact analysis for a user to complete for the associated Engineering Change (EC) revision. The task provides Reference , Impact Analysis Form , Viewer , and Task Info tabs. The Impact Analysis Task template is for use in EC processes only. It cannot be used on a Workflow process.
	Prepare ECO Task	Provides EC Requests or EC Orders for a user to complete. The Prepare ECO Task template is for use in EC processes only. It cannot be used on a Workflow process.

Symbol	Task template	Definition
	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>

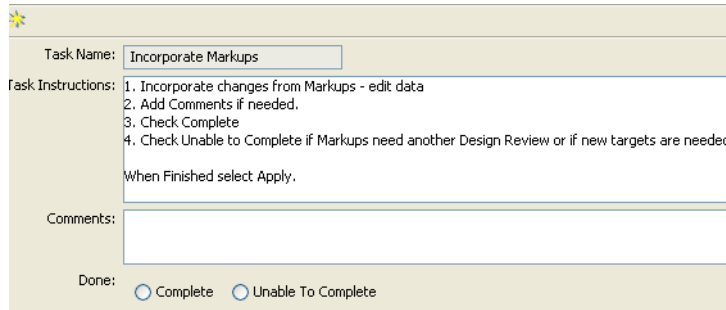
Do task templates

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, a user comment section 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**.

The **Do Task** has two options if at least one failure path is configured:

- **Complete** confirms the completion of a task and triggers the branching to a success path. This is the only option if a failure path is not configured.
- **Unable to Complete** indicates the user is unable to complete the task. It is only available if a failure path is configured.



The screenshot shows a dialog box titled 'Incorporate Markups'. It contains the following fields and controls:

- Task Name:** Incorporated Markups
- Task Instructions:**
 1. Incorporate changes from Markups - edit data
 2. Add Comments if needed.
 3. Check Complete
 4. Check Unable to Complete if Markups need another Design Review or if new targets are needed.

When Finished select Apply.
- Comments:** A text area for user comments.
- Done:** Two radio buttons: ☐ Complete and ☐ Unable To Complete.

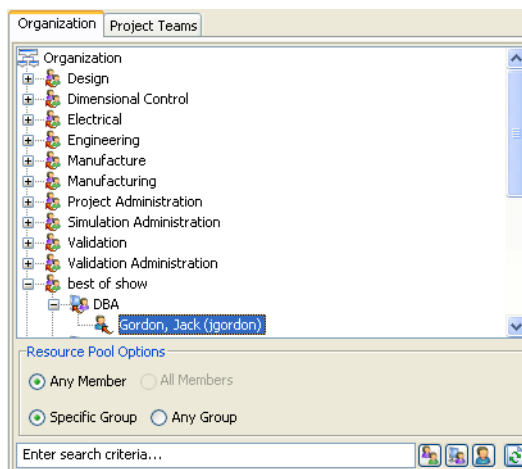
If you require user authentication before a **Do task** can be performed, add the **require-authentication** handler to the **Perform** action of the task. When you implement user authentication for this task, a password box displays below the **Comments** box. Users must type their user password in 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.

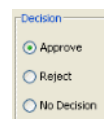
The **Review task template** uses the **select-signoff-team** and **perform-signoff** subtasks, each of which has their own dialog box.

Select-signoff-team and **perform-signoff** use selection functionality from the Teamcenter Organization application. This allows you to search by **Group**, **Role**, or **User** or navigate the Organization tree to find a **Group**, **Role**, or **User**. Participants chosen from **Project Teams** can be individual users or from a resource pool. Only active projects to which the user belongs are shown.



The user that initiates the process 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:

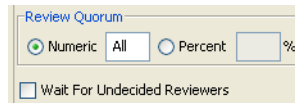
- **Approve**
- **Reject**
- **No Decision**



Signoff team members choose one of these options to perform the signoff.

You can designate the number or percentage of reviewers required for the quorum to be between one and the total number of users required for the selected signoff. The default setting is **Numeric** with the value of **All**. Select

Wait for Undecided Reviewers if you want all of the required users to have a chance to review and comment before the process can be rejected or approved.



Review Quorum

☒ Numeric ☐ Percent %

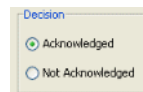
☐ Wait For Undecided Reviewers

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

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. A 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 performs the task in full or delegates the task.

Note

To view and perform tasks that are assigned to resource pools, users must add the **Resource Pool Inbox** to their **My Worklist** tab using the My Teamcenter **Tools® Resource Pool Subscription** menu.

Add Status task template

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
- Icons

- **Handlers**

Handlers are the lowest-level building blocks in EPM. 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
- Add related objects as targets
- Demote a task
- Perform a signoff
- Change a status

- **Workflow ACLs**

Workflow ACLs are created in Workflow Designer within the context of a specific task and are considered an attribute of the task.

Access privileges to data that is the target in a workflow process are controlled using the **In Job** rule condition in the Access Manager tree.

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 targets in the workflow process.

The system uses the **EPM-set-rule-based-protection** handler to determine the appropriate ACL to be applied. The workflow ACL stays current with the targets until another workflow ACL is applied.

Workflow accessors and privileges

In addition to the **In Job** rule condition, the following accessors and privileges are used to control access to in-process data:

Workflow accessors

- **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.

Workflow privileges

- **Promote**

Specifies whether the accessor is authorized to move a task forward in a workflow process.

- **Demote**

Specifies whether the accessor is authorized to move a task backward in a 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**

Use action handlers to 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
- Launching applications.

- **Rule handlers**

Use rule handlers to integrate workflow business rules into EPM processes at the task level. Rule handlers attach conditions to an action.

Many conditions defined by a rule handler are binary (that is, they are either true or false). However, some conditions are neither true nor false.

EPM allows two or more rule handlers to be combined using logical AND/OR conditions. When several rule handlers are combined using a logical OR condition, rule handler quorums specify the number of rule handlers that must return go for the action to complete.

Rule handlers confirm that a defined rule has been satisfied. If the rule is met, the handler returns the **EPM_go** command, allowing the task to continue. If the rule is not met, it returns the **EPM_nogo** command, preventing the task from continuing.

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

Examples 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.

Note

Never add a Change Management handler to a workflow process. CM handlers are designed to be used only in change processes.

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 target objects.

Caution

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-attach-related-objects**

Attaches the specified related objects of the target objects as target/reference attachments to the process. This handler searches all target objects, finds the secondary objects with the specified relation and type (if specified), then adds them as target/reference attachments.

- **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

- **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.

Activities

In the *Workflow process templates* section, do the following activities:

- Create a multitask process.
- Define reviewer profiles.
- Create a status configuration.
- Add a resource pool to the signoff task.
- Add rules based access for Workflow.
- Use task handlers.
- Test the multitask process template

Review questions

1. What is a workflow?
Select one.
 - 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.
 - **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.
 - Attachments
 - Conditions
 - Tasks
5. What is a task template?
Select all that.
 - 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.

- 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.
- 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.
- **check-status**
 - **disallow-adding-targets**
 - **require-authentication**

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Workflow is the automation of a business process.

Help answer: 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.

2. Process and task templates.

Help answer: A *process template* is a blueprint of a workflow 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.

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.

3. True

Help answer: Workflow ACLs are created in the **Workflow Designer** within the context of a specific task and are considered an attribute of the task.

4. Tasks

Help answer: A *process* is made up of tasks. Some tasks may be subtasks associated with a main task.

5. A *task template* is a blueprint of a Workflow task.

A *task* is the fundamental building block used to construct a process.

Help answer: Each task defines a set of actions, rules, and resources used to accomplish that task.

6. In a **Review** and **Acknowledge** tasks templates.

Help answer: 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.

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.

7. A group, role in a group, or a role which can be assigned tasks the same way an individual user is assigned tasks.

Help answer: A **Resource Pool** is a group, role in a group, or a role which can be assigned tasks the same way an individual user is assigned tasks. Resource Pools are 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.

8. True

Help answer: You can set requirements and restrictions on a task using Attributes. Task attributes are: Named ACL, template name, signoff quorum, and release status.

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, and change a status.

9. False

Help answer: 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.

10. **require-authentication**

Help answer: Action handler to display a password box in the perform dialog box or panel of the task within which it has been placed.

Adding secure tasks

To secure a task you use the **require-authentication** action handler. A password box appears and the user must type the logon password to complete the task.

Place the **require-authentication** handler on the **Perform** action of the following tasks:

- **Do** tasks
- **perform-signoff** task
- **Condition** tasks

Note

When working with a **Route** task, place the **require-authentication** handler on the **Perform** action of the **perform-signoff** subtask of either the **Review** or **Acknowledge** tasks.

Activity

In the *Workflow process templates* section, do the following activity:

- Secure workflow tasks.

Review questions

1. How do you add security to a task?

Select one.

- Add the **disallow-adding-targets** rule handler.
- Add the **require-authentication** action handler.
- It is not possible to secure a task.

2. Types of task that you can secure are _____.

Select one.

- **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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. Adding the **require-authentication** action handler.

Help answer: When the **require-authentication** handler is added, a password box appears in the perform dialog box or panel of the task within which it has been placed.

2. **Review** and **Acknowledge** tasks.

Help answer: Place on the Perform action folder of the following tasks: Do, Route, Condition (manual only) and the perform-signoffs subtasks.

3. False

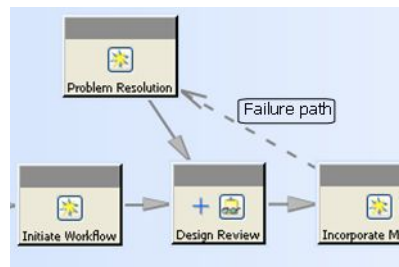
Help answer: The **require-authentication** handler needs to be placed in the **Perform** action folder.

Failure path

When creating a workflow, each path is configured as either a *success path* or a *failure path*.

A *failure path* gives an alternate course that a workflow process may follow in any of the following scenarios:

- A task is rejected.
- The user determines that the task cannot be completed.
- There is an error.



A task follows the appropriate path based on the task's outcome.

- **Failure path**

Must be configured into the process template using Workflow Designer during creation.

- **Success path**

Traversed when a task's state transitions to **Complete** or when a task is promoted and it transitions to a **Skipped** state.

A task completes upon the successful execution of the task's handlers on the **Complete** action.

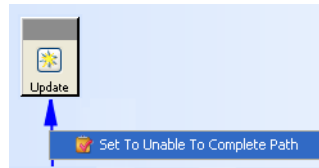
- **Backward branching**

Allows a path to be routed backward to some previous task in the process flow, including the **Start** node. It allows the re-execution of a task with a **Complete** or **Skipped** task state.

Both success and failure paths are capable of branching in a backward direction.

Create failure paths

To create a failure path, right-click an existing success path and select the appropriate failure option.



Failure paths options display differently for different tasks.

Task	Failure option
Do task	Set to Unable to Complete
Review task	Set to Reject
Route task	Set to Reject
Condition task	Set to Unable to Complete
Validate task	Set to Error Path
EPM task	Set to Unable to Complete
PrepareECO task	Set to Unable to Complete
Checklist task	Set to Unable to Complete
ImpactAnalysis task	Set to Unable to Complete

Activities

In the *Workflow process templates* section, do the following activities:

- Success and failure paths
- (Optional) Test the success and failure paths.

Adding Condition tasks

Use the **Condition Task** template to branch your process according to defined criteria. Because this task template is used to branch process flow, you must always create at least two paths branching off from the task. The paths can be either success paths, failure paths, or a combination of the two.

- Success paths can be either *true* paths, *false* paths, or paths with a *customized* result.
- Failure paths can only be generated from manual **Condition** tasks. They allow an alternate course when a specified task is rejected, a user determines the path cannot be completed, or an error occurs.

Tip

If you use a **Condition** task to branch your process, you must use one or more **Or** tasks later in the process to resolve the paths into a single path.

The system determines which of the branches flowing from a **Condition** task to perform based on the *task result*. The task result is stored in the **Condition** task. The successor tasks have a handler configured with a value that may match the task result. After the task result is set, the successor tasks are examined and any successor tasks containing a value matching the task result are started.

Use any of the following methods to set the task results:

- Create a query against the target (automatic only).
- Create a query against the task (automatic only).
- Configure the task result from the manual **Condition** task's dialog box.

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.

Task	Description
Automatic Condition task	<p>Add an action handler that sets the task's result to true, false, or a customized value.</p> <p>The simplest way to achieve this is to use the task template's interface to define a condition query at design time; this automatically inserts the action handler. Alternatively, you can create a custom action handler that uses ITK to verify criteria.</p>
Manual Condition task	<p>During design, you do not define a query or add an action handler to the task template.</p> <p>Because no query is defined and no action handler is configured to set the task result, when the process is run, the end user must manually indicate a value using an interactive dialog box. The value chosen by the end user is used to set the task result.</p>

Set Condition task flow paths

Because **Condition** tasks are used to branch your process according to defined criteria, you must always create at least two paths branching off from the task. The paths can be either success paths, failure paths, or a combination of the two.

To draw and configure success flow paths from a **Condition** task:

1. Click **Edit Mode** to open the process template for edit.
2. Create one or more tasks to succeed the **Condition** task.
3. Select the **Condition** task, placing the cursor in the body of the task (not the blue bar at the top). Draw a line from the **Condition** task to the succeeding task by dragging the cursor to the succeeding task.

A blue line displays between the two tasks.

4. Right-click the line and select the desired path type.

- **Set Path to True Path**

Creates a forward-branching path.

Creating this path automatically places a rule handler on the **Condition** task to check the condition of the specified target. When the condition is **True**, the process proceeds along this path.

- **Set Path to False Path**

Creates a forward-branching path.

Creating this path automatically places a rule handler on the **Condition** task to check the condition of the specified target. When the condition is **False**, the process proceeds along this path.

- **Set Custom Result...**

Allows you to define a custom task result. Enter any string to define the task result.

For example, you could enter **Production** to indicate the process flowing into a production-ready branch.

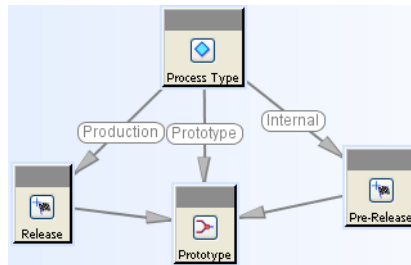
Note

If you select this option and want the **Condition** task to be automatically processed, you must ensure the task result is sent to the **Condition** task. You can do this either by writing custom code or using the **EPM-set-task-result-to-property** handler. Custom conditions can also appear as Manual condition options and appear as buttons in the **Condition** dialog box.

5. If you selected a *true* or *false* path, the flow path displays **True** or **False**, respectively.

If you defined a custom result, the flow path displays the string you entered.

For example, the flow path displays **Production**, **Prototype** or **Pre-Release** paths.



Create as many paths off of the **Condition** task as required for your process.

Example

After creating a production-ready branch, you could create **Design** and **Release** branches by creating additional succeeding tasks and creating additional customized flow lines from the **Condition** task.

Optional activities

In the *Workflow process templates* section, do the following activities:

- Configure a manual **Condition** task.
- Configure an automatic **Condition** task.
- Test the manual and automatic **Condition** tasks.

Review questions

1. What are the values of the **result** attribute in a **Condition** task?
Select one.
 - **True** and **False**
 - **True, False, and Unset**
 - **Set** and **Unset**
2. How many types of **Condition** tasks are available?
Select one.
 - **Automatic** and **Manual**
 - **Automatic** only
 - **Do, Review, and Acknowledge**
3. How do you configure an automatic **Condition** task?
Select all that apply.
 - Add a **Review** task
 - Use a condition query
 - Use 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

Instructor Note:

Activity instructor notes.

Answers to review questions

1. **True, False, and Unset**

Help answer: According to defined criteria. **Condition** tasks have a result attribute than you can set to **True, False, or Unset**.

2. **Automatic** and **Manual**

Help answer:

- **Automatic condition task** uses the task template's interface to define a condition query; this automatically inserts the action handler.
- **Manual condition task** does not define a query and does not add an action handler to the task template.

3. Using a condition query.

Using an action handler that sets the task's result to true or false.

Help answer: 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.

4. To resolve the parallel paths into a single path when using a condition task.

To reconcile task branches.

Help answer: The **Or Task** continues the workflow process when any one of its multiple task predecessors is completed or promoted. There is no limit to the number of predecessor an **Or Task** may have.

Troubleshooting your workflow

As you work through a workflow process in Workflow Viewer, the task automatically transitions from one state to another according to the actions defined in the task.

There are some situations, however, when it is necessary to override the task's defined actions. Following are some examples:

- A task is demoted and the process moves backward to the preceding task, but the preceding task does not have a demote handler.

If the preceding task has a demote handler, it is automatically initiated. But if the preceding task does not have a demote handler, the task must be initiated manually. Therefore, the responsible party or a privileged user must manually override the preceding task's defined action and change the task state to **Started**.

- A task is waiting for a signoff but is complete.

The user must manually override the task's signoff and change the task state to **Completed**.

- A process has an error or is suspended and must be canceled.

The user must abort the process to cancel the process and exit without completion.

Perform actions against a workflow process

Workflow actions transition a task from one state to another with the ultimate goal to reach the completed state.

Action	Beginning state	Ending state	Description
Suspend	Any state	Suspended	Put a task on hold.
Resume	Suspended	Any state	Start the task after being suspended.
Undo	Started	Pending	Stop a task.
	Completed		The Undo action appears only on the My Teamcenter Action menu.
	Skipped		
Demote	Started	Pending	Stop a task and change the target object state to a previous level, for example, from Pending to Started . The Demote action only appears on the Workflow Viewer Action menu.
Promote	Started	Skipped	Skip the current task and start the next task in the process.
	Suspended		
Start	Pending	Started	Begin work on a task.
Perform	Any state	No change	View the Perform dialog box to see detailed instructions.
Complete	Started	Completed	Finish the task.
Assign	Unassigned	Pending	Assign or reassign a task to a user. When an Assign action is applied to an unassigned task, the resulting state is Pending .
Abort	Any state	Aborted	Cancel a workflow process and exit without process completion. The Abort action appears only on the Workflow Viewer Action menu.

Cancel a workflow process

To cancel a workflow process and exit without process completion, use the **Action® Abort** command in Workflow Viewer.

1. Select the workflow to cancel in Workflow Viewer.
2. Choose **Actions→Abort**.

The **Abort Action Comments** dialog box appears.

3. Type your comments in the text box.
4. Click **OK** to cancel the workflow.

Resume a suspended task

When a suspended task is resumed, it is restored to the state it was in prior to being suspended.

Note

You must be the responsible party or a privileged user to resume a suspended task.

1. Select the suspended task you want to resume.
2. Choose **Actions**→**Resume**.

The **Resume Action Comments** dialog box appears.

3. Type your comments in the text box.
4. Click **OK**.

The task moves to the state it was in prior to being suspended.

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 **File® New® Process**.

Using the template filter

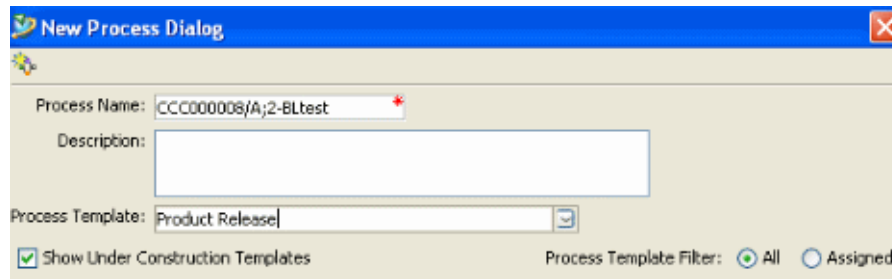
Process templates are assigned by using **Preferences** for the specified group.

In the Workflow Designer, choose **Edit® Template Filter** and then select templates for user groups and object types.

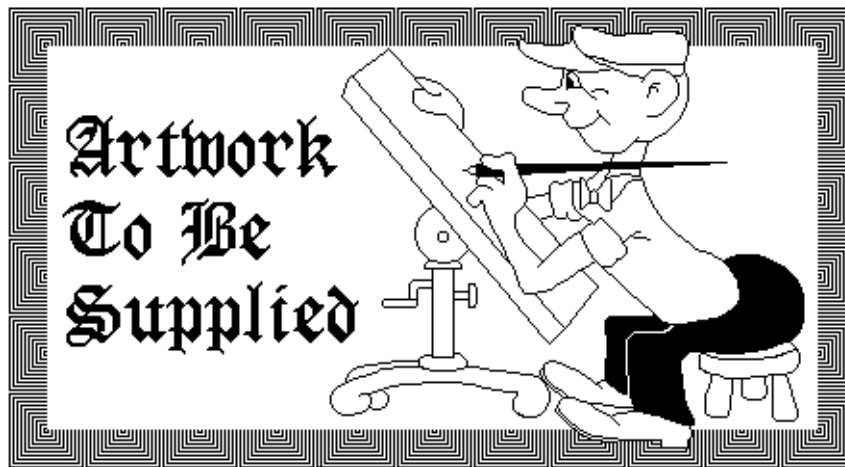
Blocking users from choosing from all process templates

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.

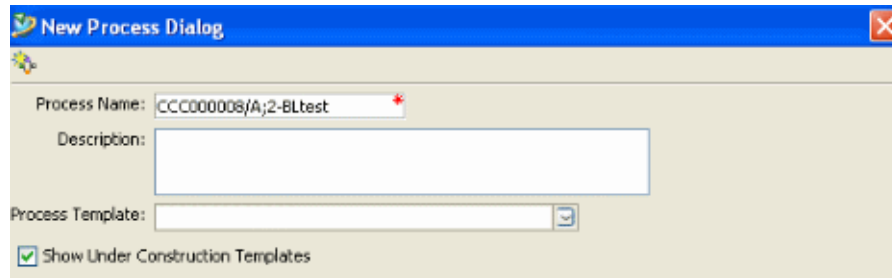


Setting **CR_allow_alternate_procedures** to **Assigned**, allows the **Assigned** option in the **Process Template Filter** to be selected.



- Setting **CR_allow_alternate_procedures** to **none**, removes the **All** and **Assigned** options from the **New Process Dialog** box.

The dialog box is populated with all assigned templates. If no assigned templates are defined for the logged on group and role, the template section of the dialog box is empty.



New Process Dialog

Process Name: CCC000008/A;2-BLtest *

Description:

Process Template:

☒ Show Under Construction Templates

Activity

In the *Workflow process templates* section, do the following activity:

- Assign a default process template.

Review question

1. How do you assign process templates to a group of users?

Select all that apply.

- Select templates for user groups and object types.
- Set the **CR_allow_alternate_procedures** preference to none.
- Use the **Template Filter** function.

Instructor Note:

Activity instructor notes.

Answers to review questions

1. All answers are correct.

Help answer: First, use **Template Filter** in the Workflow Designer to select templates for user groups and object types. Then set the **CR_allow_alternate_procedures** preference to **none** to remove the **All** and **Assigned** options from the **New Process Dialog** in My Teamcenter.

Summary

The following topics were taught in this lesson:

- Create multiple task Workflow process templates.
- Add security to tasks in process templates.
- Add conditional tasks to Workflow process templates.
- Assign templates to users and groups.
- Troubleshoot a workflow process.

Instructor Note:

Summary instructor notes.

Lesson

25 Course summary

During this course, you met the course objectives by accomplishing the following:

- You know what *administrative tasks* are done using the Business Modeler IDE and rich client interfaces.
- You configured Teamcenter using the Business Modeler IDE.
- You executed the basic Business Modeler IDE process.
- You created *business objects* and *classes*.
- To modified *properties* and *attributes*.
- You viewed the data model graphically.
- You created and attached a *list of values (LOV)* to a property.
- You created and used configuration *options*.
- You created and applied *rules* to *business objects*.
- You understand how install, deploy and package *templates*.
- You imported a data model file.
- You created an *organization*.
- You created *saved queries*.
- You configured the working environment by managing *preferences*.
- To imported and exported data using *utilities*.
- You created *report definitions*.
- You configured *access permissions*.
- You configured *projects*.

- You created *workflow process templates*.

Instructor Note:

Review the course accomplishments with the students and describe how the objectives have been met.

Appendix

A Appendix: Utilities

Purpose

The purpose ... is to ... with Teamcenter.

Objectives

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

- Understand .

Help topics

Additional information for this lesson can be found in:

- Put a URL here.

Instructor Note:

Lesson instructor notes.

clearlocks

Clears dead process locks from the database. Dead process locks typically occur when a TeamcenterTeamcenter Express session terminates abnormally. Process locks are set on an object when it is being modified or deleted. If a TeamcenterTeamcenter Express session does not terminate gracefully (by logging out), these locks can remain in place.

Dead process locks (locks held by dead sessions) can cause diverse problems that are often difficult to diagnose, and TeamcenterTeamcenter Express applications make every effort to eliminate or otherwise avoid them. Nevertheless, there are occasions when such dead process locks must be explicitly removed from the database, and the **clearlocks** utility is used for this purpose.

Note

To use the **-assert_dead** or **-assert_all_dead** options, you must specify the administrator's user name, password, and group.

The **clearlocks** utility can only obtain general information about the processes in the lock table. Normally, the PID is pulled from the table and a kill is sent to the operating system. If the PID exists, the **Alive** count is incremented. If the PID does not exist, the **Dead** count is incremented. A **Remote PID** count indicates the process was started from a node other than the one that was used to run clearlocks. On some platforms, for example, Solaris, the kill returns a security violation and no specific information about the PID. If this occurs, the **Other** count is incremented.

SYNTAX

```
clearlocks [-verbose] [-one_pass] [-retry time] [-node_names]
[-assert_dead -u=user-name -p=password -g=group-name node-name |
-assert_all_dead -u=user-name -p=password -g=group-name] [-h]
```

Caution

The **clearlocks** utility can be run with active TeamcenterTeamcenter Express sessions, provided that the **-assert_dead** or **-assert_all_dead** arguments are not used. By default, the **clearlocks** utility discriminates between valid and dead process locks; the **-assert_dead** and **-assert_all_dead** arguments defeat this feature.

ARGUMENTS

-verbose

Displays a summary of processes and states (dead, alive, and unknown). Locks associated with dead processes are cleared by the **clearlocks** utility, live processes are not cleared, and the unknown processes are all other processes.

-node_names

Lists nodes upon which the known processes exist.

-one_pass

Executes the utility once and stops. This is the default if no other arguments are supplied. Opposite of the **-retry** argument.

-retry

Continuously executes the utility according to the time, specified in seconds, before the next execution. Opposite of **-one_pass** argument.

-assert_dead

Asserts that all processes on a particular node are dead and clears all process locks held by sessions running on that node with the exception of Multi-Site Collaboration transfer locks. If any of those sessions are alive and in use, the locks held by those sessions are compromised.

Note

To use this argument, you must enter the node name and the administrator's user name, password, and group. To clear Multi-Site Collaboration transfer locks, use the **export_recovery** utility.

-assert_all_dead

Asserts that all processes in the database are dead and clears all process locks with the exception of Multi-Site Collaboration transfer locks. If any of those sessions are alive and in use, the locks held by those sessions will be compromised. Additionally, this option performs a complete cleanup of the database lock tables and reports on the sessions that were asserted to be dead.

Note

To use this argument, you must enter the administrator's user name, password, and group. To clear Multi-Site Collaboration transfer locks, use the **export_recovery** utility.

-h

Displays help for this utility.

ENVIRONMENT

As specified in ***Unsatisfied xref title***.

FILES

As specified in *Basic concepts about Project*.

RESTRICTIONS

- Do not run the **clearlocks** utility with the **-assert_dead** or **-assert_all_dead** arguments if there are any active Teamcenter Teamcenter Express sessions running. Any locks held by

active sessions will be lost and these sessions can then potentially modify data for which they no longer hold modify locks.

- The **-assert_dead** and **-assert_all_dead** arguments are powerful and potentially destructive. Therefore, these arguments should only be used to clear process locks that cannot be cleared otherwise. For this reason, you must enter the administrator's user name, password, and group when using these arguments.

- The **clearlocks** utility cannot clear the **Transfer** lock type, only the **Modify** lock. To clear **Transfer** locks, use the **export_recovery** utility. This behavior is intended to prevent cases where objects that are being transferred are forcibly unlocked and thereby exposing them to the possibility of being modified when their ownership is being transferred.
- When running Clearlocks with the **assert_all_dead** or **assert_dead** option, you may see the message:

```
Notice: There are transfer locks detected indicating active
Multi-Site transfer transactions. All transfers need to complete
before the upgrade can safely continue. Ensure that
ensure_site_consistency is successfully executed for any identified
objects before running Clearlocks.
```

This message also appears when upgrading a database to a new release if there are existing transfer locks in the database.

EXAMPLES

- To clear process locks for dead sessions, enter the following command from that node:

```
$TC_ROOT/bin/clearlocks
```

- To obtain a list of all network nodes which have process locks set on the database, enter the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

- To run the **clearlocks** utility every four hours, enter the following command:

```
$TC_ROOT/bin/clearlocks -retry 14400
```

- To clear all process locks (active and dead) on a single network node, in this example **ntssun9**, enter the following command:

```
$TC_ROOT/bin/clearlocks -assert_dead infodba infodba dba ntssun9
```

In this example, **infodba** is the administrator's user name and password, and **dba** is the administrator's group.

- To clear all process locks (active and dead) on all nodes, enter the following command:

```
$TC_ROOT/bin/clearlocks -assert_all_dead infodba dba dba
```

In this example, **infodba** is the administrator's user name and password, and **dba** is the administrator's group.

- The following is an example of a line message (report) produced by **clearlocks -verbose**:

```
Processes: 7, Alive: 1, Dead: 6, Remote: 0, Other: 0
```

**CLEARING PROCESS
LOCKS**

Perform the following steps to clear dead process locks using the **clearlocks** utility.

1. Ensure that all TeamcenterTeamcenter Express and Teamcenter Integration for NX users are logged out of the system.

When all users are logged out, all valid process locks are cleared.

2. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

The system displays a report listing network nodes that still have process locks set against the database. Because all users are logged off, these locks are dead and can be cleared.

3. Run the following command:

```
$TC_ROOT/bin/clearlocks
```

4. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

The system displays a report listing network nodes that still have process locks set against the database. Because all users are logged off, these locks are dead and can be cleared.

Any network nodes listed in this second report will require running the **clearlocks** utility with the **-assert_dead** argument to clear the difficult process locks.

5. Run the following command to clear locks held by the session of the specified nodes:

```
$TC_ROOT/bin/clearlocks -assert_dead node-name1  
node-name2 node-name3...
```

node-name is a network node listed in the report.

6. Create a report of all remaining process locks by entering the following command:

```
$TC_ROOT/bin/clearlocks -node_names
```

This report should be clean (empty). If there are any nodes listed in this report, contact the Siemens PLM Software Global Technical Access Center (GTAC) for assistance.

Utility import_file

Depending on .

When ... occurs:

- The
- The

Instructor Note:

Topic instructor notes.

Utility import_file

Depending on .

When ... occurs:

- The
- The

Instructor Note:

Topic instructor notes.

Utility xxx

Depending on .

When ... occurs:

- The
- The

Instructor Note:

Topic instructor notes.

Appendix

B Appendix: Data model reference

Purpose

The purpose ... is to ... with Teamcenter.

Objectives

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

- Understand .

Help topics

Additional information for this lesson can be found in:

- Put a URL here.

Instructor Note:

Lesson instructor notes.

Subtopic

General

REVIEW NOTE

Issue: DK: Reference the ...

Instructor Note:

Topic instructor notes.

Subtopic

General

REVIEW NOTE

Issue: DK: Samples

Instructor Note:

Topic instructor notes.

Index

A

- Access control lists 20-9
 - Bypass 20-23
 - Item Revs 20-24
 - Items 20-23
 - Vault 20-23
 - Working 20-24
- Access Manager
 - Add rule 20-29
 - Basic concepts 20-4
 - Basic tasks 20-5
 - Export 20-32–20-33
 - Group security 20-42
 - Import 20-32–20-33
 - Interface 20-3
 - Purpose 20-2
- Accessors
 - Approver 24-14
 - Approver (Group) 24-14
 - Approver (RIG) 24-14
 - Group administrator 20-22
 - Owner (owning user) 20-22
 - Owning group 20-22
 - Responsible Party 24-14
 - System administrator 20-22
 - Task Owner 24-14
 - Task Owning Group 24-14
 - World 20-22
- Account
 - Generation 1-27, 15-60
- Accounts
 - infodba 15-20
 - Organization structure 15-24
 - System administration . . . 15-19, 15-21
- ACL
 - Adding entries 20-30
 - Creating 20-30
- Action handlers 24-15, 24-17
- Activating
 - Group members 15-55
 - User accounts 15-51
- Active extension files 6-13
- Adding
 - Attributes 7-16
 - Change options 12-15
 - Classes 7-14
 - Condition tasks 24-30
 - Dataset business objects 10-6
 - Existing users to a group/role . . 15-46
 - Form business objects 7-4
 - Item business objects 6-7
 - Lists of values (LOVs) 9-5
 - Naming rules 13-15–13-16
 - New roles to a group 15-41
 - New users to a group/role 15-43
 - Notes 12-4
 - Persistent properties 6-18–6-19
 - Projects 3-2
 - Status objects 12-7
 - Tool objects 10-4
 - Units of measure 12-10
- Adding rules 20-29
- Administrative reports
 - Admin-Employee Information . . . 18-8
 - Admin-Group/Role
 - Membership 18-8
 - Object Ownership 18-8
 - Objects by Status 18-8
- AM Rules tab 21-5
- Application banner 1-12
- Application pane 1-13
 - Rich client 1-13
- Applications folder 3-14
- Approver 23-3
- Approver (Group) accessor 24-14
- Approver (RIG) accessor 24-14

- Approver accessor 24-14
 - Architecture 1-30
 - Array column 5-36
 - Array Length column 5-36
 - Assign to project privilege 20-20
 - Assigning default process templates 24-41
 - Attach
 - Naming rules 13-15
 - Attaching
 - LOVs 9-10
 - Naming rules 13-19
 - Attachment 23-3
 - Reference object 23-3
 - Target object 23-3
 - Attribute Name column 5-35
 - Attribute Selection pane 17-3
 - Attributes 24-13
 - Adding to classes 7-16
 - Data Model 18-10
 - Introduction 5-6
 - Reference 5-31
 - Table 5-35
 - Authentication 15-11, 22-6
 - Author 15-36
 - Authorization 22-7
 - Basic tasks 15-68
 - Authorization interface 15-65
 - Authorization rules
 - System level 15-67
 - Authorized data access 15-17, 22-10
 - Authorized data access licenses
 - Creating 15-39
 - autoAssignToProject
 - extension 21-15–21-16
 - Automatically assigning data to projects
 - autoAssignToProject
 - extension 21-15
 - Eligible data types 21-15
- B**
- Back and forward buttons 1-12
 - Baseline
 - Create 19-65
 - Dry run 19-68
 - Finding folder 19-67
 - Naming rules 19-64
 - Baseline folder 19-67
 - Baseline process
 - Create 23-27
 - Baseline, purpose 19-4
 - Baselines 19-62
 - Basic Access Manager concepts 20-4
 - Basic concepts 15-66, 22-4
 - Business Modeler IDE 3-8
 - Basic concepts about Report
 - Builder 18-5
 - Basic tasks 15-68
 - bmide_postupgradetotc utility 4-17
 - Browse mode 23-4, 23-6
 - Browsing
 - Business objects 3-12
 - Classes 3-13
 - Business Modeler IDE
 - Basic process 3-3
 - Concepts 3-8
 - Configuring 3-18
 - Installing 3-26–3-27
 - Perspectives 3-10
 - Preferences 3-18
 - Prerequisites 3-25
 - Purpose 3-7
 - Server profile 3-21, 3-23, 6-23
 - Start 3-28–3-29
 - Tasks 6-12
 - User interface 1-11, 3-9
 - Using 1-8
 - Views 3-11
 - Business Modeler IDE template
 - projects 3-18
 - Business object display rules
 - Adding 13-5
 - Reference 13-4
 - Business objects 5-14
 - Creating 6-14, 6-17
 - Dataset 10-6
 - Form 7-3–7-4, 7-7
 - In general 1-8
 - Item 5-17, 6-7
 - Properties 5-7
 - Relationships 13-7

- Search 5-9
- View 3-12
- Bypass switch 15-21
- C**
- Candidate Key column 5-37
- Cascading LOVs 9-18–9-19
- Change
 - Adding 12-15
 - Reference 12-12
- Change Management 23-30
- Change Manager workflow handlers 23-38
- Change notices, *see* Enterprise change notices
- Change objects 23-30
- Change ownership privilege 20-19
- Change privilege 20-19
- Change requests, *see* Enterprise change requests
- Changing user status 15-48
- CICO privilege 20-21
- Class Attribute Selection dialog box 17-24
- Class/Attribute Selection dialog box 17-3
- Classes 5-14
 - Adding 7-14
 - Applications 3-14
 - Attributes table 5-35
 - In general 1-8
 - View 3-13
- clearlocks 1-24
- clearlocks utility A-2
- Clipboard
 - Description 1-13
- Closure rules 18-12
- CMII change model 23-35
- Compound properties
 - Adding 6-26
- Compound property rules 5-38, 6-25
- Concepts
 - Custom reports 18-6
 - Item reports 18-6
 - Summary reports 18-6
- Condition tasks, adding 24-30
- Conditions
 - In Job 22-12
- Configuration items 19-43
 - Creating 19-45
 - Define effectivity mapping 19-49
- Configuring supplier security for external data 20-44
- Configuring supplier security for internal data 20-43
- Configuring the Business Modeler IDE 3-18
- Console view 3-16
- Constants
 - Framework 5-45
 - Precedence 5-46
 - Property 5-47, 5-49, 6-20
 - Working with 5-44
- Consumer 15-36
- Controlling volumes' access 15-29
- COTS (commercial off-the-shelf) 3-6, 4-14
- Counters in naming rules 13-22
- Create a new Teamcenter UML diagram wizard 5-26
- Create failure path 24-28
- CreateDescriptor tab 3-38
- Creating
 - Attributes 7-16
 - Authorized data access licenses 15-39
 - Business objects 6-14, 6-17
 - Classes 7-14
 - Dataset business objects 10-6
 - Form business objects 7-4
 - Groups 15-30
 - Item business objects 6-5, 6-7
 - Persons 15-32
 - Projects 3-2
 - Property constants 6-33
 - Queries 17-6
 - Roles 15-31
 - Users 15-34
 - Volumes 15-25
- Creating ACL 20-30
- Custom reports

- Concepts 18-6
- D**
- Data
 - In-process 20-10
 - Released 20-10
 - Working 20-10
- data model
 - files 14-2
- Data model 5-12–5-13
 - Deploying 4-10–4-12
 - Example 6-16
 - Packaging 14-5
 - Schema 5-19
- Data sharing utilities
 - import_file 16-2
 - plmxml_export 16-10
 - plmxml_import 16-6
- Dataset
 - Tool 10-3
- Dataset business objects
 - Creating 10-6
 - Reference 10-2
- Date entry (revision rules) 19-22
- Date/unit/end item, set 19-34
- Deactivating group members 15-56
- Deep copy rules 13-26
 - Adding 13-28
 - hierarchy 13-27
- Default local volume, definition .. 15-35
- Defining search criteria 17-14
- Definition pane 21-5
- Definition tab 21-5
- Delete privilege 20-19
- Demote privilege 20-19, 24-14
- Deploying
 - Data model changes 4-11
 - Operational data 4-12
 - Overview 4-8, 4-10, 4-12
- Deploying data model changes 4-9
- Displaying inactive group
 - members 15-57
- Dry run 19-68
- E**
- Eclipse
 - Downloading from IBM 3-7
 - Version 3-25
- ECN, *see* Enterprise change notices
- ECO reports
 - ECO Details Report 18-8
 - ECO Signoff Details 18-8
- ECR, *see* Enterprise change requests
- Edit Revision Effectivity dialog
 - box 19-41
- Editors
 - UML 5-19
- Effective ACL example 20-17
- Effectivity 23-31
 - Nested 19-43
 - Revision, display data 19-39
 - Revision, edit data 19-40
- Effectivity mapping 19-49
- End item, purpose 19-3
- Enterprise change notices 23-31, 23-35–23-36
- Enterprise change requests 23-31, 23-35–23-36
- Environment 1-19
- Environment variables 1-20, 1-28
- EPM-set-rule-based-protection
 - handler 22-12
- Exclude licenses 15-17
- Export As String column 5-37
- Export privilege 20-19
- Exporting
 - Preferences 2-32
- Extension files
 - Setting 6-13
- Extensions
 - View 3-14
 - Working with 13-32
- External groups 20-44
- F**
- Failure path 24-27
- Fast track 23-31, 23-35
- Filter LOVs 9-16

Find 15-3
 Group 15-52
 Inactive group members 15-52
 Reload 15-52
 Role 15-52
 User 15-52
 Folder, snapshot 19-67
 Follow on Export column 5-37
 Form business objects 7-3
 Creating 7-4
 Hide properties 7-8
 Properties 7-7
 Storage class form 7-6

G

Generic process templates
 Create 24-3
 Generic Relationship Manager 5-42
 getpropertyconstantvalue utility .. 6-35
 Getting started 15-64, 22-2, 23-34
 Getting Started button
 Rich client 1-13
 Getting started with Report
 Builder 18-2
 GRM rules
 Adding 13-7
 Group
 Administrator 20-22
 Group entries by item type (revision rules) 19-27
 Group-level security
 Configuring for
 suppliers 20-43–20-44
 External groups 20-44
 Internal groups 20-43
 Grouped entries (revision rules) .. 19-25
 Grouping entries, example (revision rules) 19-26
 Groups 15-13
 Activating members 15-55
 Creating 15-30
 Deactivating members 15-56
 Defining roles 15-12
 Displaying inactive members .. 15-57
 Group member 15-13
 Managing 15-53

Removing members 15-54
 Subgroups 15-14
 Suppressing display of inactive members 15-57

H

Handlers 24-13
 Action 24-15
 Rule 24-15
 Handlers arguments 24-15
 Has item type entry 19-28
 Has item type rule entry 19-27
 Hide properties 7-8
 Hierarchical LOVs 9-18–9-19
 Hierarchy
 Business object 3-12
 Class 3-13

I

IBM Eclipse 3-7
 ID context rules 13-23
 IDC
 Capturing structure lines 19-70
 Creating 19-69
 Import files 10-9
 Import privilege 20-19
 import_export_reports utility
 Using 18-14
 import_file utility 16-2
 Import_file utility 16-17
 Importing
 Preferences 2-31
 Projects 4-15
 Template files 4-17
 Importing workflow templates 16-6
 Imprecise assembly
 Purpose 19-3
 In Job condition 22-12
 In-process data 22-12
 Inactivating user accounts 15-49
 infodba 15-20
 Inherited column 5-36
 Initial installation 4-5
 Initial value
 Column 5-36

- Installing
 - Additional templates 4-6
 - Business Modeler IDE 3-26–3-27
 - First time 4-5
 - Templates 4-2, 14-8
 - Templates to a production
 - server 14-4
- Interdependent LOVs 9-20
- Interface 23-7
 - Business Modeler IDE 1-10
 - Rich client 1-10
- Interfaces
 - Rich client 1-12–1-13
- Intermediate data capture, *see* IDC
- Internal groups 20-43
- Introduction to the IDE 3-7
- IP Admin privilege 20-20
- IP licenses 15-17
- ITAR Admin privilege 20-20
- ITAR licenses 15-17
- Item business objects
 - Creating 6-5, 6-7
 - Data model 6-15
 - Extending 6-4
 - Reference 5-17
- Item Ownership report 18-8
- Item reports
 - Concepts 18-6
- Item revision configuration 19-1
- Items by Status report 18-8
- J**
- Java
 - Runtime Environment (JRE) 3-25
- Job 23-3
- L**
- Latest entry (revision rules) 19-20
- Latest entry by creation date, example (revision rules) 19-21
- Launching
 - Business Modeler IDE 3-28
- Licenses
 - Exclude 15-17
 - IP 15-17
 - ITAR 15-17
 - Licensing 15-11, 22-6
 - Licensing level
 - Author 15-36
 - Consumer 15-36
 - Links
 - Fail 24-27–24-28
 - List of values (LOV)
 - Add, remove or clear 9-12
 - list_users utility 1-23
 - Lists of values (LOVs)
 - Adding 9-5
 - Attaching 9-10
 - Cascading 9-13, 9-15, 9-18–9-19
 - Filter 9-13–9-14, 9-16
 - Hierarchical 9-18–9-19
 - In general 1-8
 - Interdependent 9-13, 9-20
 - interface 9-3
 - Working with 9-2
 - Literal variables in naming
 - rules 13-21
 - Lower Bound column 5-36
- M**
- make_user 1-27, 15-60
 - Examples 1-29, 15-61
- make_user utility 1-26
- Managing groups 15-53
- Menu bar 1-12–1-13
 - Rich client 1-12–1-13
- Modifiable property constant 6-22
- Modifying
 - Volume properties 15-28
- Modifying an existing project 21-13
- N**
- Name collisions 6-42
- Named references 10-9–10-10
- Naming objects 6-42
- Naming rules
 - Adding 13-16
 - Attaching 13-19
 - Counters 13-22
 - In general 13-14

Literal variables 13-21
 Navigation pane 1-13
 Rich client 1-13
 Navigator view 3-17
 Nested effectivity 19-43
 Configuring 19-44
 Effectivity mapping 19-49
 Revision rules 19-46
 No Backpointer column 5-37
 Note types
 Adding 12-4
 Reference 12-3
 Nulls Allowed column 5-36

O

Object model hierarchy 22-5
 Object-based protection 20-8, 22-9
 Occurrence
 Managing part relationships ... 19-54
 Show superseded item
 revision 19-55
 Update precise 19-56
 Open in UML Editor menu ... 5-23–5-24
 Operational data 4-12
 Options 2-4
 In general 1-8
 Setting 2-6
 Status 12-6
 Tool 10-3
 Viewing 2-6
 Working with 12-2
 Organization
 Account generation 1-27, 15-60
 Export 15-69–15-70
 Find 15-3, 15-52
 Group hierarchy 15-15
 Groups 15-13
 Import 15-69, 15-71
 Passwords 15-10
 Persons 15-8
 Roles 15-12
 Users 15-9
 Volumes 15-16
 Organization List tree 15-3
 Organization tree 15-3
 Outline view 5-21

Override folder 19-35
 Override list, purpose 19-3
 Overview
 Deployment 4-8
 Initial template installation 4-5
 Subsequent template
 installation 4-6
 Template installation 4-2
 Template installation to a production
 server 14-4
 Owner (owning user) 20-22
 Owning group 20-22

P

Packaging extensions for
 installation 14-5
 Palette, UML 5-25, 5-27
 Panes
 Attribute Selection 17-3
 Saved Queries tree 17-3
 Saved query properties 17-3
 Search criteria 17-12
 Search Criteria 17-3
 Parts
 Managing occurrence
 relationships 19-54
 Passwords 15-10
 Persistent object model (POM) 5-15
 Persistent properties,
 adding 6-18–6-19
 Persons 15-8
 Creating 15-32
 Perspectives
 Business Modeler IDE 3-10
 Platform of the IDE 3-7
 PLM XML Export Import
 Administration 5-32, 18-9
 PLM XML report data 18-9
 plmxml_export utility 16-10
 plmxml_import utility 16-6
 POM schema
 In rich client applications 5-32
 PR, *see* Problem reports
 Precise assembly
 Purpose 19-3
 Preference

- Import and export 2-30
- Preference precedence 2-13, 2-23
- Preference scope 2-11, 2-23
- Preferences 2-2, 2-4, 2-15, 2-20
 - Business Modeler IDE 3-18
 - categories 2-10
 - Exporting 2-32
 - Importing 2-31
 - Index 2-7
 - Options 2-5
 - Organization 2-19
 - Search Options 2-18, 2-22
 - Setting 2-8, 2-21
 - subcategories 2-10
 - Viewing 2-8, 2-21
- Preferences options 2-2
- Prerequisites 3-25
- Prerequisites for Change Manager 23-30
- Prerequisites for Report Builder 18-3
- Prerequisites for Workflow Designer 23-6
- Primary application buttons 1-13
 - Rich client 1-13
- Privileged team member 21-9
- Privileged use commands 19-5
- Privileges
 - Assign to project 20-20
 - Change 20-19
 - Change ownership 20-19
 - CICO 20-21
 - Delete 20-19
 - Demote 20-19, 24-14
 - Export 20-19
 - Import 20-19
 - IP Admin 20-20
 - ITAR Admin 20-20
 - Promote 20-19, 24-14
 - Publish 20-19
 - Read 20-19
 - Remote checkout 20-20
 - Remove from project 20-20
 - Subscribe 20-19
 - Transfer in 20-19
 - Transfer out 20-19
 - Unmanage 20-20
 - Write 20-19
 - Write Classification ICO 20-20
- Problem reports 23-31, 23-35–23-36
- Process template 23-4
- Process templates
 - Create 24-2
 - Export 23-12
 - Import 23-12
- Process Templates 23-3
- Product generations 19-51
- Product structure
 - Apply revision rule 19-32
- Product Structure reports 18-8
- Project
 - Assign objects 21-17–21-18
 - Modifying 21-13
 - Privileged team member 21-9
 - Project administrator 21-8
 - Project team administrator 21-8
 - Project team member 21-9
 - Security rule tree 21-12
- Project administrator 21-8
- Project team administrator 21-8
- Project team member 21-9
- Project-level security
 - Access rules 21-11
- Project-level security tasks
 - Applying Access Manager rules 21-11
 - Configuring automatic assignment to projects 21-15
- Projects
 - Automatically assigning data to 21-15
 - Business Modeler IDE template 3-18
 - Create 21-10
 - Creating 3-2
 - Files 3-4
 - Importing 4-15
- Promote privilege 20-19, 24-14
- Properties
 - Business objects 5-7
 - Constants 5-49
 - Data Model 18-10
 - Form business objects 7-7

- In general 1-8
- Make visible or required 3-38
- Persistent 6-18–6-19
- Property constants
 - Creating 6-33
 - Modifiable 6-22
 - Reference 5-47, 6-20
 - Update 6-32
- Property sets 18-13
- Protecting Teamcenter data 20-6
- Public Read column 5-36
- Public Write column 5-36
- Publish privilege 20-19
- Purpose of the IDE 3-7
- Q**
- Queries
 - Based on existing definitions . . . 17-15
 - Creating 17-3, 17-6
 - Definition window 17-3
 - Referenced-By 17-23
- Query
 - Custom item type 17-21
- Query Builder 5-32
- Quick search
 - Rich client 1-12
- Quick-release process 23-19
- R**
- Read privilege 20-19
- Reference
 - Attributes 5-31
 - Business object display rules . . . 13-4
 - Change 12-12
 - Dataset business objects 10-2
 - Note types 12-3
 - Unit of measure 12-9
- Reference Class column 5-36
- Reference object 23-3
- Referenced-by queries 17-23
- Relationships
 - Business objects 13-7
- Release Data
 - overview A-7
- Release status
 - Purpose 19-3
- Remote checkout privilege 20-20
- Remove from project privilege . . . 20-20
- Removing
 - Members from groups 15-54
- Report Builder 5-32
 - Concepts 18-5
 - Getting started 18-2
 - User interface 18-4
- Reports
 - Admin-Employee Information . . . 18-8
 - Admin-Group/Role
 - Membership 18-8
 - Admin-Item Ownership 18-8
 - Admin-Items by Status 18-8
 - Admin-Object Ownership 18-8
 - Admin-Objects by Status 18-8
 - ECO Details Report 18-8
 - ECO Signoff Details 18-8
 - PS - BOM Structure 18-8
- Required
 - Business object properties 3-38
- Resource pool subscription 24-11
- Responsible Party accessor 24-14
- Revision configuration
 - Elements 19-3
- Revision effectivity
 - Display data 19-39
 - Edit data 19-40
- Revision rules
 - Apply 19-32
 - Create rule entry 19-8
 - Creating 19-7
 - Date entry 19-22
 - Defining entries 19-13
 - Delete rule entry 19-10
 - Edit rule entry 19-11
 - Group entries by item type 19-27
 - Grouped entries 19-25
 - Grouping entries, example 19-26
 - Latest by creation date entry,
 - example 19-21
 - Latest entry 19-20
 - Modify 19-12
 - Modify current rule 19-12
 - Modify rule entry 19-9

- Nested effectivity 19-46
 - Order of precedence 19-2
 - Purpose 19-3
 - Set 19-33
 - Set date/unit/end item 19-34
 - Status entry 19-17
 - Status entry with effective date, example 19-19
 - Status hierarchy 19-18
 - Understanding 19-2
 - Unit number entry 19-23
 - Using from My Teamcenter 19-53
 - Viewing information 19-36
 - Working entry 19-14
 - Working entry with current user/group, example 19-16
 - Working entry, example 19-15
 - Rich client
 - Application banner 1-12
 - Application pane 1-13
 - Back and forward buttons 1-12
 - Clipboard 1-13
 - Getting Started button 1-13
 - Interface 1-10
 - Menu bar 1-12–1-13
 - Navigation pane 1-13
 - Primary application buttons 1-13
 - Search field 1-12
 - Secondary application buttons .. 1-13
 - Toolbar 1-12–1-13
 - Rich client interface 1-12–1-13
 - Roles 15-12
 - Adding new 15-41
 - Creating 15-31
 - Rule entry 19-3
 - Create 19-8
 - Delete 19-10
 - Edit 19-11
 - Modify 19-9
 - Rule handlers 24-15, 24-18
 - Rule tree 20-11
 - Rule tree precedence 20-11
 - Rules 13-2
 - Adding 20-29
 - Business object display 13-4–13-5
 - Compound property .. 5-38, 6-25–6-26
 - Date entry 19-22
 - Deep copy 13-26–13-28
 - Definition 20-7, 20-12, 22-8
 - Example 20-23
 - Extension 13-32
 - GRM 13-7
 - Group entries by item type 19-27
 - Grouped entries 19-25
 - Grouping entries, example 19-26
 - Has item type 19-28
 - Has item type, defining entry .. 19-27
 - ID context 13-23
 - In general 1-8
 - Latest entry 19-20
 - Latest entry by creation date, example 19-21
 - Modifying 20-31
 - Naming 13-14
 - Status entry 19-17
 - Status entry with effective date, example 19-19
 - Status hierarchy 19-18
 - Subbranch precedence 20-11
 - Syntax 20-13
 - Tree 20-11
 - Unit number entry 19-23
 - Working entry 19-14
 - Working entry with current user/group, example 19-16
 - Working entry, example 19-15
 - Rules-based protection 20-7, 20-12, 22-8
- ## S
- Saved Queries tree pane 17-3
 - Saved query properties pane 17-3
 - Scripts 1-21
 - Search Class button 17-3
 - Search criteria
 - Class attributes 17-8, 17-11
 - Defining 17-14
 - Search Criteria pane 17-3
 - Search field 1-12
 - Secondary application buttons 1-13
 - Security Rules pane 21-5

Security Services 15-11
 Server
 Connection profiles 3-18, 3-22
 Deploying changes to 4-10–4-12
 Server profile 3-21
 Two-tier 3-23, 6-23
 Set active extension files 6-13
 Snapshots
 Create 19-59
 Open 19-60
 Purpose 19-3
 View 19-61
 Source Class column 5-36
 Source control management (SCM)
 systems
 Introduction 14-3
 Standard track 23-31, 23-35
 Start the Business Modeler IDE ... 3-29
 Start the IMR 3-29
 Starting
 Business Modeler IDE 3-28
 Status 12-6
 Adding 12-7
 Status entry (revision rules) 19-17
 Status entry with effective date (revision
 rules) 19-19
 Status hierarchy (revision rules) .. 19-18
 Storage Type column 5-35
 Structure lines, capturing in
 IDC 19-70
 Subgroups 15-14
 Subscribe privilege 20-19
 Subscriptions Index-1
 Subsequent template installation ... 4-6
 Summary reports
 Concepts 18-6
 Suppressing display of inactive members
 in groups 15-57
 System administrator 20-22
 System maintenance utilities
 clearlocks A-2
 list_users 1-23
 make_user 1-26
 System-level rules 15-67

T

Tabs
 CreateDescriptor 3-38
 Target object 23-3
 Task handlers
 Creating 24-15
 Task Owner accessor 24-14
 Task Owning Group accessor 24-14
 Task template 23-4
 Task templates
 Handlers 24-15
 Tasks 23-3, 23-5
 Attributes 24-13
 Rich Client Administration 1-14
 Team members 21-9
 Privileged team member 21-9
 Project administrator 21-8
 Project team administrator 21-8
 Team member 21-9
 Teamcenter
 administrative applications 1-5
 Teamcenter Environment Manager
 (TEM) 3-26–3-27
 Teamcenter interfaces
 Rich client 1-12–1-13
 Teamcenter preferences 3-18
 Teamcenter security applications .. 22-3
 Teamcenter's reporting and
 analytics 5-32
 templates
 manage 14-2
 Templates 4-4
 Deploying 4-10–4-12
 In general 4-3
 Installing 14-8
 Overview of installation 4-2
 Packaging 14-5
 Workflow process 23-4
 Workflow task 23-4
 Tool 10-3
 Toolbar 1-12–1-13
 Tools
 Adding 10-4
 Topic

Subtopic 1-15,
 1-18, 5-18, 5-33, 5-41, 5-43, 6-3, 6-11,
 6-31, 6-39–6-41, 7-2, 7-11
 Transfer in privilege 20-19
 Transfer mode objects 18-11
 Transfer out privilege 20-19
 Transient column 5-37
 Tree
 Business object 3-12
 Class 3-13
 Tutorials 3-28

U

UML editor
 Add business object 5-27
 Add class 5-27
 Data model 5-22
 UML diagram 5-20
 Using 5-19
 Unique column 5-36
 Unit number entry (revision
 rules) 19-23
 Units of measure
 Adding 12-10
 Reference 12-9
 Unmanage privilege 20-20
 Unprivileged user commands 19-5
 Upper Bound column 5-36
 User interface
 Business Modeler IDE 1-11, 3-9
 User interface, Report Builder 18-4
 Users 15-9
 Activating account 15-51
 Adding existing 15-46
 Adding new 15-43
 Changing status 15-48
 Creating 15-34
 Inactivating account 15-49
 Users, configuring 19-5
 Utilities
 bmid_postupgradetotc 4-17
 getpropertyconstantvalue 6-35

V

Views

Business Modeler IDE 3-11
 Business Objects 3-12
 Classes 3-13
 Console 3-16
 Extensions 3-14
 Navigator 3-17
 Outline 5-21
 Visible properties 3-38
 Volumes 15-16
 Controlling access 15-29
 Creating 15-25
 Modifying properties 15-28

W

Welcome window 3-28
 workflow
 actions 24-38
 cancel 24-39–24-40
 resume 24-40
 troubleshooting 24-37
 Workflow
 Resume tasks 24-40
 Workflow accessors
 Approver 24-14
 Approver (Group) 24-14
 Approver (RIG) 24-14
 Approver (Role) 24-14
 Responsible Party 24-14
 Task Owner 24-14
 Task Owning Group 24-14
 Workflow ACLs 24-13
 Workflow Designer
 Definition 23-2
 Interface 23-7
 Tasks 23-5
 Workflow Designer interface 23-6
 Workflow privileges
 Demote 24-14
 Promote 24-14
 Workflow process template 23-4
 Workflow task template 23-4
 Workflow tasks 24-4
 Workflow templates 23-11
 Create 24-2
 Create baseline process
 templates 23-27

Create generic process		
templates	24-3	
Enterprise process modeling		
(EPM)	23-9	
Exporting	23-13	
Importing	23-14	
Model a process	23-11	
Process terms	23-3	
Tasks	24-4	
Workflow Designer interface	23-6	
Workflow templates, importing	16-6	
Working ACL	20-24	
Working data	22-11	
Working entry (revision rules)	19-14	
Working entry (revision rules),		
example	19-15	
Working entry with current user group,		
example (revision rules)	19-16	
Working revision, definition	19-4	
World	20-22	
Write Classification ICO		
privilege	20-20	
Write privilege	20-19	

Reference tear-out pages

These reference tear-out pages are provided for your convenience.

Course agenda

Day 1	Morning	
	Introduction	
	Course overview	
	Lesson 1	Introduction to administration
	Lesson 2	Introduction to organization
	Afternoon	
	Lesson 3	Introduction to the Business Modeler IDE
	Lesson 4	Data model
	Lesson 5	UML editor
Day 2	Morning	
	Lesson 6	Lists of values
	Lesson 7	Datasets
	Lesson 8	Options
	Afternoon	
	Lesson 9	Constants
	Lesson 10	Rules
	Lesson 11	Data model files
Day 3	Morning	
	Lesson 12	Organization hierarchy
	Afternoon	
	Lesson 13	Query Builder definitions
	Lesson 14	Preference management
Day 4	Morning	
	Lesson 15	Report Builder definitions
	Afternoon	
	Lesson 16	Access Manager
	Lesson 17	Projects to control access
Day 5	Morning	
	Lesson 18	Workflow process modeling
	Course summary	

Classroom data sheet

This table is provided so students can record their classroom setup, as described by the instructor. Optionally, instructors may hand out a preprinted data sheet.

Data item	Data value	Domain
OS user ID OS password		Local computer
OS user ID OS password		Virtual image
Teamcenter user ID Teamcenter password		Virtual image
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

Activities

Perform the activities to use the virtual image on to your computer.

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

Activity: Start the virtual image

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.
 - Double-click the **VMware Player** shortcut on the desktop.
 - Select **Teamcenter 8** under **Recent Virtual Machines**.
 - 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: Stop and restart the virtual image

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.

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.

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.

Instructor Note:

This topic is not covered in detail later in the course.

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.

Instructor Note:

This topic is not covered in detail later in the course.

Student profile



STUDENT PROFILE

To stay in tune with our customers, we ask for some background information. This information will be kept confidential and will not be shared with anyone outside of Education Services.

Please print:

Your name _____ **U.S. citizen** ☐ Yes ☐ No

Course title/Dates _____ / _____ **through** _____

Hotel/motel(s) while training _____ **Planned departure time after class** _____

Employer _____ **Location** _____

Supervisor/manager _____ **(Emergency) Phone** _____

Your job title/responsibilities _____ / _____

Industry: ☐ Auto ☐ Aero ☐ Consumer products ☐ Machining ☐ Tooling ☐ Medical ☐ Other

Types of products/parts/data that you work with _____

Platform (operating system) _____

Reason for training _____

Please verify/add to this list of training for NX, I-deas, Imageware, Teamcenter, Tecnomatix or Dimensional Mgmt./Visualization.
Medium means Instructor-lead (IL), Online (OL), or Self-paced (SP)

Software	From whom	When	Course name	Medium

Other CAD/CAM/CAE /PDM software you have used _____

Please check (✓) your ability/knowledge level in the following areas:

<u>Subject</u>	<u>None</u>	<u>Novice</u>	<u>Intermediate</u>	<u>Advanced</u>
CAD modeling	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CAD assemblies	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CAD drafting	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CAM	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
CAE	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PDM – usage	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PDM – system management	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
PDM – customization	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Thank you for your participation. We hope your training experience will be an outstanding one.

Course evaluation



PLM Software
Evaluation – Delivery

Course name: _____ Course #: _____

Course dates: _____ through _____

Please share your opinion in all of the following sections with a check in the appropriate box:

Instructor: ☒

If there were two 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"/>

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) _____

SEE BACK

PLM Software
Evaluation - Courseware



Course name: _____ **Course #:** _____

Course dates: _____ **through** _____

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

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"/>

Name (optional): _____ Location/room _____

☐ Please check this box if you would like your comments featured in our training publications.
(Your name is required at the bottom of this form)

☐ Please check this box if you would like to receive more information on our other courses and services.
(Your name is required at the bottom of this form)

*Thank you for your business. We hope to continue to provide
your training and personal development for the future.*