

Teamcenter Application and Data Model Administration

**Student Guide
August 2013
MT25460 – Teamcenter 10.1**

**Publication Number
MT25460_S_101**

Proprietary and restricted rights notice; Trademarks

Proprietary and restricted rights notice

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

© 2014 Siemens Product Lifecycle Management Software Inc.

Trademarks

Siemens and the Siemens logo are registered trademarks of Siemens AG. Teamcenter is a trademark or registered trademark of Siemens Product Lifecycle Management Software Inc. or its subsidiaries in the United States and in other countries. All other trademarks, registered trademarks, or service marks belong to their respective holders.

Contents

| | |
|---|------------|
| Proprietary and restricted rights notice; Trademarks | 2 |
| Course overview | 17 |
| Course objectives | 17 |
| Key benefits | 18 |
| Prerequisites | 18 |
| Audience | 18 |
| Learning tracks | 18 |
| Accessing Teamcenter online help | 19 |
| Training scenario | 19 |
| Business Modeler IDE fundamentals | 1-1 |
| Introduction to the Business Modeler IDE | 1-2 |
| Business Modeler IDE interface | 1-3 |
| Standard perspective | 1-4 |
| Advanced perspective | 1-5 |
| Interface favorites and filtering | 1-6 |
| Basic concepts for using the Business Modeler IDE | 1-7 |
| Business Modeler IDE administration tasks | 1-8 |
| Objects to be created by the Business Modeler IDE | 1-9 |
| Overview to administering Teamcenter | 1-10 |
| Rich client interface | 1-11 |
| Teamcenter command prompt interface | 1-13 |
| Teamcenter Environment Manager (TEM) interface | 1-14 |
| Organization for this course | 1-15 |
| Business Modeler IDE process overview | 1-16 |
| Developing extensions and testing | 1-17 |
| Deploying a template to a production site | 1-18 |
| Editing extensions in a live production site | 1-19 |
| Edit live data | 1-20 |
| Incorporate live data updates from the production site | 1-21 |
| Development environments | 1-22 |
| A Business Modeler IDE template project | 1-24 |
| Create a Business Modeler IDE template project | 1-25 |
| Naming objects | 1-27 |
| Files in a template project | 1-28 |
| Adding extensions to the template | 1-30 |
| Import a template project | 1-31 |

| | |
|--|------------|
| Understanding custom versus COTS templates | 1-32 |
| Introduction to templates | 1-33 |
| Using an SCM system to manage files | 1-34 |
| Configure the Business Modeler IDE | 1-35 |
| Add a server connection profile | 1-36 |
| Introduction to extension files | 1-38 |
| Add a new extension file | 1-39 |
| Set an active extension file | 1-40 |
| Extension file example | 1-41 |
| Extension file example | 1-42 |
| Project backup | 1-43 |
| Back up project data | 1-44 |
| Restore project data | 1-45 |
| Restore alternatives | 1-46 |
| Import a backed-up project | 1-48 |
| Backup and recovery of Business Modeler IDE data | 1-49 |
| Set your own user variables for the classroom activities | 1-50 |
| Activities | 1-51 |
| Summary | 1-52 |
| BMIDE process and data model | 2-1 |
| Introduction to the data model concepts | 2-2 |
| Teamcenter data model primer | 2-3 |
| What are business objects | 2-4 |
| Why create business objects? | 2-5 |
| Find business objects | 2-6 |
| POM schema importance | 2-7 |
| Viewing POM schema | 2-8 |
| Tasks for using the data model | 2-9 |
| Introduction to the UML editor | 2-10 |
| Inheritance to POM_object | 2-11 |
| Data model inheritance | 2-12 |
| Outline view in the Business Modeler IDE | 2-13 |
| Basic process for extending the data model | 2-14 |
| Add a new model element | 2-15 |
| Basic item data model | 2-16 |
| Data model extension example | 2-17 |
| Create an Item business object | 2-18 |
| Set display names | 2-20 |
| Introduction to properties | 2-21 |
| Working with properties | 2-22 |
| Create a persistent property | 2-23 |
| Attribute types | 2-24 |
| Attribute type settings | 2-25 |
| Using the Operation Descriptor tab | 2-26 |
| Operation Descriptor tab for CreateInput | 2-27 |

| | |
|--|------------|
| Operation Descriptor tab for SaveAsInput | 2-29 |
| Lists of values (LOVs) | 2-31 |
| Attach an LOV to a property | 2-32 |
| Deploying templates | 2-33 |
| Basic process for deploying a template | 2-34 |
| Install a template to a production server | 2-35 |
| Package extensions into a template | 2-36 |
| Install a template using TEM | 2-38 |
| Activities | 2-39 |
| Summary | 2-40 |
| Item business object configuration | 3-1 |
| Extending item business objects | 3-2 |
| Extending item business objects | 3-3 |
| Basic item structure | 3-4 |
| Effective use of inheritance | 3-5 |
| Using a CommonItem business object | 3-6 |
| Defining business objects | 3-7 |
| Business object properties | 3-8 |
| Add a persistent property | 3-9 |
| Property constants | 3-10 |
| Property constant behavior | 3-11 |
| Property constants | 3-12 |
| Working with property constants | 3-13 |
| Change a property constant value | 3-14 |
| Introduction to multifield keys | 3-15 |
| Multifield keys in Teamcenter applications | 3-16 |
| Configure the Object name | 3-17 |
| Import a template file | 3-20 |
| Import localization files | 3-21 |
| Add the Localization button to properties | 3-22 |
| Set display names for properties | 3-23 |
| Add or change a business object icon | 3-24 |
| Set up a business object icon overlay | 3-26 |
| View the icons in the rich client | 3-27 |
| Activities | 3-28 |
| Summary | 3-29 |
| Form business object configuration | 4-1 |
| Teamcenter forms | 4-2 |
| End user form creation | 4-3 |
| Defining forms | 4-4 |
| Forms data model | 4-5 |
| Create a form business object | 4-6 |
| Create the form business object properties | 4-7 |

| | |
|---|------------|
| Create a persistent property | 4-8 |
| Property constants | 4-9 |
| Hide properties on a form business object | 4-10 |
| Activities | 4-11 |
| Summary | 4-12 |
| LOV (list of value) extensions | 5-1 |
| Introduction to lists of values (LOVs) | 5-2 |
| List of values (LOV) interface | 5-3 |
| LOV types | 5-4 |
| LOV value types | 5-5 |
| LOV usage types | 5-6 |
| List of values (LOV) types | 5-7 |
| Add a list of values (LOV) | 5-9 |
| Attach an LOV to a property | 5-10 |
| Add, remove, or clear a value to an LOV | 5-11 |
| Set display names for lists of values (LOVs) | 5-12 |
| Classic LOVs | 5-13 |
| Filter LOV | 5-14 |
| Cascading LOV | 5-15 |
| Interdependent LOV | 5-16 |
| LOV loaded from the database | 5-17 |
| Create a filter LOV | 5-18 |
| Create a cascading LOV | 5-19 |
| Dynamic LOVs | 5-20 |
| Create dynamic lists of values | 5-21 |
| Create dynamic LOV query | 5-22 |
| Considerations for creating dynamic LOVs | 5-25 |
| Activities | 5-26 |
| Summary | 5-27 |
| Relation business object configuration | 6-1 |
| Introduction to relation business objects | 6-2 |
| Defining a new relation | 6-3 |
| New relation properties | 6-5 |
| New relation property constants | 6-6 |
| Business object property of new relation | 6-7 |
| Create a new relation business object | 6-8 |
| Add a relation property | 6-9 |
| Activities | 6-10 |
| Summary | 6-11 |
| Dataset business object configuration | 7-1 |
| Introduction to dataset business objects | 7-2 |
| Named references | 7-3 |

| | |
|--|------------|
| Dataset and named references | 7-4 |
| Create a dataset business object | 7-5 |
| Set up the dataset named references | 7-7 |
| Configure dataset view and open | 7-8 |
| Introduction to tools | 7-9 |
| Add a tool | 7-10 |
| Activities | 7-12 |
| Summary | 7-13 |
| Option extensions and BMIDE reports | 8-1 |
| Introduction to options | 8-2 |
| Introduction to notes | 8-3 |
| Add a note type | 8-4 |
| Introduction to status | 8-5 |
| Add a status | 8-6 |
| Introduction to units of measure | 8-7 |
| Add a unit of measure | 8-8 |
| Data model reports | 8-9 |
| Compare Two Data Models report | 8-10 |
| Data Model report | 8-11 |
| Other Business Modeler IDE reports | 8-12 |
| Run the Compare Two Data Models report | 8-13 |
| Run other Business Modeler IDE reports | 8-14 |
| Activities | 8-15 |
| Summary | 8-16 |
| Rule extensions | 9-1 |
| Introduction to rules | 9-2 |
| Naming rules introduction | 9-3 |
| Creating naming rules | 9-4 |
| Create and attach a naming rule | 9-5 |
| Add a naming rule | 9-6 |
| Attach a naming rule to a property | 9-7 |
| Using system variables in patterns | 9-8 |
| Other naming rules pattern control | 9-9 |
| Revision naming rule | 9-10 |
| Supplemental revision types | 9-11 |
| Add a revision naming rule | 9-12 |
| Working with business object display rules | 9-13 |
| Add a business object display rule | 9-14 |
| Deep copy rules | 9-15 |
| Deep copy rule definitions | 9-16 |
| Deep copy examples | 9-17 |
| Add a deep copy rule | 9-18 |
| Add a deep copy rule (continued) | 9-19 |

| | |
|---|-------------|
| Conditions introduction | 9-21 |
| Condition example | 9-22 |
| Conditions overview | 9-23 |
| Add a condition | 9-24 |
| Condition input parameters and signature | 9-25 |
| Condition operators | 9-26 |
| Condition examples | 9-27 |
| Attach LOVs with conditions | 9-30 |
| Project LOV use case | 9-31 |
| Activities | 9-33 |
| Summary | 9-34 |
| Data model live updates | 10-1 |
| Introduction to live updates | 10-2 |
| Data elements that can be updated live | 10-4 |
| Working with live updates | 10-7 |
| Live updates: single versus multiple administrators | 10-8 |
| Single Business Modeler IDE administrator of live updates | 10-9 |
| Multiple Business Modeler IDE administrators of live updates | 10-10 |
| Business Modeler IDE process review | 10-11 |
| Incorporate live data updates from the production site | 10-12 |
| Live update process: single administrator | 10-14 |
| Enable a template for live updates and deploy it | 10-15 |
| Configure the Live Update preference | 10-16 |
| Create a live update project | 10-17 |
| Perform live updates | 10-18 |
| Package a live update project for installation to other sites | 10-20 |
| Incorporate latest live updates | 10-22 |
| Live updates reference tasks | 10-25 |
| Synchronize data model | 10-26 |
| Data model compare preferences | 10-27 |
| Make live updates visible to end users | 10-28 |
| Package live updates in the database | 10-29 |
| Study the merge samples | 10-30 |
| Activities | 10-32 |
| Summary | 10-33 |
| Organization | 11-1 |
| Introduction to Organization | 11-2 |
| Organization interface | 11-3 |
| Organization hierarchy | 11-4 |
| Basic concepts for using Organization | 11-5 |
| Typical organization administration tasks | 11-7 |
| What is a person? | 11-8 |
| What is a user? | 11-9 |

| | |
|---|-------------|
| Specifying password restrictions | 11-10 |
| Teamcenter Security Services | 11-11 |
| What is a role? | 11-12 |
| What is a group? | 11-13 |
| What is a subgroup? | 11-14 |
| Group hierarchies | 11-15 |
| What is a volume? | 11-16 |
| Considerations for managing administration accounts | 11-17 |
| infodba account | 11-18 |
| System administration accounts | 11-19 |
| Creating your virtual organization | 11-20 |
| Creating the organization structure | 11-21 |
| Creating a volume | 11-22 |
| Modifying volume properties | 11-25 |
| Controlling volume access | 11-26 |
| Creating a group | 11-27 |
| Creating a role | 11-28 |
| Creating a person | 11-29 |
| Creating a user | 11-31 |
| Group member settings | 11-36 |
| Add a new role to a group using the Organization Role wizard | 11-37 |
| Add a new user to a group/role using the Organization User wizard | 11-39 |
| Add an existing user to a role/group using the Organization User wizard | 11-42 |
| Changing user status | 11-44 |
| Inactivate a user account | 11-45 |
| Activate a user account | 11-47 |
| Using Organization find | 11-48 |
| Managing group members | 11-49 |
| Remove a member from a group | 11-50 |
| Deactivate a group member | 11-52 |
| Activate a group member | 11-53 |
| Suppress the display of inactive group members in the Organization tree | 11-54 |
| Utility account generation | 11-55 |
| make_user utility examples | 11-56 |
| Set your own user variables for the classroom activities | 11-58 |
| Activities | 11-59 |
| Summary | 11-60 |
| Access Manager | 12-1 |
| Getting started with Access Manager | 12-2 |
| Access Manager interface | 12-3 |
| About using Access Manager | 12-4 |
| Basic tasks using Access Manager | 12-5 |

| | |
|--|-------|
| Protecting Teamcenter data | 12-6 |
| Rules-based protection | 12-7 |
| Object-based protection | 12-8 |
| Access control lists | 12-9 |
| Lifecycle of data | 12-10 |
| Access Manager rule tree | 12-11 |
| How rules are defined | 12-12 |
| Rule syntax | 12-13 |
| Evaluating the rule tree for the effective ACL | 12-14 |
| Example rule tree evaluation by order of precedence | 12-15 |
| Activity | 12-16 |
| Example of compiling an effective ACL | 12-17 |
| Access privileges | 12-19 |
| Categories of accessors | 12-22 |
| Rule tree conditions | 12-26 |
| Complex rule tree example | 12-36 |
| Understanding the rule creation process | 12-39 |
| Add an Access Manager rule | 12-40 |
| Create an access control list (ACL) | 12-41 |
| Modify an Access Manager rule | 12-42 |
| Import and export the Access Manager rule tree | 12-43 |
| Import and export guidelines | 12-44 |
| Good rule practices | 12-45 |
| Cautions for using rule trees | 12-47 |
| About verifying the effect of access rules | 12-48 |
| View the rules from which privileges are derived | 12-49 |
| About controlling access to working data | 12-50 |
| About configuring access to working data | 12-51 |
| Guidelines for applying the delete and change privileges | 12-52 |
| Controlling access to revision rules | 12-53 |
| About controlling access to in-process data | 12-54 |
| Workflow accessors and privileges | 12-55 |
| Workflow ACL example | 12-57 |
| Parallel task and parallel process ACL conflict resolution | 12-58 |
| Workflow access examples | 12-59 |
| About configuring group-level security | 12-61 |
| Example of configuring security to prevent suppliers from viewing internal data | 12-63 |
| Example of configuring security for data owned by a supplier (external data) | 12-64 |
| Example of configuring supplier security using hierarchical groups | 12-65 |
| Example of configuring security for special project data using hierarchical groups (fully restrictive external group security) | 12-66 |
| About configuring security for project and program data | 12-67 |
| Basic concepts about projects | 12-68 |

| | |
|--|-------------|
| What are groups? | 12-69 |
| Applying project security (Access Manager) rules | 12-70 |
| Access rules for projects and programs | 12-72 |
| Project-level security based on groups | 12-73 |
| Placement of rules in the Access Manager rule tree | 12-74 |
| Set security precedence | 12-75 |
| Implementation considerations for program-level security | 12-76 |
| Activities | 12-77 |
| Summary | 12-78 |
| Projects to control access | 13-1 |
| Introduction to Project | 13-2 |
| Project Administration window | 13-3 |
| Project administration buttons | 13-4 |
| Project administration tabs | 13-5 |
| Project quick links | 13-6 |
| Basic concepts about projects | 13-7 |
| Project administrators and team members | 13-8 |
| Using projects | 13-10 |
| Applying project security (Access Manager) rules | 13-11 |
| Project security rule tree | 13-12 |
| Activating and deactivating projects | 13-13 |
| Modifying existing projects | 13-14 |
| Assigning data to projects | 13-15 |
| Teamcenter utilities for projects | 13-16 |
| Activities | 13-28 |
| Summary | 13-29 |
| Managing preferences | 14-1 |
| Managing preferences | 14-2 |
| Preference definition elements | 14-4 |
| System and hierarchical preferences | 14-8 |
| Managing protection scope | 14-9 |
| Using the rich client Options dialog box | 14-10 |
| Set a preference option | 14-11 |
| Create a new preference definition | 14-12 |
| Create a preference instance for a group | 14-14 |
| Create a preference instance for a nonadministrator user | 14-15 |
| Creating and editing preferences from preference XML files | 14-16 |
| Edit a preference XML file | 14-17 |
| Specify dual OS values for a single preference | 14-20 |
| Generating preference reports | 14-21 |
| Create preference reports | 14-22 |
| Import and export preferences | 14-23 |
| Import preferences in to the database | 14-24 |

| | |
|---|-------------|
| Export preferences from the database to an XML file | 14-25 |
| Activities | 14-26 |
| Summary | 14-27 |
| Query Builder definitions | 15-1 |
| Introduction to Query Builder | 15-2 |
| Basic concepts for using Query Builder | 15-3 |
| Query Builder interface | 15-4 |
| Search Criteria pane | 15-5 |
| Properties for workspace objects | 15-8 |
| Using search criteria clauses | 15-9 |
| Using class attribute selections | 15-10 |
| Creating customized searches using Query Builder | 15-12 |
| Create a new query based on an existing definition | 15-13 |
| Create a query using the hints feature | 15-14 |
| Create a query using the IS_NULL operator | 15-15 |
| Create queries | 15-16 |
| Class/Attribute selection | 15-17 |
| Add class attributes to search criteria | 15-18 |
| Custom item type query definitions | 15-19 |
| Importing and exporting query definitions | 15-21 |
| Import query definitions | 15-22 |
| Export query definitions | 15-23 |
| Activities | 15-24 |
| Summary | 15-25 |
| Report Builder definitions | 16-1 |
| Introduction to Report Builder | 16-2 |
| Report Builder interface | 16-3 |
| Basic concepts for using Report Builder | 16-4 |
| Report Builder definition types | 16-5 |
| Report definition structure | 16-6 |
| Standard Teamcenter report definitions | 16-7 |
| Importing and exporting report definitions | 16-8 |
| Activities | 16-9 |
| Summary | 16-10 |
| PLM XML import and export | 17-1 |
| What is PLM XML/TC XML Export Import Administration? | 17-2 |
| PLM XML/TC XML Export Import Administration interface | 17-3 |
| Teamcenter data model quick review | 17-4 |
| Basic tasks using PLM XML/TC XML Export Import Administration | 17-5 |
| What are transfer modes? | 17-6 |
| Create closure rules | 17-7 |
| Defining property sets | 17-8 |

| | |
|--|-------------|
| Create transfer mode objects | 17-9 |
| Transfer mode best practices | 17-10 |
| Transfer mode limitations | 17-11 |
| Use specific type identifiers for better performance | 17-12 |
| Unload objects in transfer mode to speed processing | 17-13 |
| Improve export performance | 17-14 |
| Maintain COTS scope rules | 17-15 |
| Activities | 17-16 |
| Summary | 17-17 |
| Introduction to Workflow Designer | 18-1 |
| Introduction to Workflow Designer | 18-2 |
| Workflow process terminology | 18-3 |
| Workflow process approaches | 18-4 |
| Basic concepts for using Workflow Designer | 18-5 |
| Basic tasks using Workflow Designer | 18-6 |
| Working with Workflow Designer | 18-7 |
| Workflow Designer interface | 18-8 |
| Workflow process templates | 18-9 |
| Workflow task templates | 18-10 |
| Enterprise Process Modeling | 18-12 |
| Defining a process model | 18-14 |
| Creating Workflow process templates | 18-15 |
| Create a generic process template | 18-16 |
| Do task templates | 18-17 |
| Review task template | 18-18 |
| Resource pool | 18-19 |
| Acknowledge task template | 18-20 |
| Add Status task template | 18-21 |
| Activities | 18-22 |
| Summary | 18-23 |
| Building workflow process templates | 19-1 |
| Modifying task behavior | 19-2 |
| Workflow accessors and privileges | 19-3 |
| Adding task handlers | 19-5 |
| Examples of useful handlers | 19-7 |
| Adding secure tasks | 19-9 |
| Deploying process templates | 19-10 |
| Activities | 19-12 |
| Summary | 19-13 |
| Workflow sharing and additional paths | 20-1 |
| Sharing process templates | 20-2 |
| Export a process template | 20-3 |

| | |
|---|-------------|
| Import a process template | 20-4 |
| Linking tasks in a workflow process template | 20-5 |
| Link tasks manually | 20-6 |
| Failure path | 20-7 |
| Create failure paths | 20-8 |
| Adding Condition tasks | 20-9 |
| Configuring Condition tasks | 20-11 |
| Set Condition task flow paths | 20-12 |
| Using Validate tasks | 20-14 |
| Validate task example: Close gaps in your workflow | 20-15 |
| Validate task behavior | 20-16 |
| Activities | 20-17 |
| Summary | 20-18 |
| Course summary | 21-1 |
| Additional LOV extensions | A-1 |
| Create an interdependent LOV | A-2 |
| Test the interdependent LOV | A-3 |
| Batch lists of values | A-4 |
| Create batch LOVs | A-6 |
| Create the externally managed LOV values | A-7 |
| Convert an LOV managed in a template to an externally managed LOV | A-8 |
| Convert an externally managed LOV to an LOV managed in a template | A-10 |
| Considerations for externally managing LOVs | A-11 |
| Activities | A-12 |
| Summary | A-13 |
| Compound property configuration | B-1 |
| Compound properties | B-2 |
| Working with compound properties | B-3 |
| Add a compound property | B-4 |
| Example – one segment rule path | B-5 |
| Example – two segment rule path | B-6 |
| Example – Structure Manager rule path | B-7 |
| Add a compound property example steps | B-8 |
| Summary | B-11 |
| Additional rules and operations | C-1 |
| Additional rules and operations | C-2 |
| Using operation extensions | C-3 |
| Working with GRM rules | C-11 |
| Activities | C-17 |

| | |
|---|------------|
| Summary | C-18 |
| Rich client interface configuration | D-1 |
| Property display using XML style sheets | D-2 |
| Using predefined style sheets | D-3 |
| Display format | D-4 |
| Accessing the style sheet viewer | D-5 |
| Examples of style sheet definitions | D-6 |
| Modify a predefined style sheet | D-7 |
| Creating form preferences for new business objects | D-8 |
| Verify the registration of forms in the rich client interface | D-9 |
| Teamcenter utility and preferences | D-10 |
| Summary | D-16 |
| Key preferences reference | E-1 |
| Key preferences reference | E-2 |
| Additional Teamcenter security | F-1 |
| Introduction to Teamcenter security administration | F-2 |
| Teamcenter security applications | F-3 |
| Basic concepts | F-4 |
| Teamcenter object model hierarchy | F-5 |
| Authentication | F-6 |
| Authorization | F-7 |
| Controlling access to working data | F-8 |
| Controlling access to in-process data | F-9 |
| Authorized data access | F-10 |
| Configuring authorized data access (ADA) | F-11 |
| More about configuring ADA | F-12 |
| Basic tasks for configuring and administering ADA | F-13 |
| Enable authorized data access | F-14 |
| Define ITAR clearance and classification levels | F-15 |
| For ITAR/IP Admin users, assign the role and grant privileges .. | F-16 |
| Assign geographic locations to sites and nationality to groups .. | F-17 |
| Create an authorized data access license | F-18 |
| Assigning government classification values to data objects | F-19 |
| Associate licenses with data objects | F-20 |
| Activities | F-21 |
| Summary | F-22 |
| Structure Manager revision rules | G-1 |
| Revision rules review | G-2 |
| Revision configuration terms | G-3 |
| Understanding revision rules | G-4 |
| Revision Rule Definitions | G-5 |

| | |
|---|----------------|
| Configuring privileged and unprivileged users | G-7 |
| View or set a revision rule | G-8 |
| Revision rules setup | G-10 |
| Create, modify, or delete revision rules | G-11 |
| Create new revision rule | G-12 |
| Working entry | G-13 |
| Status entries | G-14 |
| Override entry | G-15 |
| Date and Unit entries | G-16 |
| Precise and Latest entries | G-17 |
| Ordering and grouping rule entries | G-18 |
| Setup considerations | G-19 |
| BOM precision preference | G-20 |
| Activities | G-21 |
| Summary | G-22 |
| Administering workflow processes | H-1 |
| Administering workflow processes | H-2 |
| Releasing data with a process | H-3 |
| Troubleshooting your workflow | H-9 |
| Key workflow utilities | H-14 |
| Activities | H-15 |
| Summary | H-16 |
| 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. You will also understand what *administrative tasks* are done using the Business Modeler IDE and Teamcenter rich client interfaces.

Data Model administration

- To work with data model files and *template* projects.
- To perform data model changes with *live data model updates*.
- 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 create and attach a *list of values (LOV)* to a property.
- To create and configure *options*.
- To create *rules* for *business objects*.

Application administration

- To create an *organization*.
- To configure *access permissions* and *projects* to control access.
- To configure the working environment and manage *preferences*.
- To create *saved queries* and *report definitions*.
- To create and manage *workflow process templates*.
- To create and work with *PLM XML import and export*.

Key benefits

Key benefits for completing the course objectives include:

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

Prerequisites

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

Audience

The audience for this course includes:

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

Learning tracks

Learning tracks for Teamcenter are found on the Siemens PLM Software training Web site:

<http://training.industrysoftware.automation.siemens.com/tracks/index.cfm>

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 CTRL+F1.
- 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→Current Application** or press the F1 key.

Note

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

Training scenario

This course uses the Classic Car Company scenario as background for administering and using Teamcenter in a training environment.

Where the *Using Teamcenter* emphasizes end user task scenarios, this course emphasizes administering Teamcenter task-based scenarios.

Classic Car Company (CCC)

CCC has been in business for 20 years as one of the first companies to successfully design and sell classic kit cars. Because of the increased demand for their product, the company has grown from 5 employees to more than 80 employees. The company operates out of offices in the United States and Europe.

CCC designs and manufactures a variety of classic cars in the following categories:

- High Performance – racing car
- Best of Show – show car

CCC currently delivers 100 cars per year; the process takes 18 months from concept to delivery. CCC wants to:

- Increase their output.
- Reduce time to market.
- Reduce development costs.

Classic Car Company (CCC) business case

CCC began to experience problems managing their large amounts of data and tracking project and assembly information. Problems they began to experience include:

- Lost data.
- Duplicate data resulting in higher labor costs.
- Working with the wrong revision of the part causing assembly problems and quality problems.
- Lengthy product development cycles.
- Poor communication, both internally and externally.

CCC needed a tool to allow their employees to share accurate, up-to-date information at any point in the product development life cycle. Consultants recommended they investigate the use of a product lifecycle management system to track the data and information required to design, manufacture, and support their products.

CCC chose Teamcenter as their tool; the Teamcenter software was recently installed.

The Teamcenter administrator manages the Teamcenter data model. CCC has decided to use as much standard functionality as possible, but some extensions to the data model were performed to suit CCC business processes.

Lesson

1 *Business Modeler IDE* fundamentals

Purpose

The purpose of this lesson is to explain important fundamentals of the Business Modeler IDE.

Objectives

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

- Describe a Business Modeler IDE project.
- Describe the Business Modeler IDE process.
- Navigate the Business Modeler IDE interface.
- Create a project in the Business Modeler IDE.
- Work with Business Modeler IDE template files.
- Set up and test a connection to Teamcenter.

Help topics

Additional information for this lesson can be found in:

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

1.1

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
- Perform C++ customizations.
- Migrate data using the Mapping Designer.

Note

- 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>

1.1.1 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.

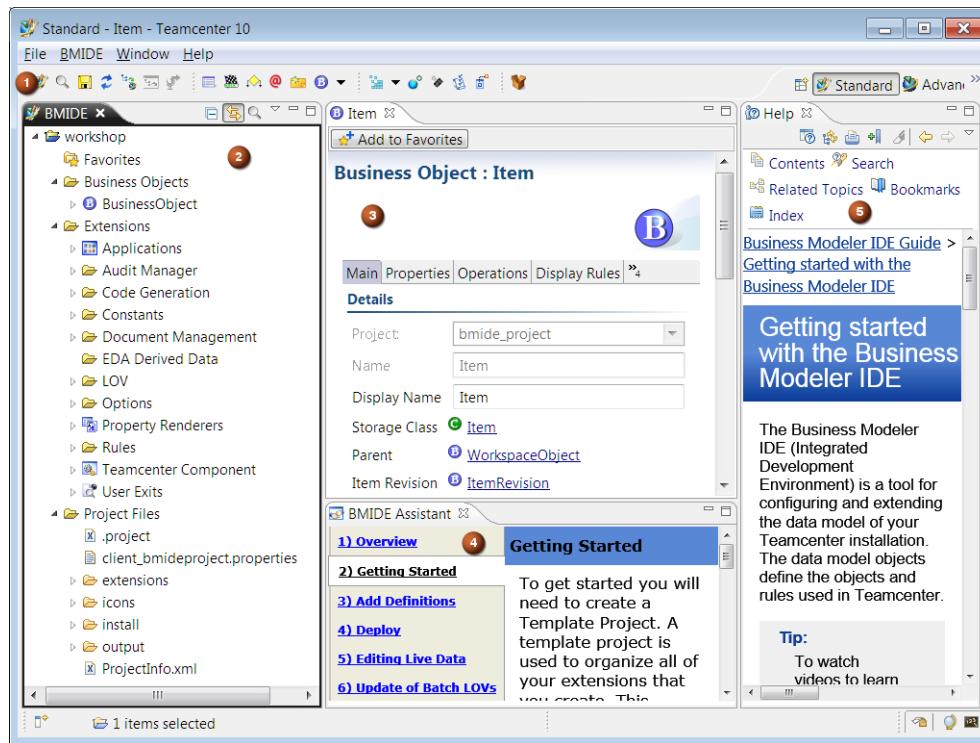
The user interface is comprised of two perspectives, **Standard** and **Advanced**.

To open one of these perspectives, choose **Window→Open Perspective**.

| Perspective | Description |
|--------------------------------|--|
| Standard perspective | BMIDE view, which provides a centralized location for favorites, business object elements, and project files. Toolbar contains buttons for data model actions. Editor that is more narrow than the advanced perspective. Console view that appears on the right of the standard perspective. |
| Advanced perspective | Tabs that show individual trees for the Business Objects , Classes and Navigator (project files). Tabs that show Extensions tree, the Outline and the Console views. Toolbar contains buttons for data model actions and C++ customizations. Editor that provides area needed configuration of complex elements. Console view that appears on the lower left of the advanced perspective. |

1.1.2 Standard perspective

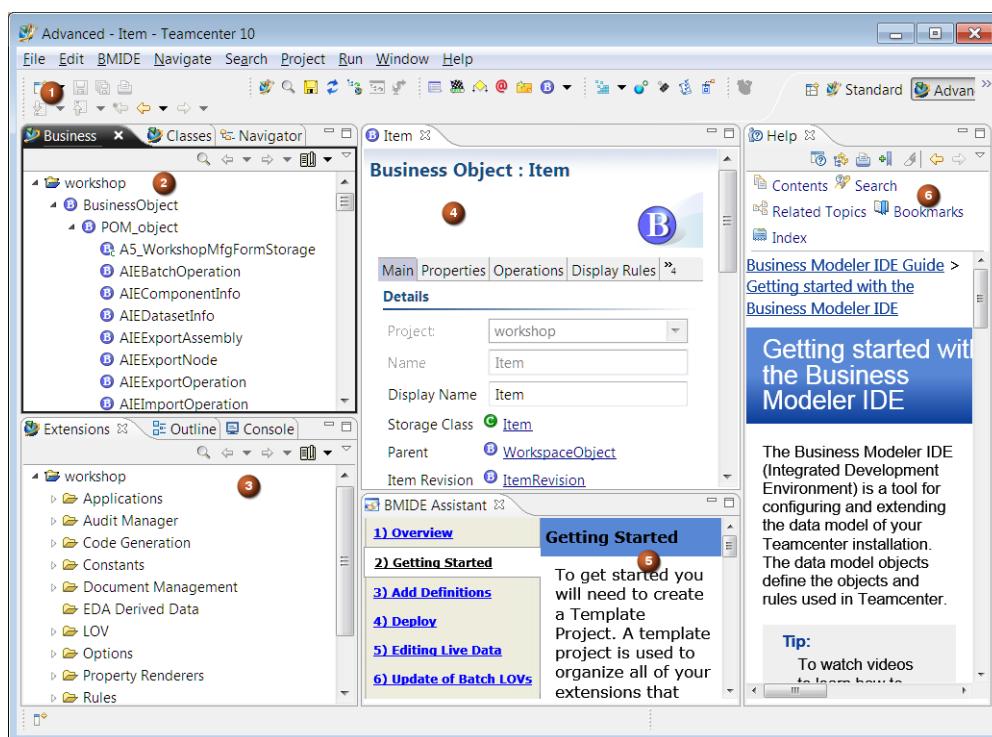
The **Standard** perspective provides a simplified user interface. It contains the **BMIDE** view, which provides a centralized location for favorites, data model elements, and project files.



- | | |
|--------------------------------------|---|
| 1 Toolbar | Contains buttons for the most commonly used actions. |
| 2 BMIDE view | Provides a single view for favorites, data model elements, and project files. |
| 3 Editor | Allows users to edit data model elements. |
| 4 BMIDE Assistant view | Helps new users get started using the Business Modeler IDE. |
| 5 Help view | Provides online help for the Business Modeler IDE. You can launch this view by pressing the F1 key or clicking the question mark button ? in the lower left corner of any dialog box. |

1.1.3 Advanced perspective

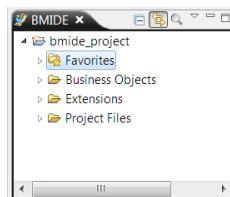
The **Advanced** perspective provides a full-featured user interface. It contains separate views for business objects, classes, project files, and data model extensions.



- | | |
|---------------------------------------|---|
| 1 Toolbar | Contains buttons for the most commonly used actions. |
| 2 Business Objects view | Displays business objects, the fundamental objects that model business data. |
| 3 Extensions view | Displays extensions to the data model. |
| 4 Editor | Allows users to edit data model elements. |
| 5 BMIDE Assistant view | Helps new users get started using the Business Modeler IDE. |
| 6 Help view | Provides online help for the Business Modeler IDE. You can launch this view by pressing the F1 key or clicking the question mark button ? in the lower left corner of any dialog box. |

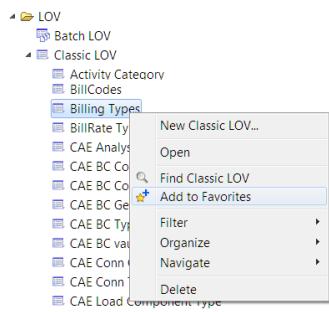
1.1.4 Interface favorites and filtering

The **Favorites** folder holds data model elements for quick access.

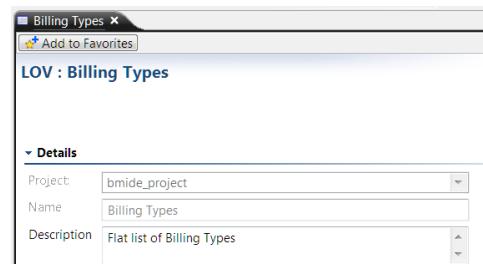


Favorites folder

To add elements to the **Favorites** folder, right-click an element and choose **Add to Favorites** from the context menu, or click the **Add to Favorites** button on an editor.



Add to Favorites from context menu



Add to Favorites from editors

Filtering hides the selected element category or only hides COTS (commercial-off-the-shelf) elements. This reduces screen clutter.

- Hide elements

Right-click a folder and choose **Filter→Hide All COTS element-name elements** or **Hide this folder and all element-name below**.

- Customize hidden groups

Choose **Filter→Customize Hidden Groups** to quickly hide entire categories of element types.

- Configure business objects to display

To filter business objects, right-click the **Business Objects** folder and choose **Filter→Configure Business objects to display**. Select the **Customize Business Objects to Show** check box to display only the selected business object types.

1.2

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).

1.2.1 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.

1.2.2 Objects to be created by the Business Modeler IDE

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. |
| 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. |

1.3

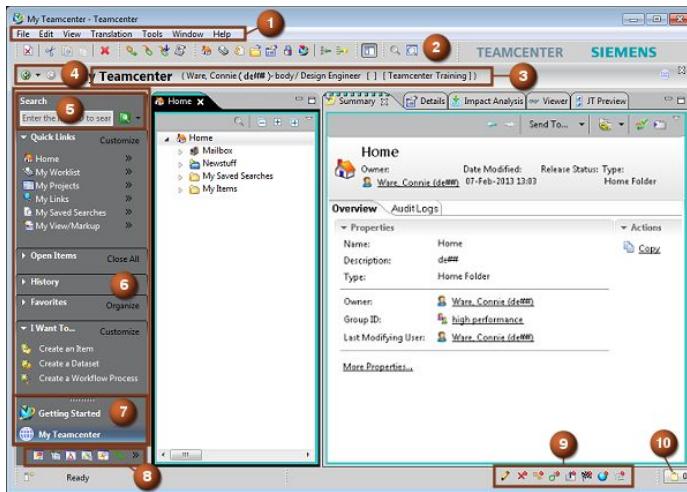
Overview to administering Teamcenter

Teamcenter administration commonly involves more than one individual. This is intended to provide an example of those individuals and tasks.

| Teamcenter administrator | Tasks |
|---------------------------------|---|
| Data model administrator | Using the Business Modeler IDE interface, you define Teamcenter business objects, properties, and business rules based on your set of functional requirements. |
| Application administrator | Using the Teamcenter rich client interface, you define your organization in Teamcenter, establish access controls for your data, create queries and reports, build your workflow processes, set up projects, and other administrative application administration tasks. |
| System administrator | Using the Teamcenter Environment Manager (TEM) interface, you install the template package to a production environment. |

1.3.1 Rich client interface

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



- | | | |
|---|--------------------------------|--|
| 1 | Menu | Each perspective provides some common menu commands and other menu commands specific to the perspective. |
| 2 | Toolbar | Each perspective provides some common toolbars specific to that perspective. |
| 3 | Application banner | The application banner shows the name of the active perspective and lists the current user and role. You can 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. |
| 4 | Back and Forward | The Back and Forward buttons allow you to move between loaded Teamcenter perspectives. The small arrows next to the buttons let you select from the list of currently loaded perspectives. |
| 5 | Quick Search box | The Search box provides predefined quick searches using dataset, item ID, item name, keyword search, and advanced search features. |
| 6 | Navigation pane | The navigation pane provides quick access to the data you use most. In addition to finding, organizing, and accessing your data, you can configure the display of the Teamcenter perspective buttons in the navigation pane to display only those perspectives that you use regularly to perform your tasks. |

| | | |
|----|------------------------|---|
| 7 | Primary applications | Primary application buttons provide access to your most frequently used Teamcenter perspectives. |
| 8 | Secondary applications | Secondary application buttons provide access to Teamcenter perspectives you use infrequently. |
| 9 | Information center | Information center symbols convey where-used and where-referenced, access privilege, child count, and status information about the selected object. To display the information, point to the symbol. The information displays in the form of a tooltip. |
| 10 | Clipboard button | The clipboard button displays the Clipboard Contents dialog box, which contains references to objects that have been cut or copied from your workspace. The total number of objects on the clipboard is displayed to the right of the symbol. |

1.3.2 Teamcenter command prompt interface

Teamcenter administrators usually run one or more Teamcenter utilities to perform their tasks. The Teamcenter utilities are run from the Teamcenter command prompt interface like the following.

Note

es1_es1 represents the node ID and depends on the installation.

```

es1_es1 Command Prompt

D:\es1_root\tc_menu>SET TC_ROOT=D:\es1_root
D:\es1_root\tc_menu>SET TC_DATA=D:\es1_data
D:\es1_root\tc_menu>D:\es1_data\tc_profilevars

D:\es1_root\tc_menu>set TC !more
TCFS_Service=tcfss
Tcfss_svc_tcfss=1528
TC_BIN=D:\es1_root\bin
TC_DATA=D:\es1_data
TC_MODEL=D:\es1_data\model
TC_CONNECT=infodba:iGX19B1_g7A@es1
TC_DISABLE_FLASH=true
TC_HELPFILE=D:\es1_root\web\htdocs\tchelp\tchelp.hlp
TC_HTDOLCS=D:\es1_root\web\htdocs
TC_INCLUDE=D:\es1_root\include
TC_INSTALL_DIR=D:\es1_root\install
TC_LIBRARY=D:\es1_root\lib
TC_LOG=D:\es1_data\log_dtc1xxx_es1
TC_MSG_ROOT=D:\es1_root\lang\textserver
TC_NX_DEFAULT_PART=D:\es1_data\blank_english.prt
TC_PRINTER=lpr.exe
TC_ROOT=D:\es1_root

```

Teamcenter utility programs are run from the Teamcenter command prompt interface because they require Teamcenter environment variables. Other Teamcenter environment variables are propagated in the Teamcenter command prompt interface with **TC_DATA\tc_profilevars.bat**.

To open the Teamcenter command prompt interface with the Teamcenter environment variables in a Windows operating system, choose the following:

Start→All Programs→Teamcenter 10→es1_es1 Command Prompt

Configure the Teamcenter environment

To manually configure the Teamcenter environment, run the **TC_DATA\tc_profilevars.bat** batch file. This batch file is called automatically when exiting to an MS-DOS shell from the Teamcenter menu, but the environment can also be set manually. To manually set the Teamcenter environment, enter the following commands:

```

set TC_ROOT=D:\es1_root
set TC_DATA=D:\es1_data
call %TC_DATA%\tc_profilevars

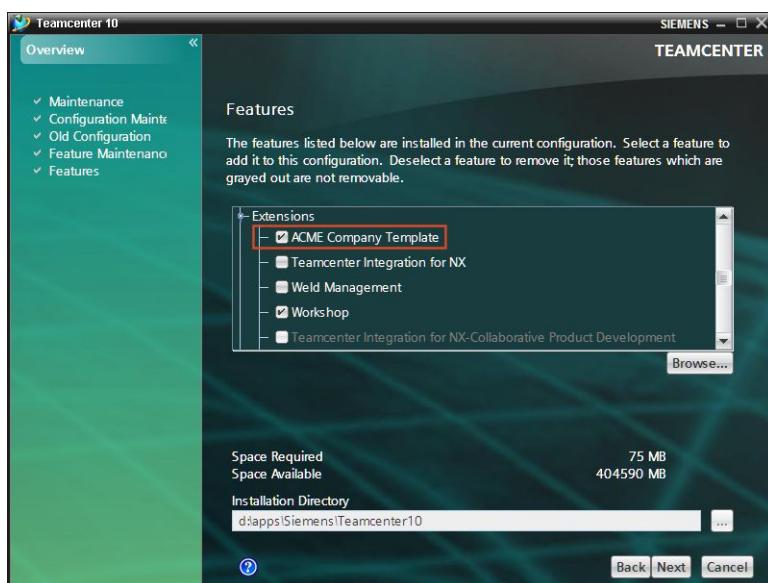
```

These folders are set up when Teamcenter is installed. Teamcenter installations differ.

- **TC_ROOT**
- **TC_DATA**

1.3.3 Teamcenter Environment Manager (TEM) interface

A system administrator will install the template package extensions to a production environment using the Teamcenter Environment Manager (TEM) interface.



Basic steps

1. Copy the template files from the **packaging** directory on your Business Modeler IDE client to a directory that is accessible by the server.
2. Select the new template in the **Features** panel.
3. To install the new template, click **Start** in the **Confirmation** panel.
4. 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.

1.3.4 Organization for this course

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.

| Group: dba | | |
|-------------------|----------------|---|
| Role | User | Description |
| DBA | jgordon | User with system administrative privileges. |

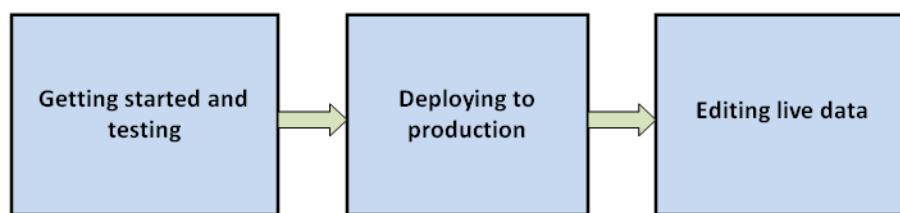
| Group: Engineering | | |
|---------------------------|----------------|--|
| Role | User | Description |
| Engineer | jgordon | User to test display rule. |
| Designer | twhite | Regular user with end user privileges. |

| Group: Simulation Administration | | |
|---|----------------|--|
| Role | User | Description |
| Simulation Administrator | jgordon | User to test condition on a naming rule pattern. |

1.4

Business Modeler IDE process overview

The Business Modeler IDE is a tool that shows you the business objects, properties and business rules that control the system. You can use the Business Modeler IDE to extend Teamcenter with your own business objects, properties, and business behavior. When you extend Teamcenter using the Business Modeler IDE, you should follow best practices for storing extensions into a template, testing the template in a test Teamcenter database, and then packaging and deploying the validated template to your production Teamcenter site. Later you may want to provide updates to a live running production site. All of these best practices have established processes for you to follow. The Business Modeler IDE provides tools to assist you through these tasks so that these steps are easy to manage.



Business Modeler IDE process flow

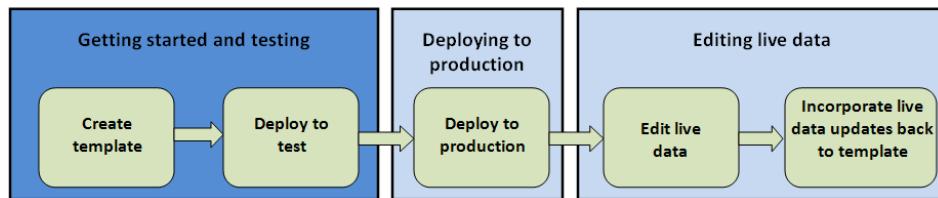
Note

How is this high-level Business Modeler IDE process used in this course?

- The early lessons and activities use **Getting started and testing**.
- Within the early lessons there is explanation and an activity on **Deploying to production**.
- Later in the course, a lesson and activities use **Editing live data**.

1.4.1 Developing extensions and testing

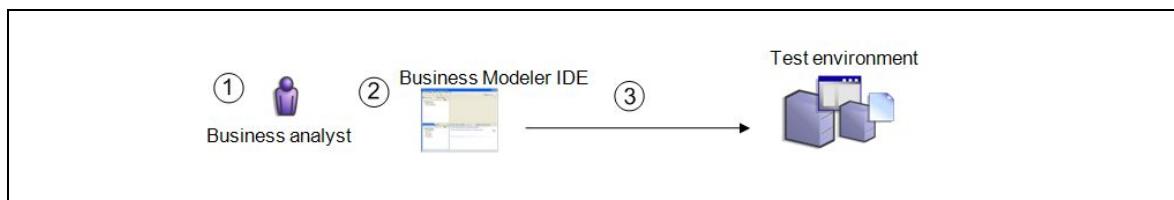
The Business Modeler IDE and template are considered to be your development environment. You can deploy this template as a single unit to any Teamcenter database server for testing or production usage. The deployment process ensures that all of your extensions are added to the default Teamcenter configuration.



First, create a new template project so that you can store your extensions in it for easy deployment to database sites. Then, perform a live deploy to push your template to a test Teamcenter server where you can validate it in a safe environment.

Typically, during this phase of development, you add new business objects and properties in addition to business behavior. Business objects and properties are considered schema changes and require a production Teamcenter server to be shut down and all users logged off to make the update. However, you can update a test server with a live deploy as long as you are the only user logged on to the system.

Follow this process:



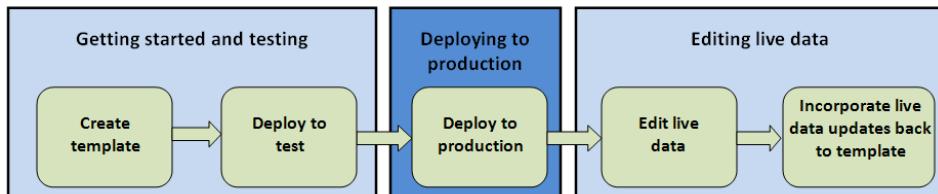
Developing extensions and testing

1. Create a Business Modeler IDE template project if you have not already done so.
2. Use the Business Modeler IDE to create new extensions to the data model and behavior.
3. Perform a live deploy to push your template to a test environment and validate your extensions.

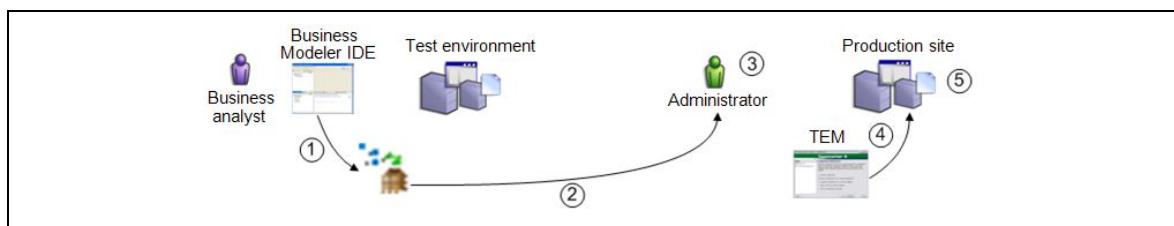
To continue adding new extensions repeat steps 2 and 3.

1.4.2 Deploying a template to a production site

After you validate your template in the test environment, you can update your production environment. Use the Business Modeler IDE to create a template package, shut down the production server, and use Teamcenter Environment Manager (TEM) to install the template package.



Follow this process to update your production site with your template.



Deploying a template to a production site

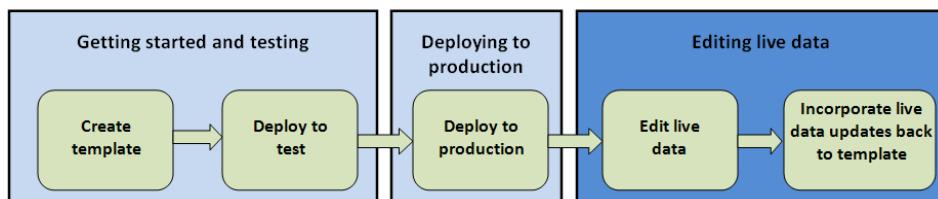
1. After you validate your extensions in a test environment, use the Business Modeler IDE to build a template package.
2. Send the template package to the production site administrator.
3. Log off all users and shut down the production Teamcenter site.
4. The production site administrator uses TEM to install the template package into the production site.
5. The production site administrator starts up the server at the production site and allows users to log on.

If you add more extensions later and have validated the template in a test environment, you can use TEM to update your template at the Teamcenter site by using the **Teamcenter Foundation→Update the database** menu commands in the **Feature Maintenance** panel.

1.4.3 Editing extensions in a live production site

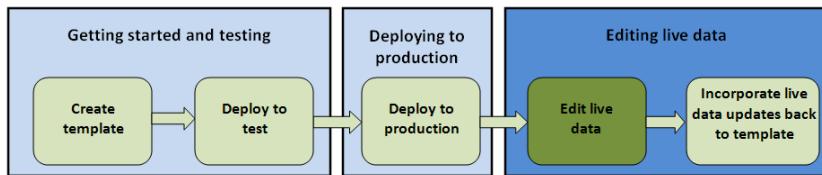
After you deploy your extensions to a production server, you may want to update some of your Business Modeler IDE extensions. For example, you may want to add a new value to a list of values (LOV). This type of update is called a *live update* because you are updating the data on a live system. Typically, you make this change directly to the extensions that are already deployed to the production database so you do not have to shut down the Teamcenter server. Therefore, only nonschema type changes, such as LOVs, statuses, units of measure, business rules, and so on, can be made at this time.

Follow this process to edit your extensions deployed to a live production environment.

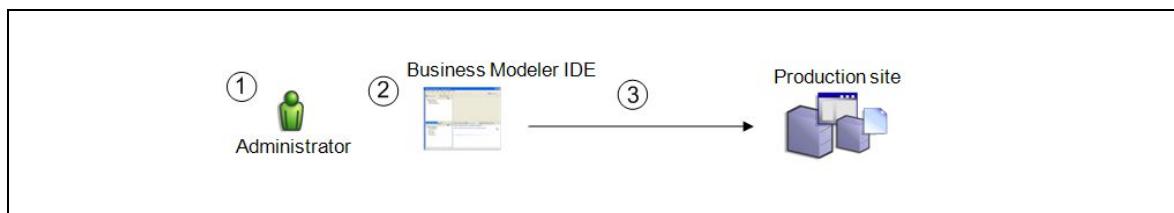


The Business Modeler IDE provides the live update project that points directly to the Teamcenter production database and allows you to make updates to your existing extensions in your deployed template. These updates are made to the extensions stored in the Teamcenter database; they do not affect the development environment template. Do not make the updates to your development environment template at this point. The development environment may contain ongoing work as you prepare for your next system downtime. This ongoing work may contain schema changes intended for the test environment, and it is difficult to separate them from the LOVs and status changes intended for the production site.

1.4.4 Edit live data



Once the production site is running, you can edit the live data at any time. First, you must configure your Business Modeler IDE to point to the production site, and then you can edit the data.

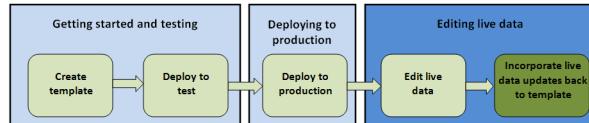


Editing live data

1. Configure your Business Modeler IDE to point to a production site and create a live update project so that it can download the template data.
2. Use the Business Modeler IDE to create new extensions to the data model and behavior.
3. After you finish making changes, click the **Deploy Template** button  on the toolbar to push these changes to the production site.

To continue adding new extensions repeat steps 2 and 3.

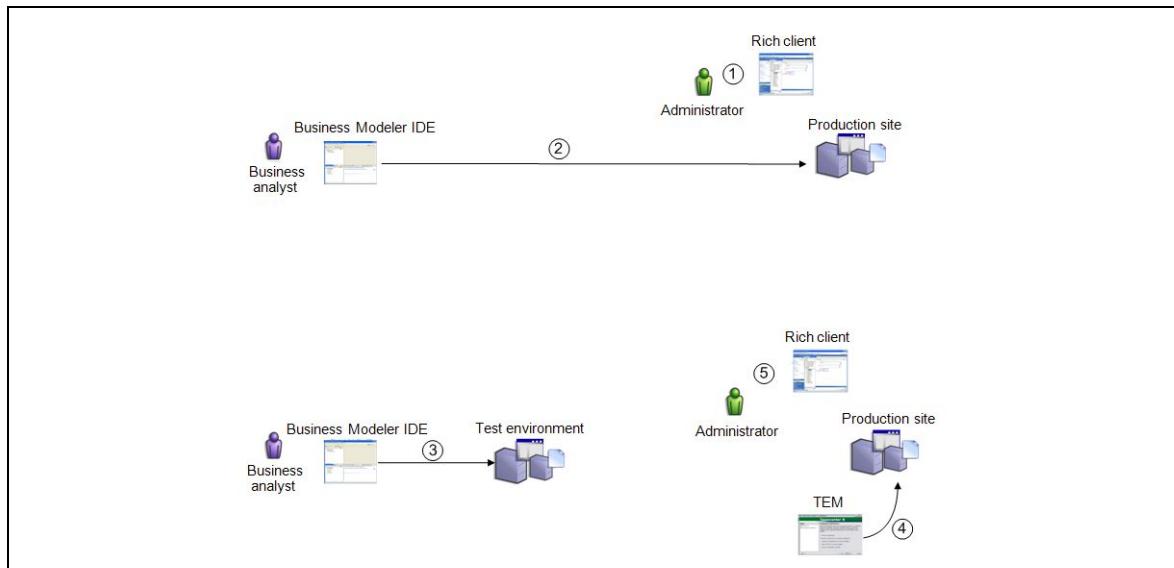
1.4.5 Incorporate live data updates from the production site



Eventually, you must incorporate all the live data updates performed at the production site back into the development environment. This event occurs as you prepare for your next system downtime.

If you do not perform this step, and you deploy your development environment template (which does not include the latest live data updates) to the production site, the live data updates already present in the production site are removed. Because this is undesirable, the system blocks the deployment. To ensure that the deployment is not blocked, you must perform this step.

To incorporate the latest live data, you can run a wizard from the development environment Business Modeler IDE to log on to the production Teamcenter server and retrieve the latest extensions. Or, if your production site is offline from your Business Modeler IDE, you can ask an administrator to run a command line to extract the latest updates from the site and send them to you for incorporation. In either case, once the extensions are incorporated, you can continue with testing, packaging, and deploying to your production site.



Incorporating live data updates from the production site

1.5 Development environments

Single production database environment

The first scenario is the simplest. It consists of a single production database and the Business Modeler IDE client. In this environment, a single user uses the Business Modeler IDE client to create a template for extending the data model. The production database may support one to a small number of users. If no other users are logged on to Teamcenter, the Business Modeler IDE user can log on Teamcenter to establish the connection and then live update the template from the Business Modeler IDE.

This scenario is ideal for a small community of users.

Caution

One disadvantage of this scenario is that the extensions cannot be tested in a safe environment before going live with the production environment.

Test and production database environment

This scenario is very similar to the first scenario except that it adds a test environment for the Business Modeler IDE user. In this scenario, the Business Modeler IDE user creates a template for extending the data model. The Business Modeler IDE user live updates the template to a test database, where the extensions can be tested.

After the extensions are tested to a satisfactory level, the user community is notified of a system downtime. When the system is shut down, use the package wizard in the Business Modeler IDE to build the template feature, and then use Teamcenter Environment Manager (TEM) to install or update the feature into the production environment.

This scenario offers a safer process for extending, testing, and rolling out to production than the first scenario. Extensions can be created, deployed, and tested in the safe test environment and the process is reiterated until requirements have been met. This process protects the production environment from any undesirable changes or stability issues, thus keeping the user community running while reiterating the development and test process.

User testing environment

This environment is very similar to the previous scenario except that it adds a third environment specifically for user testing or user training. In this scenario, the Business Modeler IDE user creates a template for extending the data model. The Business Modeler IDE user live updates the template to a test database, where the extensions can be tested.

After the extensions are tested to a satisfactory level, the company is ready to roll out these changes. Before rolling the template out, you can set up a safe environment for training or for user acceptance testing. Use the package wizard to build the template feature, and then use Teamcenter Environment Manager (TEM) in the user testing environment to install or update the template.

This scenario has the same benefits of a reiterative process as the previous scenario. It also adds the ability to let the user community test or train in a safe environment without impact to the real data in the production system. The user testing environment can be a useful place to work out any issues between requirements and design. The training environment allows the user community to get comfortable with any new processes, objects, or behaviors before going live with the production environment.

Multiple developer environment

This scenario is similar to the previous scenarios with the added requirement that there are two or more users of the Business Modeler IDE working on the template, rather than only one. In this scenario, many users are contributing extensions to the same template. For two or more developers to work on a template project concurrently, a source control management (SCM) system must be connected to the Business Modeler IDE client. Examples of SCM systems include: CVS, Subversion, ClearCase, Perforce, and Visual Source Safe. By default, the Business Modeler IDE is equipped to connect to a CVS repository.

In this scenario, one Business Modeler IDE user should create the template project and add the project directory and all of its subfolders and files into the SCM. This is usually performed by checking all files and directories. Next, all other Business Modeler IDE users should synchronize their SCM repository to get the project files and directories downloaded to their machine and then import the project into the Business Modeler IDE using the import wizard.

After all extensions have been tested in the integration environment, the template feature can be installed into the user testing environment and production environments using TEM.

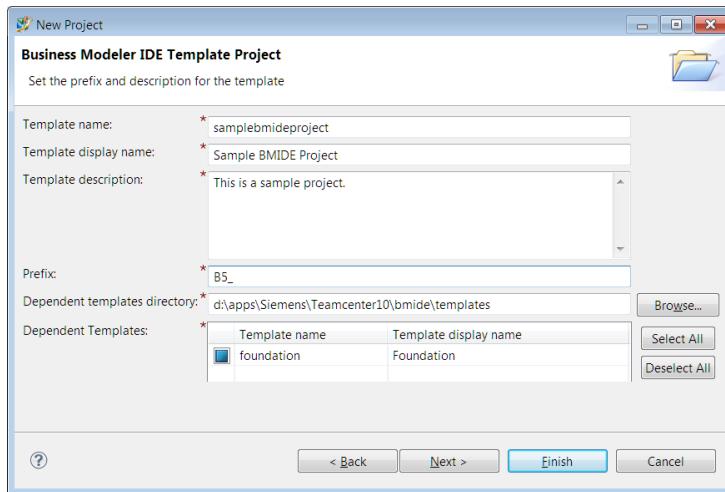
1.6 A Business Modeler IDE template project

A Business Modeler IDE *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. Before you can extend the Teamcenter data model, you must create a template project. You can also import an existing template project.

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

- ▲ **CCC_Corporate**
 - ▲ **extensions** ← Template source files
 - ▲ **lang** ← Language (display) source files
 - ccccorporate_template_en_US.xml**
 - default.xml**
 - dependency.xml**
 - master.xml**
 - ▷ **icons** ← Icon graphics and source files
 - ▷ **install** ← Install and upgrade support files
 - ▷ **output** ← Package files generated from the template source files
 - .project**
 - client_ccccorporate.properties**
 - ProjectInfo.xml**

1.6.1 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. Define the following project properties by following the wizard.
 - The **Project name** (do not use spaces).
 - The **Location** for your project files. Use the default location or specify a location for your project files. The destination directory must be named the same as the project.
 - **Template name** defaults to your project name. This is the name of the template file created when this project's extensions are packaged for distribution.
 - **Template display name** defaults to your project name. This is the template name that appears in Teamcenter Environment Manager when this template is installed on other servers.
 - The **Template description** is the description that appears in Teamcenter Environment Manager.
 - The project **Prefix**, which is a 2- to 4- character prefix that will be affixed to the name of all objects created in the template to ensure unique naming. The first character must be an uppercase letter, and the second must be a number between 4 and 9.

4. Choose the **Dependant templates** directory where the data model template XML files are stored.
5. Select the **Dependant Templates**. At a minimum, select the **Foundation** template.

Note

Make sure that you select the same templates that are on the Teamcenter server.

6. Select the languages in the **Locales Selector** dialog box if you want the display names of data model you create in the template to be viewable in different languages in the Teamcenter end-user interface.
7. If you plan to write code, set up the location of generated source files in the **Code Generation Information** dialog box.
After you click **Finish**, the new project loads and appears in your perspective.
8. View the business objects in the data model.

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

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

1.6.2 Naming objects

When you create new data model objects, Siemens PLM Software requires that you add a prefix to the name to designate the objects as belonging to your organization. The prefix is set with the **Prefix** box when you create a template project. Thereafter, whenever you create new objects in that project, the same prefix is added to the name of all objects.

For example, if the prefix is set as **A4_** when the project is created, when you create a new **Item** class, you could name it **A4_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 requires that you apply this consistent naming convention to all your new data model.

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).

Use following conventions when choosing a prefix:

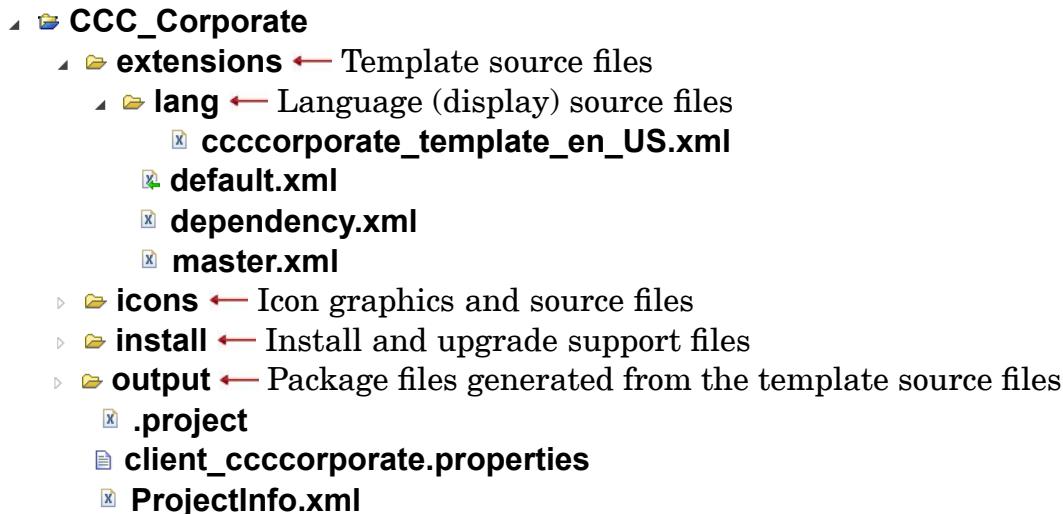
- Ensure the prefix contains two to four characters. Underscores are allowed.
- Place an uppercase letter in the first character position.
- Place a digit in character position two, three, or four.
- You can use digits 4–9. Do not use digits 0, 1, 2, or 3 because 0 and 1 are reserved for Siemens PLM Software templates, and 2 and 3 are reserved for third-party template development.

Following are examples of acceptable prefixes:

A4
B5cd
F7_
X999

1.6.3 Files in a template project

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



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.

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 within your template.

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 files

- 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.

1.6.4 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.

1.6.5 Import a template project

You can import a project from another Business Modeler IDE installation. Importing also migrates the template to the latest version of the data model.

1. Choose **File→Import**.

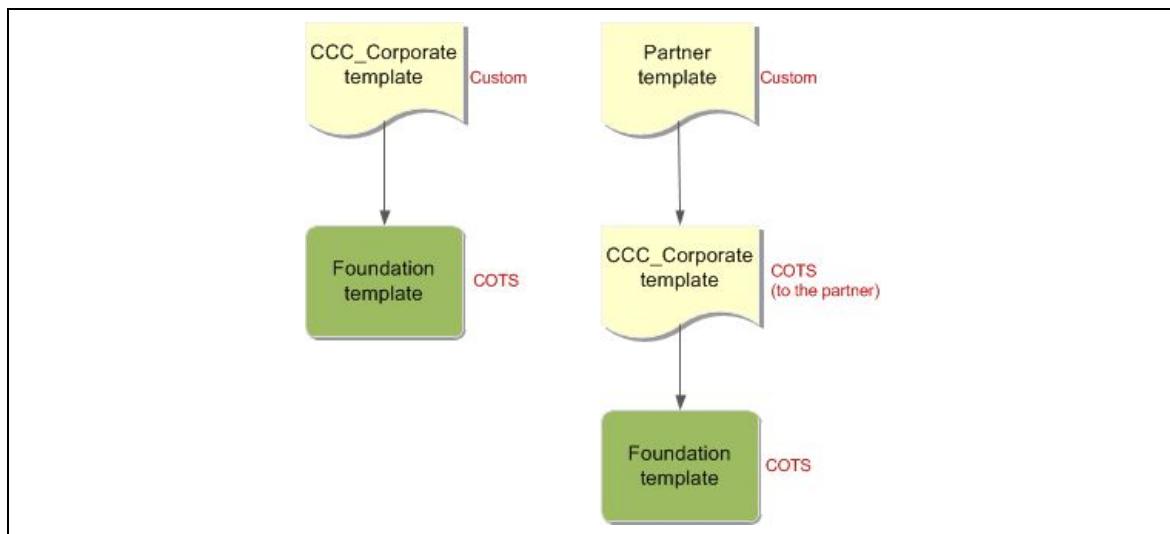
The Import wizard runs.

2. In the **Select** dialog box, choose **Business Modeler IDE→Import a Business Modeler IDE Template Project**. Click **Next**.
3. 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.
 - b. Click the arrow in the **Select active file** box to choose the extension file to make active after the project is imported.
 - c. If necessary, 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 **Finish**.
4. Create extensions using the new project just as you would a project you created yourself.

1.7 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.

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.



Custom versus COTS template example

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.

1.7.1 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 choosing **BMIDE→Deploy Template** on the menu bar. This is also known as *live update*. You can also use live update 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 **BMIDE→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.

1.7.2 Using an SCM system to manage files

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>

1.8

Configure the Business Modeler IDE

There are several 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 the active extension file.

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

3. Set Business Modeler IDE preferences.

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

4. 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 **Business Modeler IDE 2-tier** or the **Business Modeler IDE 4-tier** option under **Base Install** when you installed the Business Modeler IDE, a server connection profile is already added.

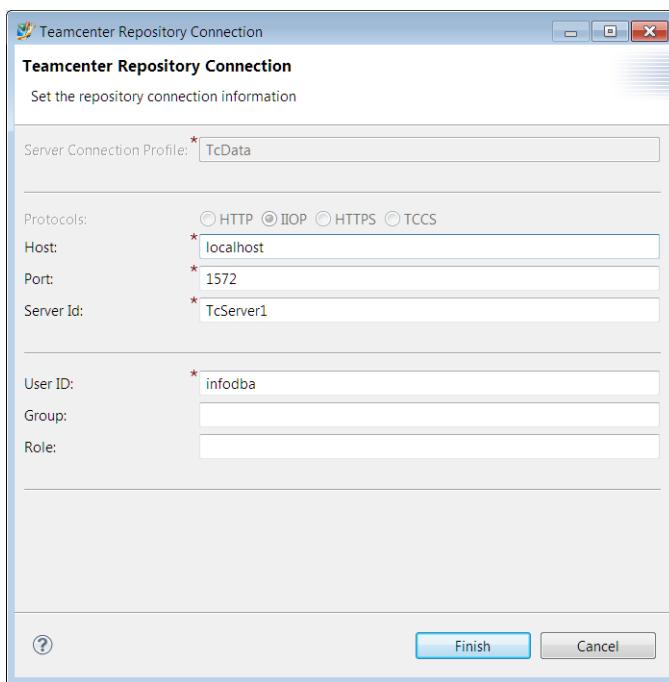
5. Schedule daily project backup.

You must back up your Business Modeler IDE project files daily so you can restore your environment in case of a deployment failure.

1.9 Add a server connection profile

A server connection profile is automatically created when you install the Business Modeler IDE. Create a server connection profile to deploy data model changes to a test server.

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



4. In the **Server Connection Profile** box, type the name of your profile.
5. For **Protocols**, select one of the following:
 - **IIOP** – Sets a network communication protocol (used in a two-tier Teamcenter environment).
 - **HTTP** – Sets a Web communication protocol (used in a four-tier Teamcenter environment).

Depending on what you selected for the protocol, enter the following:

- a. **Host** – the Teamcenter server name. For IIOP, this is typically set to **localhost**.
- b. **Port** – the Teamcenter server port (for the IIOP protocol) or the Web application server port (for the HTTP protocol). For IIOP, this is typically set to **1572**.

- c. If you selected IIOP as the protocol, a **Server ID** box appears for the server on the network, for example, **TcServer1**.
6. Add the connection profile credentials.
 - a. In the **User ID** box, type the ID of the authorized user on the Teamcenter server.
 - b. In the **Group** box, type the group the user is assigned to (for example, type **dba** for the database administration group). This is optional.
 - c. In the **Role** box, type the role the user is assigned (for example, type **DBA** for the database administrator role). This is optional.
7. Click **Finish**.

The server profile is added to the list.
8. In the **Preferences** dialog box, click **OK** to save the preferences.
9. Perform the following steps to test the connection to the server:
 - a. Start the server by launching the Teamcenter rich client or by running the **start_imr** file, for example:

```
TC_ROOT\iiopservers\start_imr.bat
```
 - b. Open the **Item** business object and click the **Display Rules** tab.
 - c. Click the **Add** button in the **Display Rules** editor.
 - d. In the **Display Rule** dialog box, 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. (You are not asked again to log on if the Business Modeler IDE needs to query the server for data. You only have to log on once per session.)
 - e. If the connection is made properly, the **Search Organization** dialog box displays the groups and roles from your server.

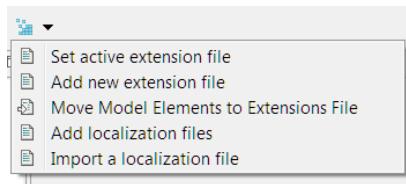
Now that you have configured the Business Modeler IDE, you are ready to extend the data model.

1.10 Introduction to extension files

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

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

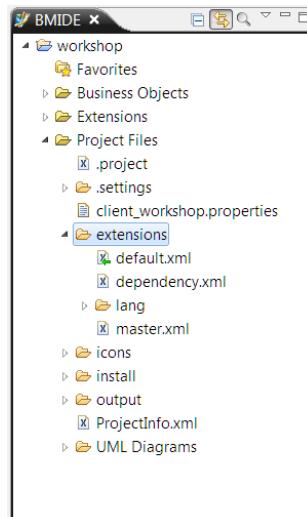
Use the **Organize Extensions** button on the toolbar to work with extension files.



You can set the extension file in two methods:

- Click **BMIDE**→**Organize Extensions**→**Add new extension file**.
- Right-click a project and choose **Organize**→**Set active extension file**.

In the **Project Files\extensions** folder, a green arrow symbol appears on the active extension file (see the following figure).



1.10.1 Add a new extension file

The Business Modeler IDE provides a default set of extension files in your project's **extensions** folder. You can create your own extension files.

1. Click **BMIDE**→**Organize Extensions**→**Add new extension file**.

Note

In the **Navigator** view, you may right-click the **extensions** folder and choose **Organize**→**Add new extension file**.

2. In the **File name** box, type the name of your new extension file.
3. Confirm the **Set as active extension file** box is selected .
4. Click **Finish**.

Move a custom object to an extension file

If you create other extension files besides the **default.xml** file, you can always move a custom object to another extension file. Custom objects display a subscript **c** symbol indicating they are custom.

1. Right-click the custom object you want to move.
2. Choose **Organize**→**Move to extension file**.
3. Choose the extension file and click **OK**.
4. Choose **BMIDE**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

The custom object's XML nodes are moved from their original extension file to the target extension file.

1.10.2 Set an active extension file

1. Open one of the following views: **Business Objects**, **Classes**, or **Extensions**.
2. Set the active extension file.
 - a. Choose **BMIDE**→**Organize Extensions**→**Set active extension file**.
 - b. Select the file you want to save your data model extensions.

Note

To see which file is currently selected as the active extension file, access the **Navigator** view, open the project, and expand the **extensions** folder. A green arrow symbol appears on the active extension file.

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

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

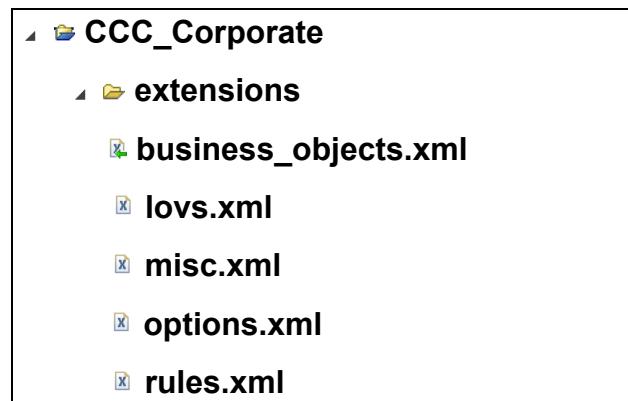
1.10.3 Extension file example

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.

Example 1

You could organize your extensions generally in one of the following areas with the files matching the types of business data.

- Business objects
- LOVs
- Miscellaneous
- Options
- Rules

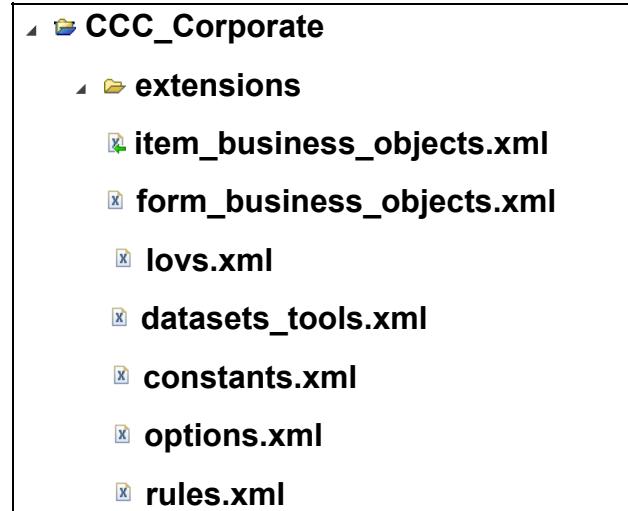


1.10.4 Extension file example

Example 2

You could organize your extensions more specifically matching your site data extensions.

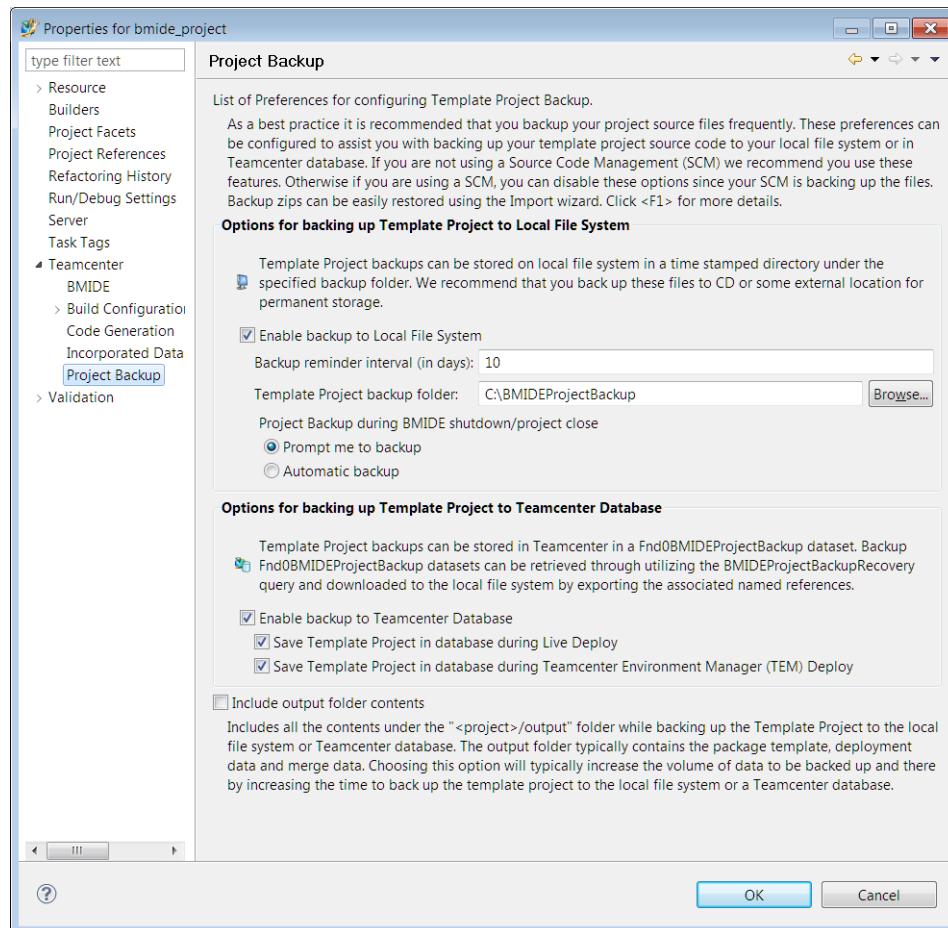
- Item business objects
- Form business objects
- LOVs
- Dataset business objects and Tools
- Constants
- Options
- Rules



1.11 Project backup

Maintain a regular backup of all relevant data so that you can restore data to the state it was in prior to failure. For example, when deployment of a template to a database fails, you must restore data to the state it was in prior to failure.

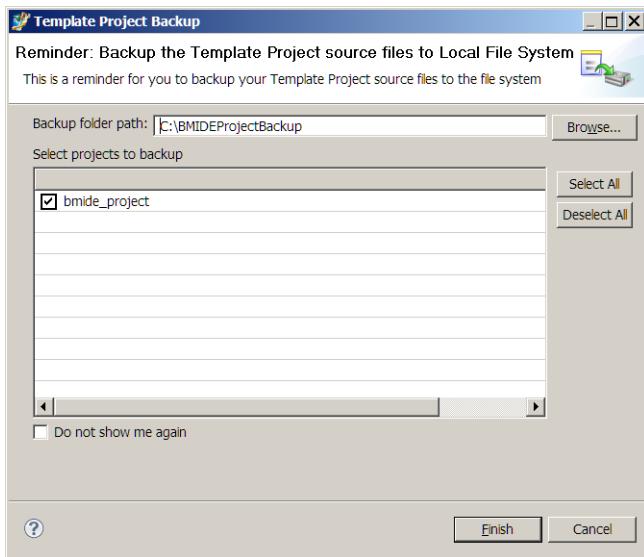
Use the **Project Backup** dialog box to set project backup. To access this dialog box, right-click the project and choose **Properties**→**Teamcenter**→**Project Backup**.



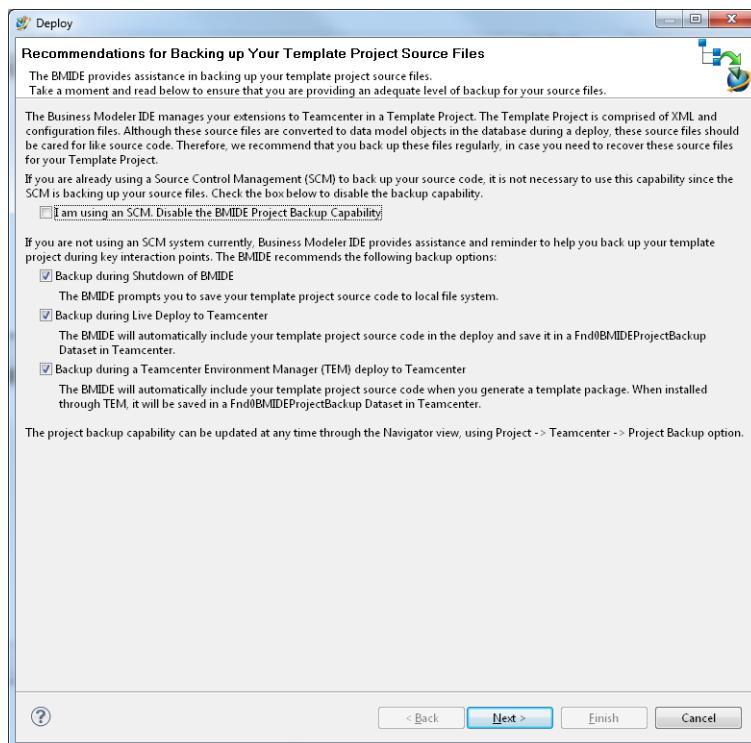
By default, projects are backed up to your local computer when you close a project or shut down the Business Modeler IDE (**Enable backup to Local File System** check box), and backed up to the database when you deploy the project to a server (**Enable backup to Teamcenter Database** check box).

1.11.1 Back up project data

When you close a project or shut down the Business Modeler IDE, execute the backup using the following dialog box.

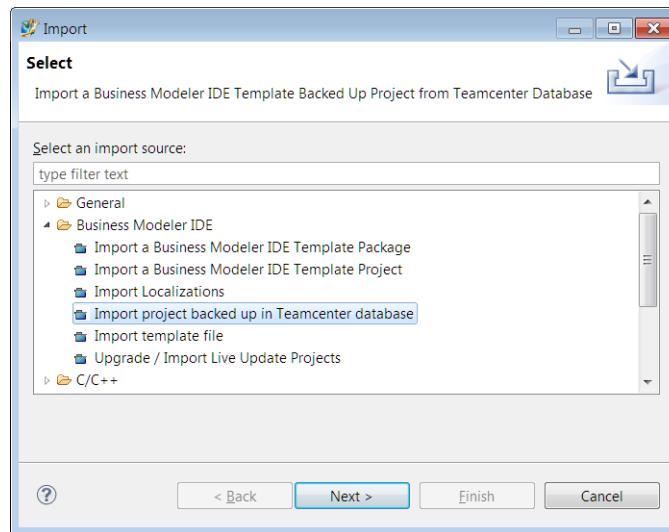


When you deploy the project to a server, execute the backup using the following dialog box.



1.11.2 Restore project data

To restore the backed-up project from the database, choose **File→Import** and in the **Import** dialog box, choose **Business Modeler IDE→Import project backed up in Teamcenter database**.

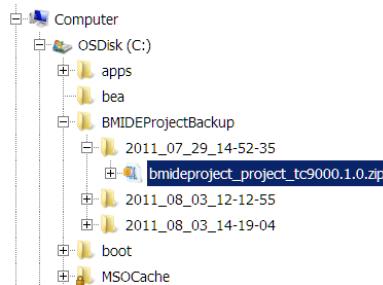


1.11.3 Restore alternatives

If importing the project backup does not work, you can try the following alternative methods:

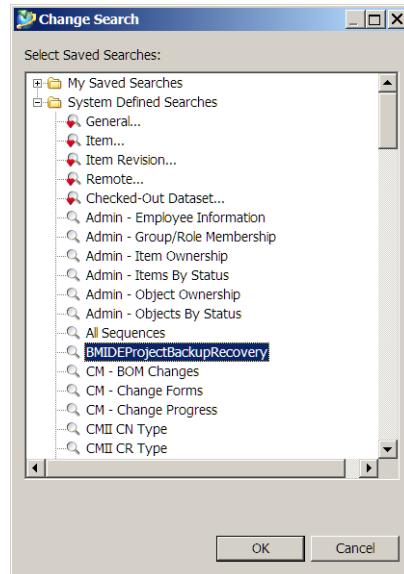
- Local backup

The backed-up projects are saved on your hard disk (by default in the **C:\BMIDEProjectBackup** directory).



Unzip the retrieved project backup ZIP file and import the project into the Business Modeler IDE.

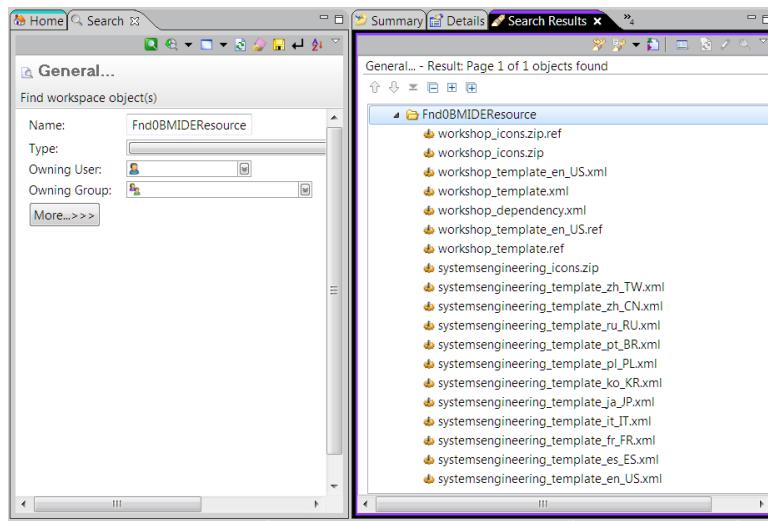
- Database backup
 1. To locate backed-up project ZIP files in the database, in the rich client use the **BMIDEProjectBackupRecovery** saved query.



2. In the search results, right-click the project archive file, choose **Named References**, select the file, and click **Download**.

- Resource dataset files

Additional Business Modeler IDE resource dataset files are stored in the **Fnd0BMIDEResource** folder. You can also use these files to restore your Business Modeler IDE environment if needed.



- Use the **manage_model_files** utility to retrieve projects backed up in the database.

1.11.4 Import a backed-up project

You should regularly back up your project in the database to ensure against data loss. To back up your project, right-click the project and choose **Properties**→**Teamcenter**→**Project Backup**.

To restore the project later, you can import it.

1. Choose **File**→**Import**.

The Import wizard runs.

2. In the **Select** dialog box, choose **Business Modeler IDE**→**Import a project backed up in Teamcenter database**. Click **Next**.
3. In the **Teamcenter Login** dialog box, log on to the database server and click **Next**.
4. Click **Next**.
5. Perform the following steps in the **Import Business Modeler IDE Template Project** dialog box:
 - a. Click the arrow in the **Project** box to choose the backed-up project to import. Click **Next**.
 - b. Click the **Browse** button to the right of the **Project contents** box to choose the workspace location to import to.
 - c. Click the arrow in the **Select active file** box to choose the extension file to make active after the project is imported.

During normal extension work, you set the active file to hold the extensions.

- d. Click the **Browse** button to the right of the **Dependent 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.
- e. Click **Finish**.

The wizard imports the project and displays it in views.

1.11.5 Backup and recovery of Business Modeler IDE data

Maintain a regular backup of all relevant data so that you can restore the data to the state it was in prior to failure. Use the **Project Backup** dialog box to set project backup. To access this dialog box, right-click the project and choose **Properties**→**Teamcenter**→**Project Backup**. By default, projects are backed up at Business Modeler IDE shutdown and project close. Then retrieve the backed-up project files and restore your project data.

For example, a deployment fails when using Teamcenter Environment Manager (TEM). To correct the problem, you must restore the database, volumes, and *TC_DATA\model* directory. You must also rename your *TC_ROOT\install\template* folder (for example, **mytemplate_old**). However, after correcting the package file and attempting to install it again using the TEM update, the deployment fails again.

TEM cannot determine whether or not a template folder in the *TC_ROOT\install* directory has been renamed, so installation failures result unless that directory is also rolled back.

Components for Business Modeler IDE project backup and recovery:

- Teamcenter Environment Manager (TEM) and the *TC_ROOT\install* directory
- Business Modeler IDE and the *TC_DATA\model* directory
- Business Modeler IDE project backup locations
 - Local backup – backed-up projects are saved on your hard disk.
 - Database backup – backed-up project ZIP files in the database.
 - Resource dataset files – dataset files stored in a database folder.

1.12 Set your own user variables for the classroom activities

Teamcenter Application and Data Model Administration

Prior to working with activities, you *must* enter your own user information on the title page of the electronic activities. Your user variables then appear in the activities.

Set them as supplied by your instructor.

Teamcenter Application and Data Model Administration

Student Activities
August 2013
MT25460 - Teamcenter 10.1

| | |
|---------------------------|---|
| User ID: | jgordon |
| Password: | jgordon |
| User Name: | Gordon, Jack |
| Student Files location: | D:\users\student\student_files |
| TC_ROOT location: | D:\tc_root |
| TC_DATA location: | D:\tc_data |
| Installed Node ID: | tc_TC101svr |
| Connection Profile: | Teamcenter Training |
| Teamcenter Help location: | https://lmdcontent.industrysoftware.automation.s... |

Make sure to click **Save** to save your user variables.

1.13 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

1.14 Summary

The following topics were taught in this lesson:

- Describe a Business Modeler IDE project.
- Describe the Business Modeler IDE process.
- Navigate the Business Modeler IDE interface.
- Create a project in the Business Modeler IDE.
- Work with Business Modeler IDE template files.
- Set up and test a connection to Teamcenter.

Lesson

2 BMIDE process and data model

Purpose

The primary purpose of this lesson is to learn the Teamcenter data model extension process. data model terms, concepts, and the best techniques for extending the data model when needed.

Objectives

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

- Perform the Business Modeler IDE data model extension process.
- Describe important criteria about business objects.
- Extend the data model with an item business object.
- Perform a Business Modeler IDE deployment.

Help topics

Additional information for this lesson can be found in:

- *Business Modeler IDE Guide*
- *Managing data model*

2.1

Introduction to the data model concepts

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.

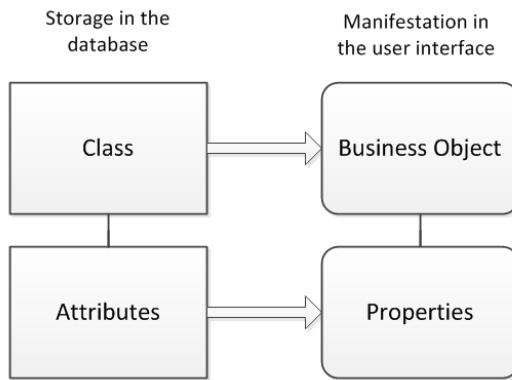
This section covers the following topics:

- 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**.

2.1.1 Teamcenter data model primer

A *data model* is a structured organization of abstract objects to represent business data. In concrete terms, data model represents the part designs, design documents, and relationships between them, as well as the business processes applied to them. Teamcenter has its own data model that you extend using the Business Modeler IDE. Before extending the Teamcenter data model, you should have a basic familiarity with its structure.



Class and attributes

A class can be thought of as a storage class in the database. It is the persistent representation of an object table in the database, and each row in that table is an instance of the class. The columns of the table are the attributes of the class, and attributes are the characteristics of the class. The attributes are also persistent, since they are stored in the database.

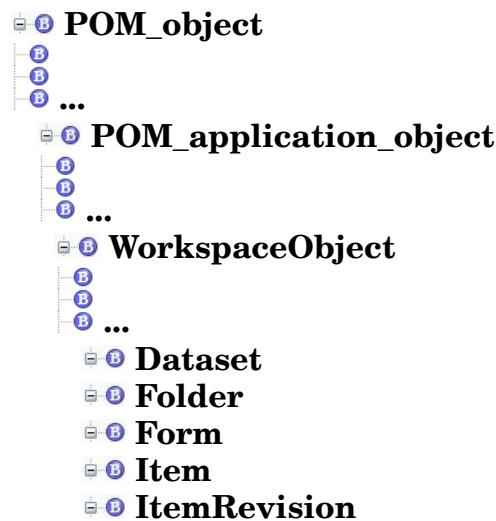
Business objects and properties

Many business objects have a storage class just for them that has the same name, and their properties are stored as attributes on that storage class. These are known as primary business objects. Primary business objects are persistent in the database and have persistent properties. Secondary business objects are business objects that store their information on their parent's storage class, and do not have a storage class of their own.

Properties come in four flavors: persistent, runtime, compound, and relation. Properties belong to business objects, and attributes belong to classes. Persistent properties are stored in their business object's storage class as attributes. In other words, attributes of a storage class are expressed as persistent properties on the business object that uses that storage class.

2.1.2 What are business objects

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



Business objects are abstract implementations of the Teamcenter classes. Each primary business object is mapped to a Teamcenter class with the same name primary business object. Additional business objects are derived from classes such as **Dataset**, **Folder**, **Form**, **Item**, **ItemRevision**, and others.

2.1.3 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 information 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.

2.1.4 Find business objects

To find objects, use one of these methods:

- Click the **Find** button  at the top of a view.
- Choose **BMIDE→Find**.
- Click a **Browse** button in a dialog box (when available).

In the **Find** dialog box, you can choose to search for the custom objects you have created or the standard COTS (commercial off-the-shelf) objects that ship with Teamcenter.

Use a wildcard * to find all instances of the object, or before and after a search string to find all instances of the string.

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.

2.1.5 POM schema importance

At the very top of the **BMIDE** folder, the parent of all business objects is displayed as **POM_object**. *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 defines an architecture (schema) for managing Teamcenter data in a database and in a running Teamcenter session. POM manages data in tables within a relational database, such as Oracle. Keep in mind 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.

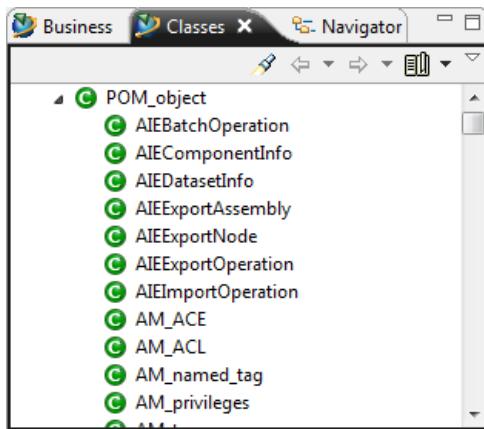
The following terms are used to describe classes, business objects, and properties.

| Term | Definition |
|-------------------------|--|
| Class | A class is the definition of an object implemented in the Teamcenter data model. A class has a set of attributes, some of which are inherited from parent classes and some of which are defined directly for the class. |
| Attribute | An attribute is a persistent piece of information that characterizes all objects in the same class. |
| Business object | A business object is the specialization of a Teamcenter class based on properties and behavior. |
| Primary business object | A primary business object corresponds to each POM class (that is, the primary business object name is the same as the POM class name). |
| Sub-business object | Sub-business objects are all business objects that belong to the same POM class. Sub-business objects inherit all the properties and behavior of the primary business object. Sub-business objects are also known as secondary business objects. |
| Property | A property is a piece of information that characterizes all objects of the same type. At a minimum, all sub-business objects have properties that correspond to the attributes for the associated class. |

2.1.6 Viewing POM schema

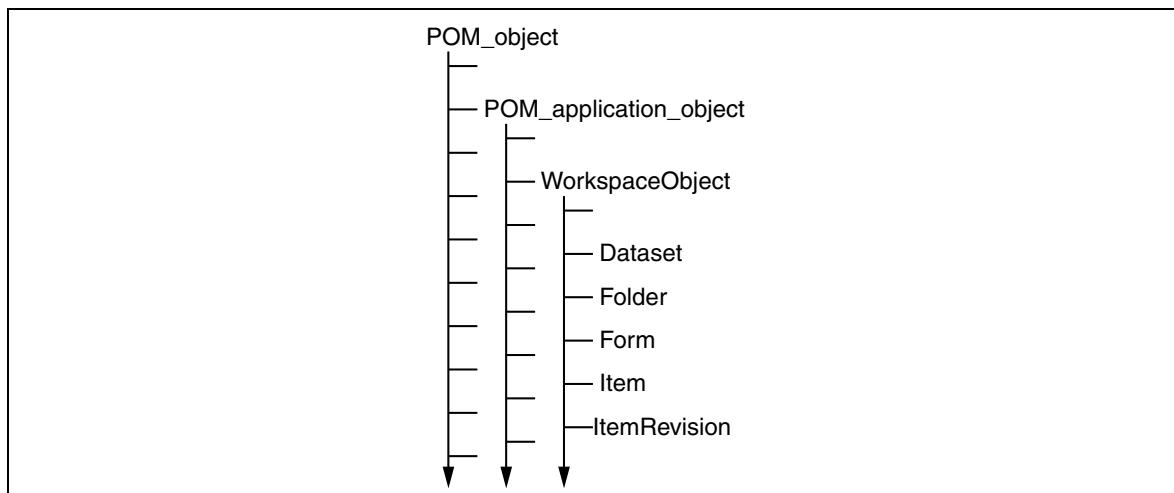
It is important to understand the persistent object manager (POM) schema when using certain applications in the rich client. Knowing how classes and their attributes are related to one another helps you create certain rules.

To view the persistent object manager (POM) schema in the Business Modeler IDE, open the **Classes** view in the **Advanced** perspective.



Classes view (the POM schema)

The following figure shows an abbreviated view of the POM schema.



Abbreviated view of POM

To see the attribute information on a class, right-click a class in the **Classes** view and choose **Open in UML Editor**.

2.1.7 Tasks for using the data model

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

- Create queries in Query Builder.
- Create property sets in Report Builder.
- Create closure rules in PLM XML/TC XML Export Import Administration.

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.

2.2 Introduction to the UML editor

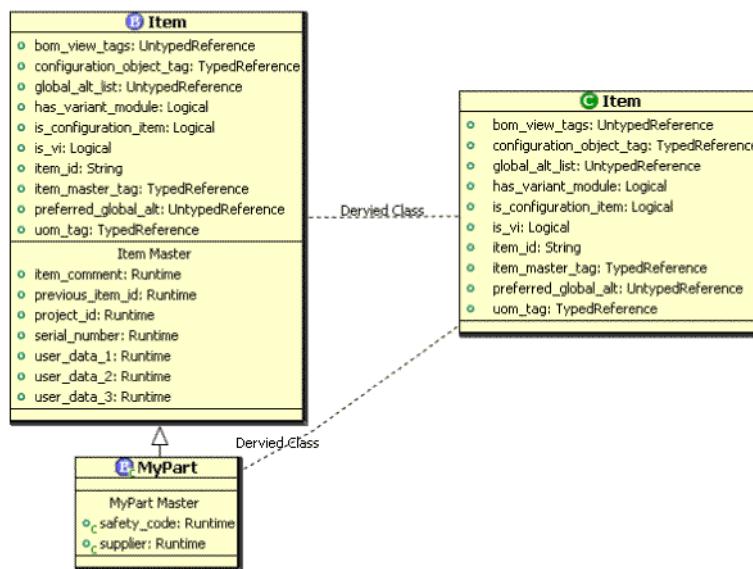
The UML (Unified Modeling Language) editor shows business objects and classes in a graphical format. To access the UML editor, right-click a business object in the **Business Objects** view or a class in the **Classes** view and choose **Open In UML Editor**. The business object or class appears in the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

UML (Unified Modeling Language) is a commonly used method to graphically represent data models. For more information about UML, see the following URL:

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

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.

Business objects Class

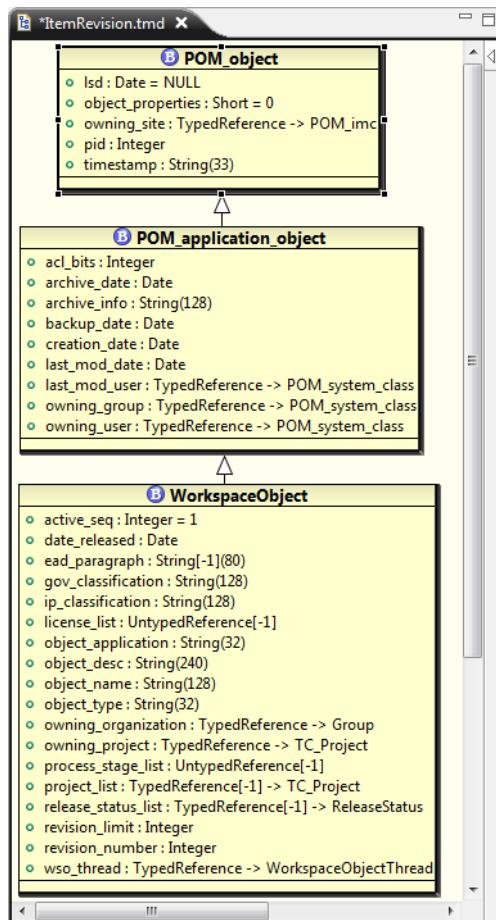


Note

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.

2.2.1 Inheritance to POM_object

The following figure shows the inheritance to root for the **WorkspaceObject** class.

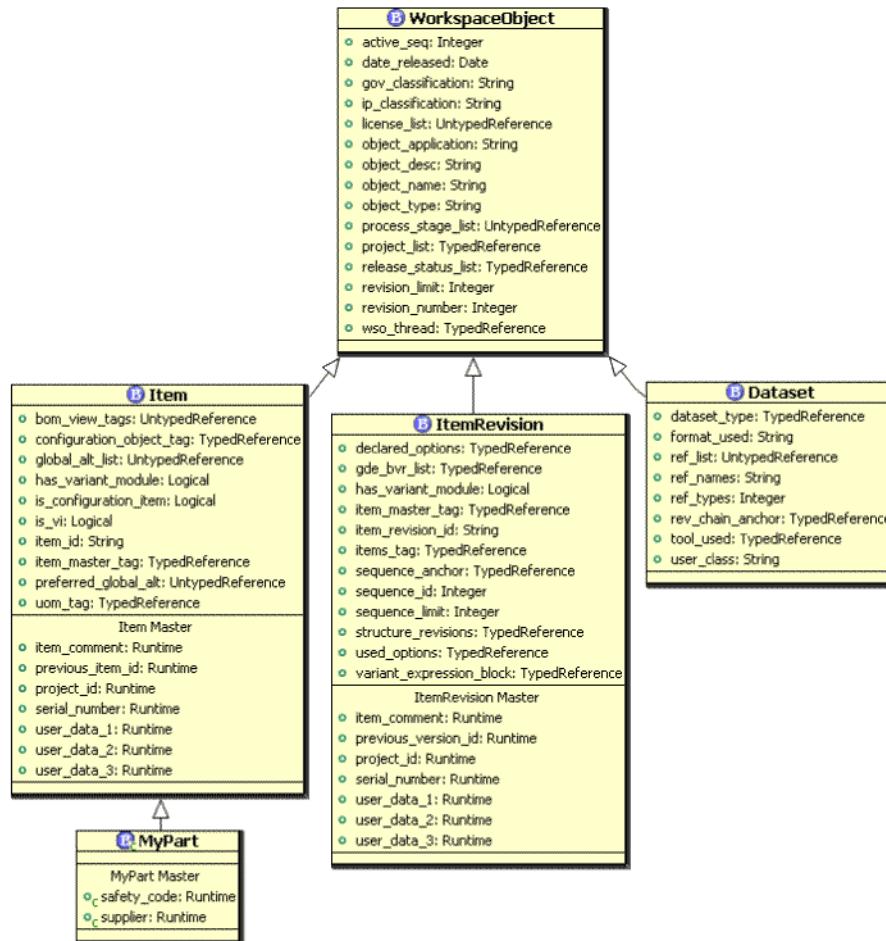


Inheritance to root for the **WorkspaceObject** class

To see the inheritance relationships the class has to other classes, right-click the top of the class box in the UML Editor (right on the class name) and choose **Show→Inheritance to Root**. All POM objects derive from the root class **POM_object**.

2.2.2 Data model inheritance

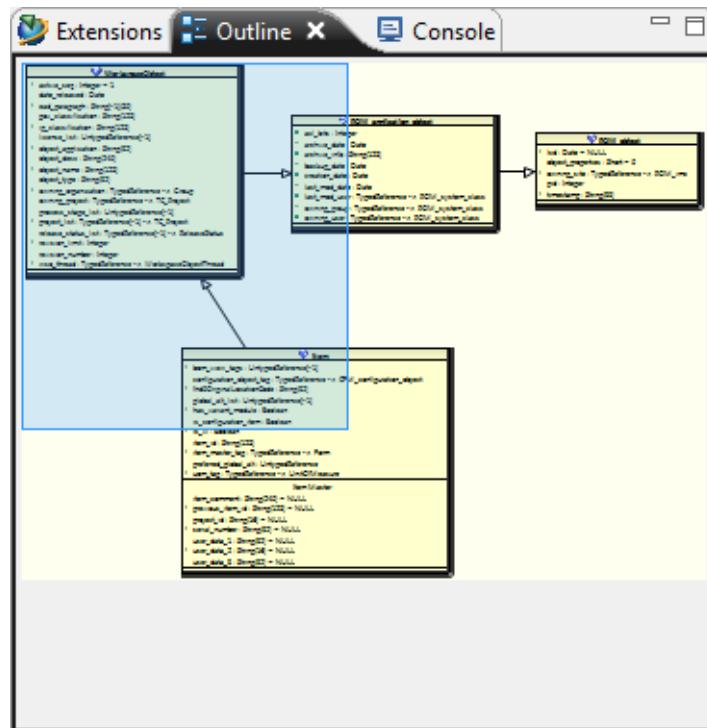
The **MyPart** business object inherits properties and behavior from its parents, **Item** and **WorkspaceObject**. **ItemRevision** and **Dataset** inherit from **WorkspaceObject**.



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

2.2.3 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

2.3

Basic process for extending the data model

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

Note

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.

Choose **BMIDE**→**Organize Extensions**→**Set active extension file**.

2. Perform the extension work.

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

- In the **Business Objects** view, right-click the parent business object and choose **New Business Object**. The New Business Object wizard runs.
- Click the **Add** button to the right of the **Properties** table to add a property to the business object. The New Property wizard runs.

3. Save your work.

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

4. Deploy your changes to the test server.

Choose **BMIDE**→**Deploy Template**, or click the **Deploy Template** button on the main toolbar.

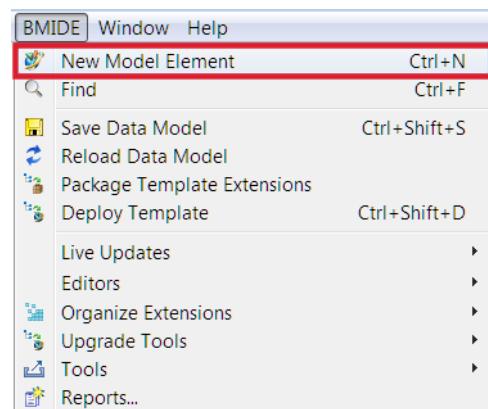
5. After deployment, test your new business object in the Teamcenter rich client by creating an instance of it.

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

2.3.1 Add a new model element

The Add New Model Element wizard allows you to create custom model elements.

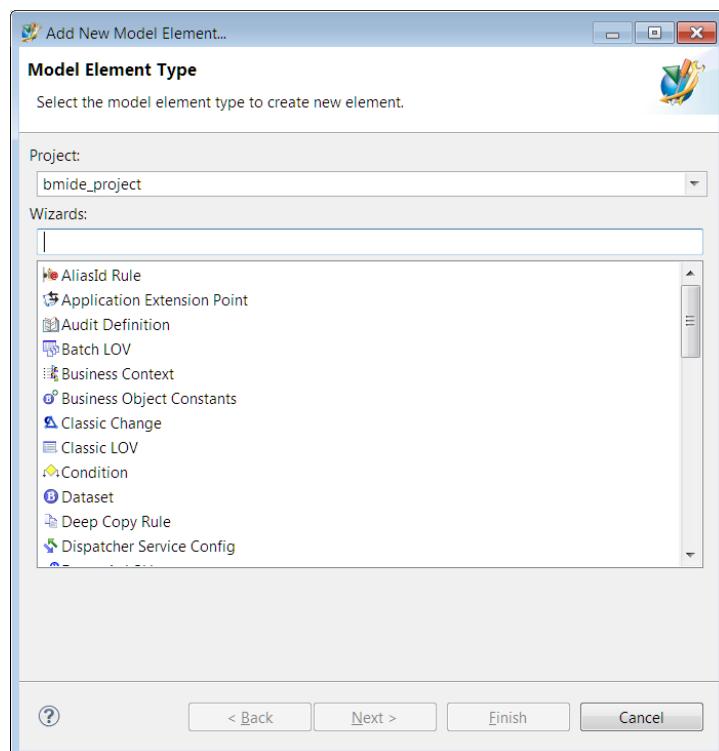
1. On the menu bar, choose **BMIDE**→**New Model Element**.



Launching the Add New Model Element wizard

2. Select the model element you want to create and click **Next**.

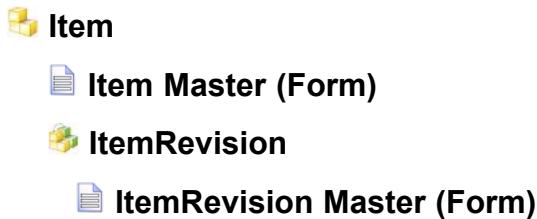
The creation wizard for the selected model element runs.



Add New Model Element wizard

2.4 Basic item data model

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



Item

Collects data that is globally applicable to all revisions of the item. You typically add custom attributes directly to the custom item business object.

ItemMaster (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.

ItemRevisionMaster (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.

Form storage classes

Item

ItemMaster

ItemMasterS (form storage class)

ItemRevision

ItemRevisionMaster

ItemRevMasterS (form storage class)

ItemMasterS (form storage class)

The storage object for the item master form.

ItemRevMasterS (form storage class)

The storage object for the item revision master form.

2.4.1 Data model extension example

Item and item revision properties should be added directly to the new business objects.

| Business objects | Classes |
|---|---|
| <p>B Item</p> <p>B MyPart</p> <p>safety_code supplier</p> <p>B ItemRevision</p> <p>B MyPart Revision</p> <p>material_code material_description weight</p> | <p>C Item</p> <p>C MyPart</p> <p>safety_code supplier</p> <p>C ItemRevision</p> <p>C MyPart Revision</p> <p>material_code material_description weight</p> |

Denotes properties

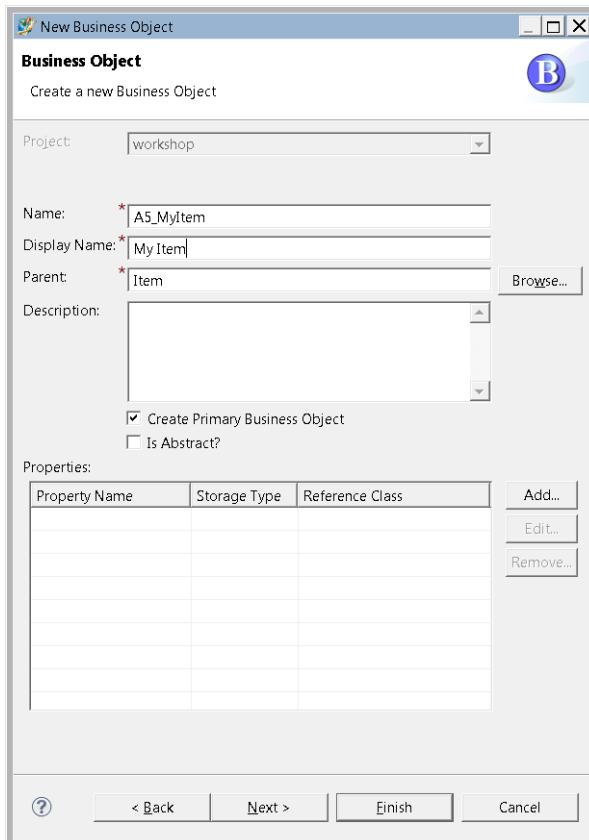
Denotes attributes

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

- **MyPart** business object is created under **Item**.
- **MyPartRevision** business object is created under **ItemRevision** automatically.
- A primary business object has the same name as its associated storage class.
- New *properties* are added to the new business object(s) and indirectly associated storage class gets new *attributes*.

2.4.2 Create an Item 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.



Creating a custom item business object

1. Choose **BMIDE**→**Organize Extensions**→**Set active extension file** to select the file where you want to save the data model changes, for example, **item_business_objects.xml**.
2. Start the **Business Object** wizard.
 - On the menu bar, choose **BMIDE**→**New Model Element**, type **Item** in the **Wizards** box, and click **Next**.
 - Click the **Find** button  at the top of the **BMIDE** view, search for the **Item** business object, right-click it, and choose **New Business Object**.

3. In the **Business Object** dialog box, enter the *item* information:

| Value | Definition |
|---------------------------------------|---|
| Name | The name you want to assign to the new business object in the database. |
| Display Name | The name as you want it to appear in the user interface. |
| Parent | The business object you already selected as the parent business object. |
| Description | The description for the new business object. |
| Create Primary Business Object | Select Create Primary Business Object to make a new class that stores the data for the new business object. |
| Is Abstract? | Only select Is Abstract? if instances of the business object will not be created in user interfaces such as the rich client. |

4. Click the **Add** button to the right of the **Properties** table to add a property to the item business object.
5. In the **Business Object** dialog box, enter the *revision* information:

| Value | Definition |
|---------------------|--|
| Display Name | In the Display Name box, type the name of the item revision as you want it to appear in the user interface. |
| Description | In the Description box, type a description for the new business object revision. |

6. 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 to enter item properties.
7. Click **Finish**.

2.5 Set display names

You can set the display name for your custom objects and properties as well as override display names on COTS (standard) Teamcenter objects.

You can use the Business Modeler IDE to define the name that displays for data model objects in Teamcenter user interface clients. For example, if you create a new part business object named **A4_Bolt**, you can use the **Display Name** box in the creation dialog box to set the display name as **Bolt**.

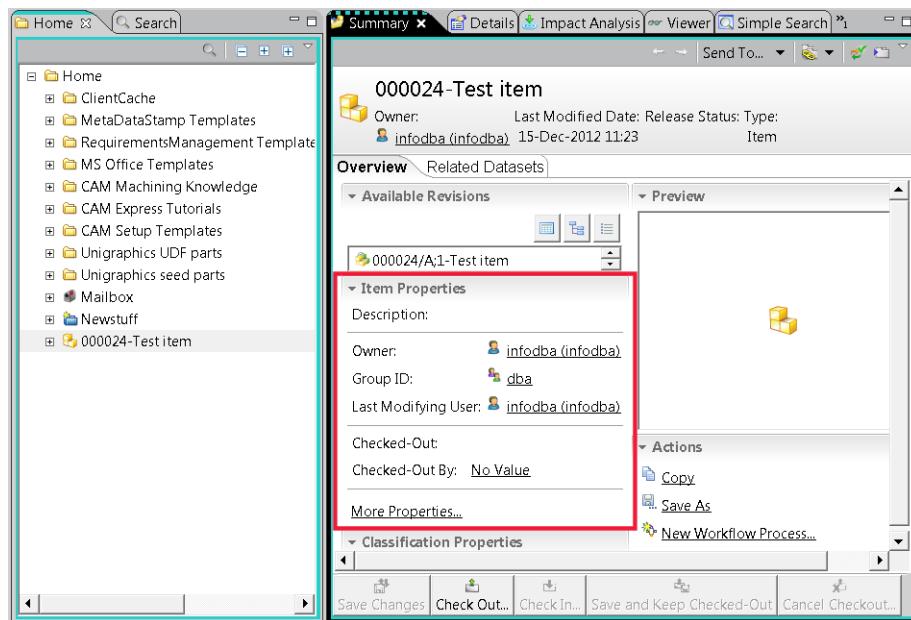
Properties use **Display Name** to define the name that displays for properties in the Teamcenter user interface.

Note

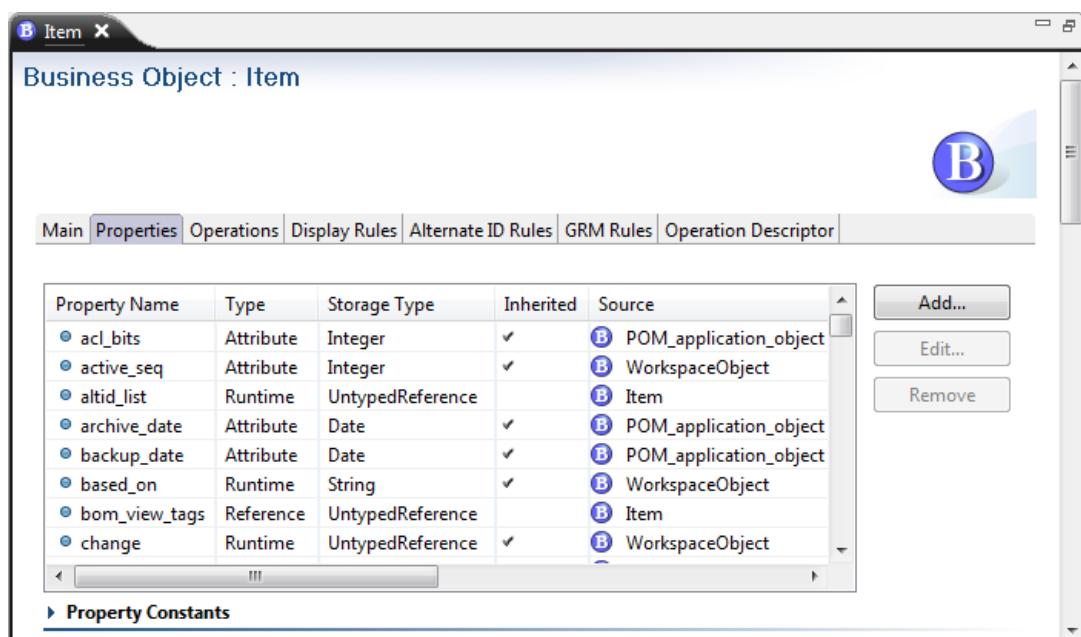
To add the display name for the object in other languages, open the new business object and click the **Add** button to the right of the **Localization** table. There will be more on localization in the next lesson.

2.6 Introduction to properties

Properties contain information such as name, number, description, and so on. A business object derives its properties from its persistent storage class. 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.



Properties in the rich client

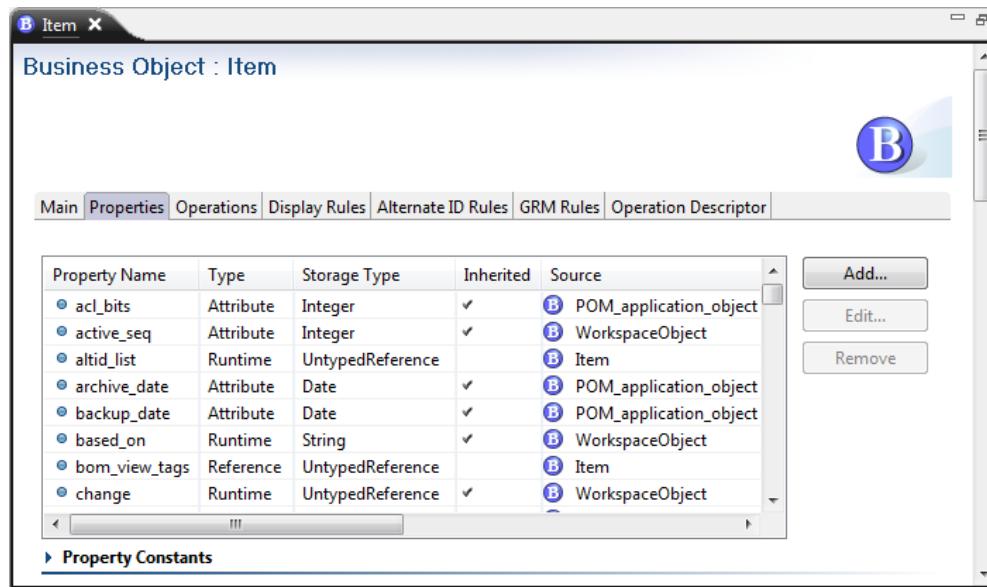


Properties in the Business Modeler IDE

2.6.1 Working with properties

Properties contain information such as name, number, description, and so on.

- *Persistent properties* – A business object derives its properties from its persistent storage class.

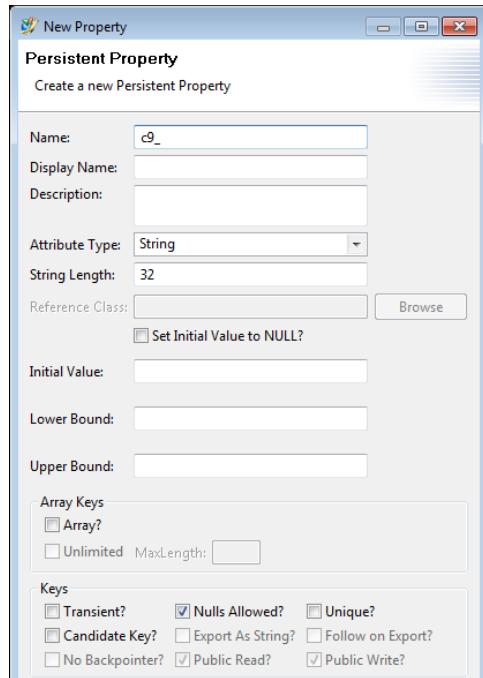


In addition to the properties that are derived from the persistent storage class, business objects can also have these additional properties.

- *Run-time properties* are derived each time the property is displayed. Their data is derived from one or more pieces of system information (for example, date or time) that are not stored in the Teamcenter database.
- *Compound properties* are properties on business objects that can be displayed as properties of an object (the display object) although they are defined and reside on a different object (the source object).
- *Relation properties* are properties that define the relationship between objects. For example, a dataset can be attached to an item revision with a specification, requirement, or reference relation, among many others.

2.6.2 Create a persistent property

This is a **Persistent Property** dialog box for a **String** persistent property.



- **Name**

The property name as you want it to appear in the database. The name cannot contain spaces. A prefix from the template is automatically affixed to the name.

- **Display Name**

The name as you want it to appear in the user interface.

- **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**

Selecting the **Nulls Allowed** check box allows an empty value for the attribute.

2.6.3 Attribute types

The user interface shows a display name for the attribute using properties. For example, the **object_desc** property displays as **Description** in the user interface. To be proficient with attributes, you know both the internal name of the attribute and its display name. You can change the settings in the rich client to display the internal name of an attribute in the user interface. Log on to the rich client as an administrator, choose **Edit→Options**, and in the left pane of the **Options** dialog box, choose **Options→General→UI**. In the right pane, click the **SysAdmin** tab and select **Real Property Name**. To verify the change, select an item in the rich client and choose **View→Properties**.

- 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.

| Type | Description |
|--------------------------|---|
| Boolean | Allows two choices to the user (True or False). |
| 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. |
| LongString | A string of unlimited length. |
| Short | An integer number without decimals from 1 to 9999. |
| String | A string of characters. |
| TypedReference | Points to a Teamcenter class. |
| UntypedReference | Points to any class of data. |

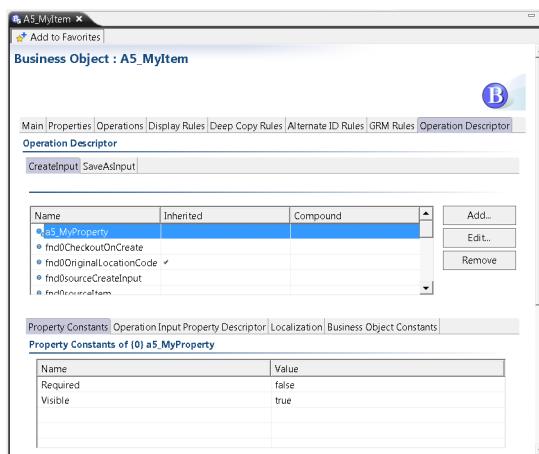
2.6.4 Attribute type settings

When defining a property, these attribute settings can be used. 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 to see these settings.

| Column | Description |
|------------------------|--|
| Size | If the attribute is a String attribute, the character length of the attribute. |
| 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. |
| Nulls Allowed | Allows an empty value for the attribute. |
| Unique | Specifies that the attribute value cannot be duplicated. |
| Candidate Key | Specifies that when exporting an object, send this attribute and rely on the receiving site to look up this string in the local database. This is typically used for system administration defined classes such as unit of measure where a local administrator may want to control what units exist on the site. |

2.7 Using the Operation Descriptor tab

You can make properties visible and required in create and save as dialog boxes in the rich client or thin client. Use the **CreateAsInput** and **SaveAsInput** operations on the **Operation Descriptor** tab for business objects. For example, when the My Teamcenter user chooses **File→New→Item** or **File→Save As** on an item, these properties are visible and required in the create and save as dialog boxes.



Property configured with the Operation Descriptor tab

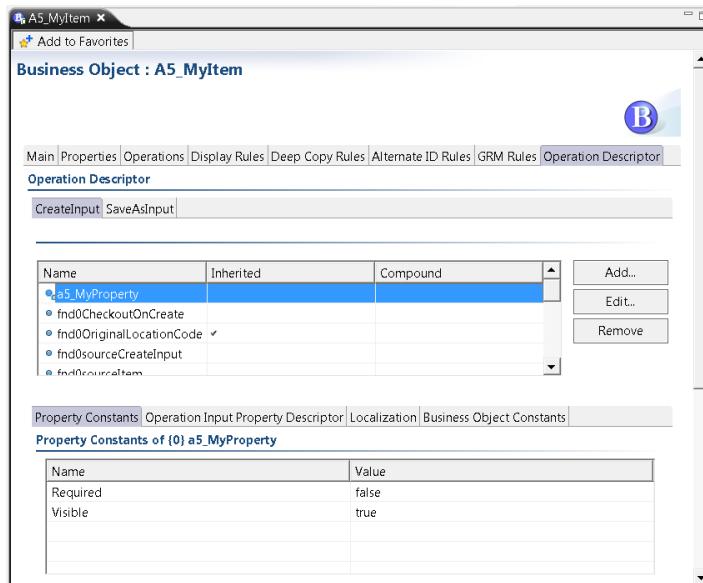
Key points

1. To access the **Operation Descriptor** tab, in the **Business Objects** view, right-click a business object, choose **Open**, and click the **Operation Descriptor** tab.
 2. In the **Operation Descriptor** tab, select a property on the table and click the **Edit** button to make that property visible or required on creation dialog boxes.
 3. Click the **Add** button to the right of the table to add a new property to the list of visible properties on this business object.
- The **Operation Input Property Descriptor** area on the tab provides details of the selected property.

2.7.1 Operation Descriptor tab for CreateInput

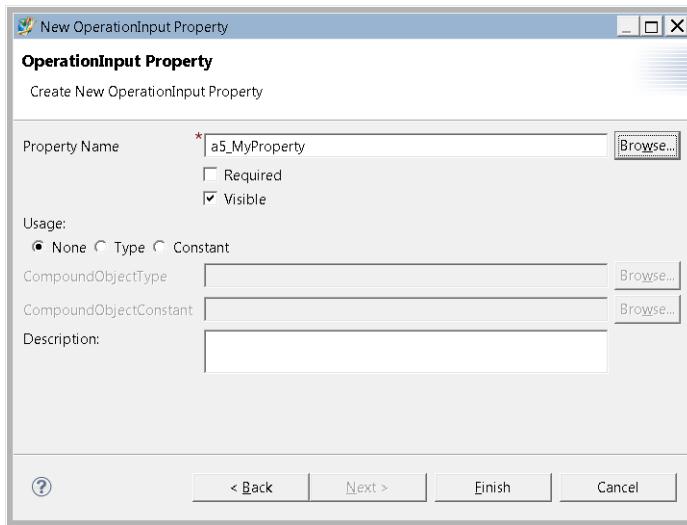
You can make properties visible and required in creation dialog boxes in the rich client or thin client by using the **CreateInput** operation on the **Operation Descriptor** tab for business objects. For example, if certain properties are selected as visible and required for the **CreateInput** operation on the **Operation Descriptor** 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. You can also use this method on the relation business object to make properties visible and required while associating a secondary business object with the primary business object using the relation business object in the rich client or thin client.

For example, when a My Teamcenter user chooses **Edit→Paste** to associate a secondary object with the primary business object using the relation business object, these properties are visible and required in the **Enter the Values for Properties on Relation** dialog box.



Property configured for the CreateInput operation

1. Click the **Operation Descriptor** tab.
2. In the **Operation** box, select **CreateInput**.
3. Select the property in the table.
 - The **Edit** button displays the **OperationInput Property** dialog box to select the **Required** or **Visible** check boxes.
 - The **Add** button behavior depends on the property type (persistent, runtime, compound or relation property).



Adding a property for the CreateInput operation

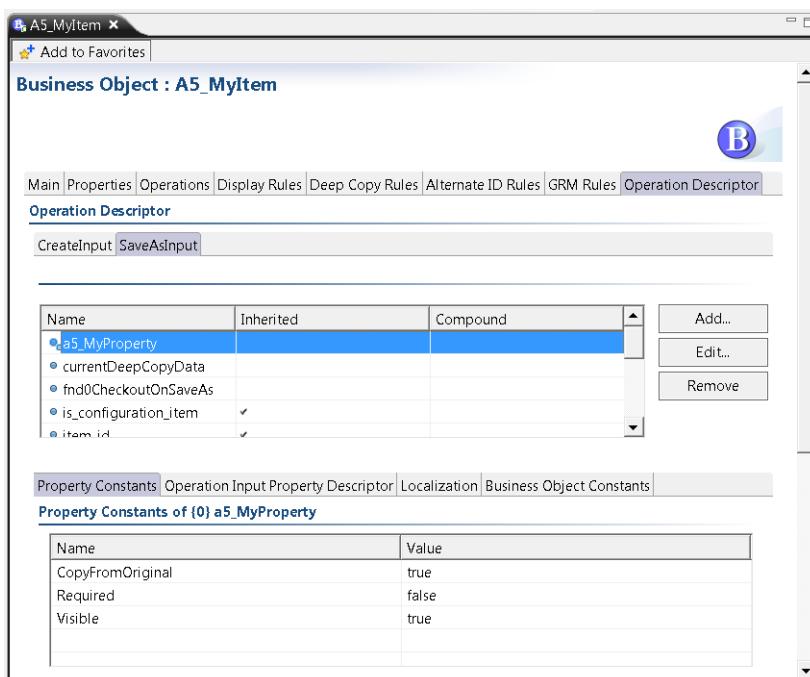
Click the **Browse** button to the right of the **Property Name** box to select an existing property.

Additional settings:

- **Required** will make the property required for entry in 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.
- **Visible** will make the property display in the creation dialog box.
- **Description** box is the description of this property.
- Under **Usage**, select one of the following:
 - o **None**
Select if the property is not a relation or reference property that references a secondary object.
 - o **Type**
Select if this is a property associated with a secondary object that is to be created from the value in the **CompoundObjectType** box.
 - o **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.

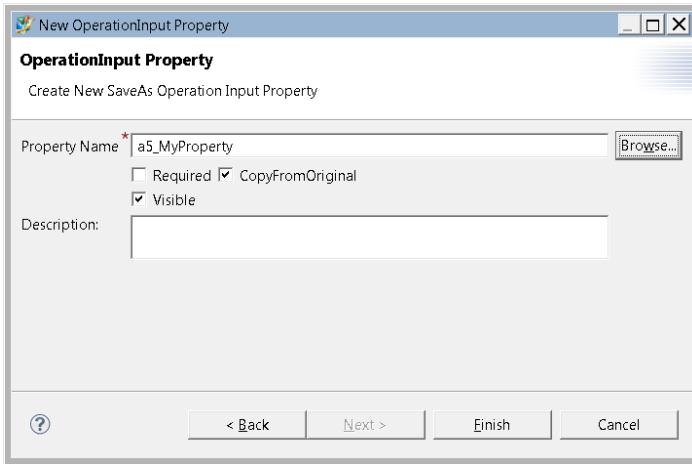
2.7.2 Operation Descriptor tab for SaveAsInput

You can make properties visible and required in save as dialog boxes in the rich client or thin client by using the **SaveAsInput** operation on the **Operation Descriptor** tab for business objects. For example, if certain properties are selected as visible and required for the **SaveAsInput** operation on the **Operation Descriptor** tab for the **Item** business object, when a My Teamcenter user selects an item object and chooses **File→Save As**, these properties are visible and required in the save as dialog boxes.



Property configured for the SaveAsInput operation on the Operation Descriptor tab

1. Click the **Operation Descriptor** tab.
2. In the **Operation** box, select **SaveAsInput**.
3. Select the property in the table.
 - The **Edit** button displays the **OperationInput Property** dialog box to select the **Required** or **Visible** check boxes.
 - The **Add** button behavior depends on the property type (persistent, runtime, compound or relation property).



Adding a property for the SaveAsInput operation

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

1. Click the **Browse** button to the right of the **Property Name** box to select an existing property.
2. Select the following behaviors for the property:
 - **Required** makes the property required for entry in the save as dialog box. The property on the save as dialog box displays a red asterisk indicating that the user is required to fill it in.
 - **CopyFromOriginal** copies the value of the property from the original object.
 - **Visible** makes the property appear in the creation dialog box.
3. In the **Description** box, type the description of this property.

2.8 Lists of values (LOVs)

An LOV is commonly accessed by Teamcenter users when they click an arrow in a data entry box.

| Safety Code: | 1 | |
|--------------|----------------------|--------------------------------|
| | Value | Description |
| 1 | Does not meet safety | Does not meet safety standards |
| 2 | | Does not meet safety standards |
| 35 | | Under review for safety |
| 40 | | Safety approval pending |

The **LOV** folder in the **Extensions** folder is used for working with lists of values (LOVs). Teamcenter is shipped with numerous LOVs for common use. This is just a small list of examples.

| | |
|---------------|----------------|
| BillCodes | Make Buy |
| Billing Types | Priority Types |
| Countries | Timezone |
| Regions | TimeCategory |

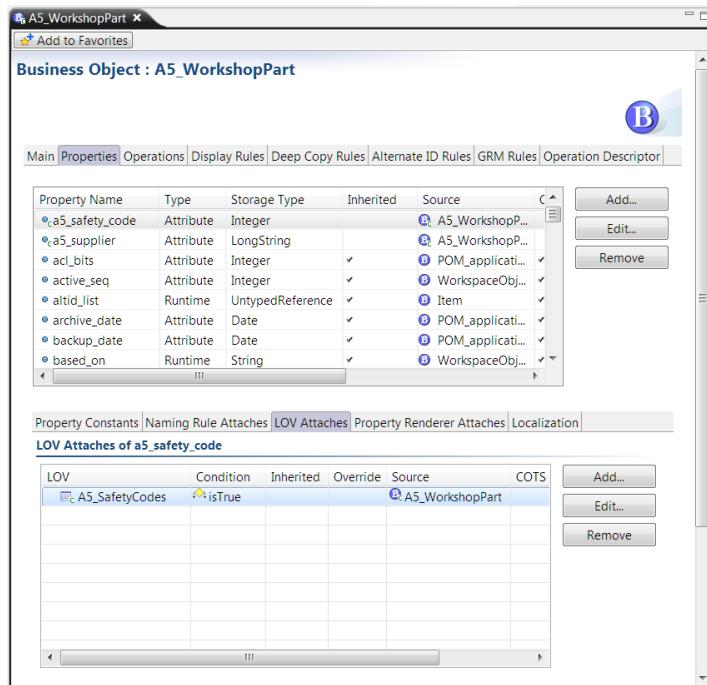
LOV attachments

These existing LOVs can be attach it to a property on a business object.

| LOV Attachments | | | | |
|------------------------------------|-----------|-----------|----------|------|
| Business Object.Property(Property) | Condition | Inherited | Override | COTS |
| C9_Item.c9_Category | !> isTrue | | | |
| | | | | |
| | | | | |

2.8.1 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.



Attaching an LOV to a property

1. Open the business object.
 2. Click the **Properties** tab.
 3. Select the property on the properties table.
 4. Click the **LOV Attaches** tab.
 5. Click the **Add** button to the right of the **LOV Attaches** table.
 6. In the **LOV** box, select the LOV.
 7. Click **Finish**.

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.

2.9 Deploying templates

After you make changes to the data model, you can deploy them to a server using the Deploy wizard. Choose **BMIDE→Deploy Template** on the menu bar, or select the project and click the **Deploy Template** button  on the main toolbar. This is also known as *live update* (formerly known as hot deploy). All your extensions are rolled up from your individual extension files into a single template and placed in the database.

Use the Deploy wizard in two different situations:

- **Deploy a template to a test server**

Live update 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.

Caution

Do not use live update to a test server when your customizations include the following:

Libraries

If you have codeful customizations, you cannot use live update because codeful customizations have libraries. Instead, you must package your template and install it using Teamcenter Environment Manager (TEM).

Localizations

Do not use live update to place localization changes on a server. Instead of using live update, install localized templates using Teamcenter Environment Manager (TEM).

- **Deploy data to a production server**

Live update to a production server when you have data to place on the server, such as LOVs and rules. You can live update nonschema data that must be updated on a regular basis. In this situation, create a template that only contains data that can be updated live, and use a source control management (SCM) system to manage the versioning of the template source file.

2.9.1 Basic process for deploying a template

Deploy a template when you want to send it to a test server prior to installing it to a production server, or when you want to send nonschema data such as LOVs to a production server.

If you are deploying live updates to production servers, you can also use the **Deployment Page** by choosing **BMIDE**→**Deployment Page** or right-clicking a live update project and choosing **Open Deployment Page**.

1. Log on to the Teamcenter test server to ensure the Teamcenter server is running. Also, check that the **BMIDE_ALLOW_DEPLOYMENT_FROM_CLIENT** preference on the server is set to **TRUE**.
2. To save any uncommitted changes to the data model, choose **BMIDE**→**Save Data Model**.
3. On the menu bar, choose **BMIDE**→**Deploy Template**.
4. Fill in the **Teamcenter Login** dialog box and click **Connect**. The Business Modeler IDE client connects to the server.
5. Optionally, select the **Generate Client Cache?** and/or **Generate Server Cache?** check boxes to regenerate cache as part of deployment.

Caution

Be aware that deployment takes longer if you select these options because it takes extra time to update the cache.

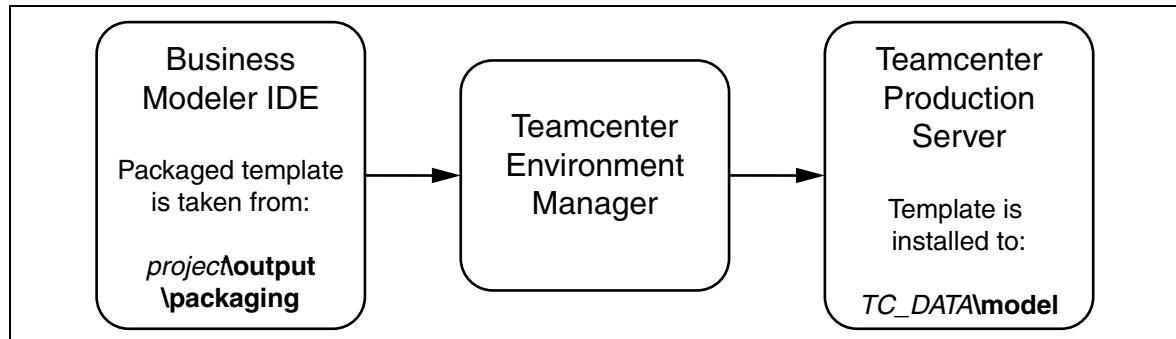
6. Click **Finish**.

The **Deployment to Teamcenter Server** dialog box displays the progress of the deployment.

7. When deployment is done, the **Deployment Results** dialog box displays the outcome of deployment.
8. Verify the data model changes are in the server by launching the Teamcenter rich client.

2.9.2 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 following figure).



Install a packaged template to a production server

1. In the Business Modeler IDE choose **BMIDE→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 **Features** panel, click **Browse** to locate the template in your project workspace.

The template is added to the **Features** panel under the **Extensions** group.

6. In the **Features** panel, choose the new feature in the **Extensions** group.

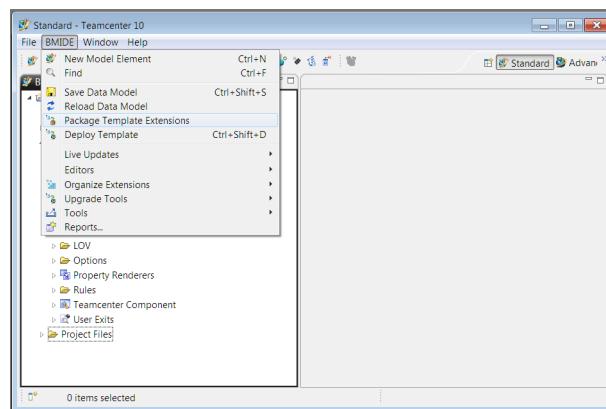
2.9.3 Package extensions into a template

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.

Note

If you have coded customizations in C++, the Business Modeler IDE packages the built C++ library as part of its template packaging. The complete solution that includes the data model and the C++ runtime libraries are packaged together.

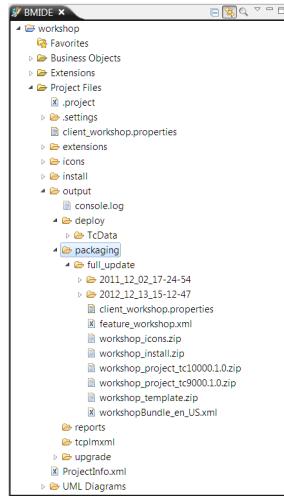
1. Choose **BMIDE→Package Template Extensions**.



2. Fill in the **Package Template Extensions** dialog box:

- **Project** box – the project whose extensions you want to package into a template.
- **Use default location** check box – leave selected if you want the template files to be placed in your workspace in the **output\packaging** folder under your project.

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**.



The **project/output/packaging** directory contains the following template files:

- ***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-nameBundle_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.

3. Install the packaged template to a Teamcenter server using Teamcenter Environment Manager (TEM).

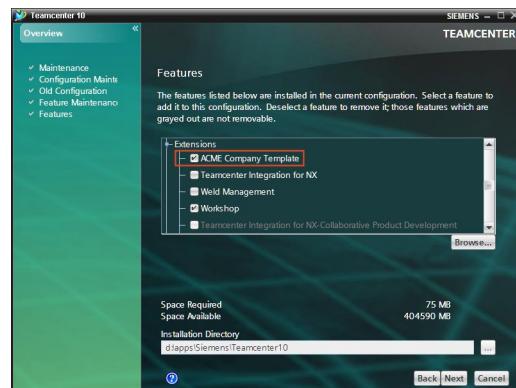
2.9.4 Install a template using TEM

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.

Warning

You should back up your data on a regular basis so that you can restore it in the event of a template installation failure.

1. Copy the template files from the **packaging** directory on your Business Modeler IDE client to a directory that is accessible by the server.
2. In the **Features** panel, click the **Browse** button locate the template files
3. Select the new template in the **Features** panel. Click **Next**.



4. In the **Confirmation** panel, click **Start**. The new template is installed.
 5. 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.

2.10 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

2.11 Summary

The following topics were taught in this lesson:

- Perform the Business Modeler IDE data model extension process.
- Describe important criteria about business objects.
- Extend the data model with an item business object.
- Perform a Business Modeler IDE deployment.

Lesson

3 *Item business object configuration*

Purpose

The purpose of this lesson is to extend Teamcenter with **Item** business objects and **ItemRevision** business objects.

Objectives

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

- Describe the recommendation for the **Item** extension.
- Create **Item** business objects.
- Configure a variety of **Item** properties.
- Add an icon to a custom business object.

Help topics

Additional information for this lesson can be found in:

- [*Business Modeler IDE Guide*](#)
- [*Creating data model objects to represent objects in Teamcenter*](#)

3.1 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.

3.1.1 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 type of data.

EndItem

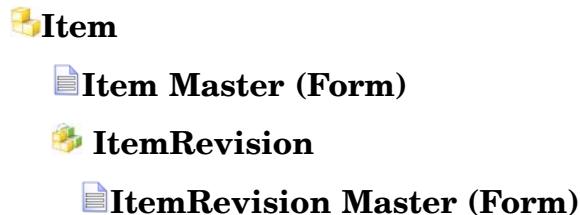
EndItem Master (Form business object)

EndItem Revision

EndItem Revision Master (Form business object)

3.1.2 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:



Object definition and purpose

- **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**

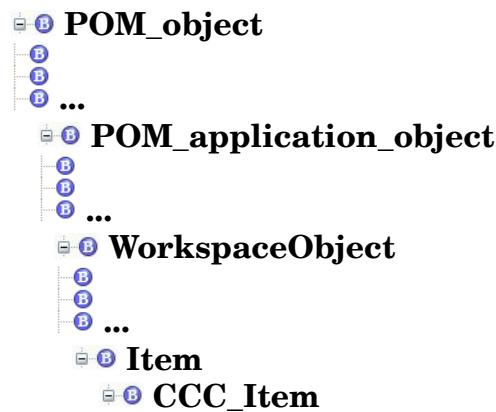
Collects data that is applicable to a single revision of the item.

- **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.

3.1.3 Effective use of inheritance

The following is an illustrated view of property and behavior *inheritance* in **CCC_Item** from the **Item** business object.



What happens?

Property inheritance

All of these properties from above and including the **Item** business object are inherited to the **CCC_Item**

Property additions

Additional properties can be added to the **CCC_Item** business object.

Behavior inheritance

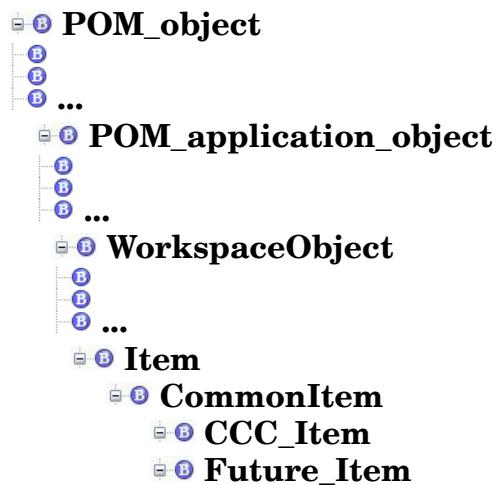
All behavior (for example, naming rules) from above and including the **Item** business object are inherited to the **CCC_Item**

Behavior optionally overridden

Different than properties, the behavior can be overridden by the **CCC_Item** (as long as the parent business object **Overridden** check box is selected).

3.1.4 Using a CommonItem business object

The following is an illustrated view of using a *uninstantiable* **CommonItem** business object in the Teamcenter business object hierarchy.



Note

In the hierarchy, **CommonItem** is an abstract class.

Advantages

Scalability for properties

For example, if you want to add **CCC_Item**, first add **CommonItem** under **Item**, and then add **CCC_Item** under **CommonItem**. Add all of your properties to **CommonItem**. That way **CCC_Item** picks up these properties.

This allows **Future_Item** to be extended at a later date and to get the inherited properties from **CommonItem**. The business logic of **Future_Item** is independent of **CCC_Item**.

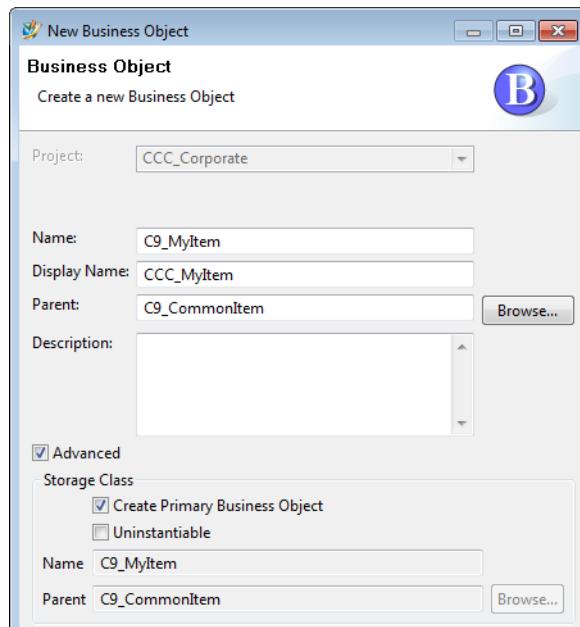
Scalability for operations

A business object operation is a function on a business object. It is defined with parameters, a return value, inheritance, and whether it is overridable (allows child business objects to override this method). After you create a business object operation, you must generate code and write an implementation for the operation.

3.1.5 Defining business objects

To extend the **Item** business object:

In this example, the **Item** business object has been extended to the **C9_CommonItem**. The following image shows the **C9_CommonItem** being extended to the **C9_MyItem** (for purposes explained earlier).



Note

When you create an **Item** business object, an **ItemRevision** business object is created automatically. You cannot create an **ItemRevision** business object directly.

To add properties to the **C9_MyItem** business object:

The following image shows the properties table that is used to add properties to the **C9_MyItem** business object.

| Properties: | | |
|---------------|--------------|-----------------|
| Property Name | Storage Type | Reference Class |
| | | |

3.1.6 Business object properties

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

| Type of property | Description |
|------------------|---|
| Attribute | A simple value (for example, integer, string, or date). The value is stored in the database as an attribute and mapped to the property. |
| Compound | A property displayed from one type as if it were the property of that type, though it actually resides on another type. Though run-time properties can be used to display such a property, they require custom coding to do so. Compound properties allow you to create such properties without custom coding. |
| Reference | A reference to another object. The reference is stored in the database as a reference attribute and mapped to the property. |
| Relation | A reference to secondary objects of an ImanRelation business object. The reference is stored in the database as a relation type and is derived from that ImanRelation business object. |
| Runtime | A property that is defined at run time and attached to types. Run-time properties do not map directly to persistent attributes, references, or relations. Instead, run-time properties derive data from one or more pieces of system information (for example, date and time) that are not stored in the Teamcenter database. Run-time properties can also be used to display a property of one type as if it were a property of another type. |

3.1.7 Add a persistent property

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

1. From the **Business Object** view, click the **Find Business Object** button to locate the business object to which you want to add the property.
2. 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.
3. Click the **Add** button to the right of the properties table. The Business Modeler IDE runs the New Property wizard.
4. Under **Property Types**, select **Persistent**.
5. Enter values in the **Persistent Property** dialog box to define the property.
6. 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.

3.1.8 Property constants

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 screenshot shows the 'Property Constants' section in the Teamcenter interface. It consists of two tables:

- Property Name Table:**

| Property Name | Type | Storage Type | Inherited | Source |
|---------------|-----------|------------------|-----------|-------------------|
| acl_bits | Attribute | Integer | ✓ | POM_applicatio... |
| active_seq | Attribute | Integer | ✓ | WorkspaceObject |
| altid_list | Runtime | UntypedReference | | Item |
| archive_date | Attribute | Date | ✓ | POM_applicatio... |
| backup_date | Attribute | Date | ✓ | POM_applicatio... |
| based_on | Runtime | String | ✓ | WorkspaceObject |
| bom_view_tags | Reference | UntypedReference | | Item |
| change | Runtime | UntypedReference | ✓ | WorkspaceObject |
- Property Constants Table:**

| Name | Value | Overridden | COTS | Template |
|--------------------------------|------------|------------|------|------------|
| ComplexProperty | | | ✓ | foundation |
| Enabled | false | | ✓ | foundation |
| Exportable | Prohibited | | ✓ | foundation |
| Fnd0SecurityPropagationEnabled | false | | ✓ | foundation |
| InitialValue | | | ✓ | foundation |
| Localizable | false | | ✓ | foundation |
| Modifiable | Read | | ✓ | foundation |
| Required | false | | ✓ | foundation |

The following constants can be applied to business object properties:

| Property constants | Description |
|-------------------------|--|
| 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. |

3.1.9 Property constant behavior

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 create property constants for a number of situations. Some examples are:

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

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.

3.1.10 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 screenshot shows the 'Property Constants' section of the configuration interface. At the top, there is a table listing various properties with their types, storage types, inheritance status, and source. Below this is a section titled 'Property Constants' containing another table with columns for Name, Value, Overridden, COTS, and Template. The 'Name' column lists several constants: ComplexProperty, Enabled, Exportable, Fnd0SecurityPropagationEnabled, initialValue, Localizable, Modifiable, and Required. The 'Value' column contains values like false, Prohibited, and Read. The 'Template' column indicates they all belong to the 'foundation' template.

| Property Name | Type | Storage Type | Inherited | Source |
|---------------|-----------|------------------|-----------|-------------------|
| acl_bits | Attribute | Integer | ✓ | POM_applicatio... |
| active_seq | Attribute | Integer | ✓ | WorkspaceObject |
| altid_list | Runtime | UntypedReference | | Item |
| archive_date | Attribute | Date | ✓ | POM_applicatio... |
| backup_date | Attribute | Date | ✓ | POM_applicatio... |
| based_on | Runtime | String | ✓ | WorkspaceObject |
| bom_view_tags | Reference | UntypedReference | | Item |
| change | Runtime | UntypedReference | ✓ | WorkspaceObject |

| Name | Value | Overridden | COTS | Template |
|--------------------------------|------------|------------|------|------------|
| ComplexProperty | | | ✓ | foundation |
| Enabled | false | | ✓ | foundation |
| Exportable | Prohibited | | ✓ | foundation |
| Fnd0SecurityPropagationEnabled | false | | ✓ | foundation |
| initialValue | | | ✓ | foundation |
| Localizable | false | | ✓ | foundation |
| Modifiable | Read | | ✓ | foundation |
| Required | false | | ✓ | foundation |

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.

3.1.11 Working with property constants

Enabled property constant

The **Enabled** property constant allows you to specify whether a property is enabled in the user interface. Valid values for the **Enabled** constant are **true** and **false**.

By default, the **Enabled** property constant value is displayed as **false**, but it actually obtains its value from the **Modifiable** property constant. The **Enabled** property constant is actually set to true or false only if you manually set it. To set it to **true**, select the property in the table on the **Properties** tab and select the **Enabled** property constant, and then click the **Edit** button and select the **Value** check box. This puts a check mark in the **Overridden** column of the constant, indicating that this overrides the default setting. If you now clear this check box, the value changes back to **false**, and a check mark appears in the **Overridden** column indicating that the constant is actively set to **false**.

This constant cannot be set to **true** for read-only properties. In addition, if this constant is set to **false**, users cannot modify the values through the Teamcenter rich client and thin client interfaces; however, the property values can be modified using the ITK interface according to the **Modifiable** property constant configuration.

Modifiable property constant

The **Modifiable** property constant allows you to place restrictions on the modifiability of an object property. The following restrictions can be applied:

- **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.
- **Write Only If Null** – Allows users to modify the property only if the existing value is null, or empty.

Required property constant

The **Required** property constant allows you to specify whether a value must be entered for a specific object property. Valid values for the required constant are **true** and **false**. The **Required** property constant applies only to the following object business objects: **Item**, **ItemRevision**, **Alias**, **Identifier**, **Dataset**, and **Form**.

You can also use the **Operation Descriptor** tab on business objects to make a property required in creation dialog boxes.

3.1.12 Change 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 a **Boolean** value, select or clear the **Value** box.
 - For a **List** value, select a value from the list in the **Value** box.
 - For a **String** value, type a value in the **Value** box.
4. Choose **Finish**.

Note

Once a property constant is changed, the **Overridden** property is selected and the **Template** property is set to the current template.

3.2 Introduction to multifield keys

Multifield keys are identifiers assigned to each object to ensure their uniqueness in the database. Administrators use the **MultiFieldKey** business object constant to assign the key definitions to different business object types. Administrators can add multiple properties to define a key.

The multifield key is composed of a *domain* name and a combination of the object's properties:

domain{properties}

Default multifield key definition

The default multifield key definition for **Item** business objects is **Item{item_id}**. Because children business object types inherit the key definition from their parent, they belong to the same domain as the parent business object.

Configured multifield keys example

Suppose you have a document and a drawing that both describe an item, and you want to give the item, drawing, and document objects the same item ID number so that others can see at a glance that they all describe different aspects of the same thing. Prior to Teamcenter 10, you could not give two different items the same **item_id** value because the value of the **item_id** property was assumed to be an object's unique ID. But now using multifield keys, you can give different item types the same **item_id** value by assigning different domains (business object types) to the key definitions.

In this example, you can use the **Item** business object's default multifield key definition (**Item{item_id}**), but change the multifield key definition for **Drawing** business objects to **Drawing{item_id}** and for **Document** business objects to **Document{item_id}**.

When the keys are installed to the Teamcenter database, end users are allowed to create instances of these different object types with the same item ID because the key definitions each have a different domain (**Item**, **Drawing**, and **Document**).

3.2.1 Multifield keys in Teamcenter applications

Teamcenter applications support the use of multifield keys. However, there are some considerations that you must keep in mind:

- Utilities

Arguments containing **-*key***, such as **-key**, **-itemkey**, and **-keyFileName**, are used in some utilities.

- External CAD applications

Teamcenter integrates with some external CAD applications that do not support multifield keys. These CAD applications attempt to find items in Teamcenter using the **item_id** property only.

To use one or more of these nonmultifield key compliant CAD applications with custom multifield key definitions, you must have a domain in Teamcenter with a multifield key that contains only the **item_id** property. The domain must include all the business object classes used by the nonmultifield key compliant CAD applications. The domain for the CAD application business objects is specified in the **TC_MFK_DEFAULT_DOMAIN** preference.

- Classification

You cannot use the graphics builder feature in Classification if multifield key support is enabled.

- Visualization

Most visualization features do not require any special configuration to work with multifield key data, but there are a few exceptions.

Additional information can be found in the *Business Modeler IDE Guide*.

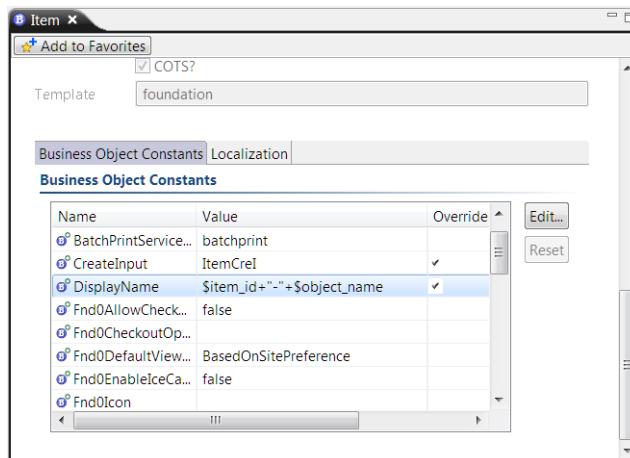
3.3 Configure the Object name

If you want to select the properties to display item instance names in the user interface, use the **DisplayName** business object constant. This constant provides the value used for the **object_string** property, which displays names in the **Object** box in property dialog boxes and in other places in the user interface.

Note

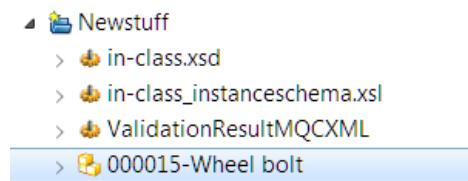
The **DisplayName** business object constant should not be confused with naming rules or the localized display name of business objects.

1. For example, open the **Item** business object and select the **DisplayName** business object constant.



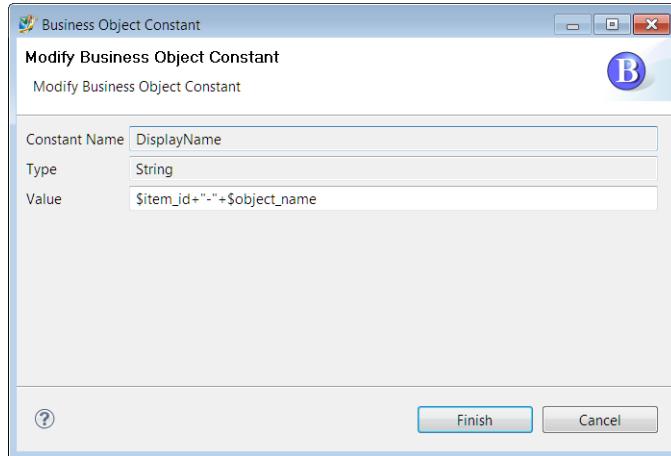
Default DisplayName for Item business object constant

By default, the constant is set to **\$item_id+"-"+\$object_name** for **Item** business objects. This displays the item name in the user interface.



Default displayed name in the user interface

2. To change the constant, select the **DisplayName** constant in the **Business Object Constants** table and click the **Edit** button to the right of the table. The **Modify Business Object Constant** dialog box is displayed.

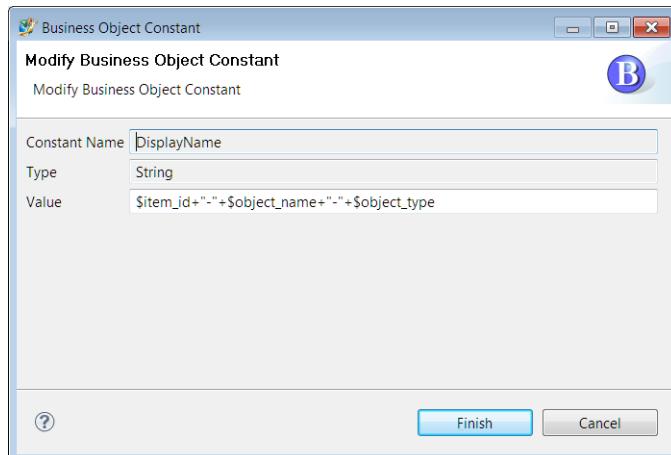


Modifying the DisplayName business object constant

3. In the **Value** box, type the properties you want to display using this format:

+\$property

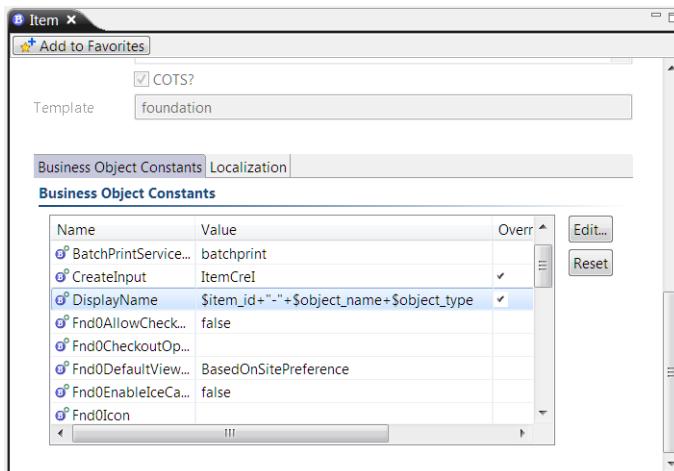
For example, if you want to add the **object_type** property (which shows the name of the business object type), add it as shown.



Property added to the DisplayName business object constant

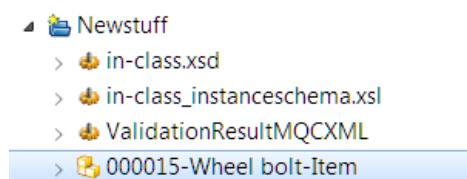
4. Click **Finish**.

The new constant definition for this business object type is shown.



Modified DisplayName for Item business object constant

5. Deploy your custom template to the database.
6. Verify the modified display name format in the user interface.



Modified displayed name in the user interface

3.4 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.

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.

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 to open it in an editor view.

3.5 Import localization files

You can import the localizations (display names) from a template file into a Business Modeler IDE project. The display names in the template file are written to localization file. This is useful when you want to import display names into your project. You can also import a localization file from one project template that you want to share with another project template.

1. Choose **File→Import**.

The Import wizard runs.

2. In the **Select** dialog box, choose **Business Modeler IDE→Import Localizations**.
3. Click **Next**.
4. Perform the following steps in the **Import Localizations** dialog box:
 - a. Click the arrow in the **Project** box to select the project to receive the localization files.
 - b. Click the **Add** button to the right of the **Files to Import** table to browse to the location of the localization files.
 - c. Click **Finish**.

To extract user interface text in bulk to translate into different languages, you can export the text into localization files using the **l10n_import_export** utility or the **Tools→Localization→Export Translations** menu in the rich client.

After you enter translated text into these files, you can import them to your template using the **Import Localizations** menu in the Business Modeler IDE:

3.5.1 Add the Localization button to properties

The **Localization** button allows localization administrators to enter localized text for the property values. To place the **Localization** button on properties in the rich client or the thin client, set the **Localizable** property constant to **true** for those properties.

Note

A system administrator must use the Organization application to add localization administrators to a translator group to give them authorization to enter the localized text.

1. In the Business Modeler IDE, open a business object and click the **Properties** tab.
2. Select a string property in the table, for example, **object_name**.
3. In the **Property Constants** table, select the **Localizable** property constant.
4. Click the **Edit** button to the right of the **Property Constants** table.
The **Modify Property Constant** dialog box appears.
5. Click the arrow in the **Value** box to change the value to **true**.
6. Click **Finish**.
7. Package your template and install it using Teamcenter Environment Manager (TEM).
8. Verify the **Localization** button appears on the property in the rich client.

For example, if you want to add the **Localization** button to the **object_name** property on the **Item** business object, open an **Item** business object and click the **More Properties** link in the **Summary** tab. The **Localization** button appears to the right of the **Name** box.

Note

To remove the **Localization** button from this property, in the Business Modeler IDE, select the **Localizable** property constant for the property and click the **Reset** button. Then deploy the change.

3.5.2 Set display names for properties

Properties display information about objects in Teamcenter, such as name, creation date, owner, and so on. You can define the name that displays for properties in the Teamcenter user interface.

1. Open a business object and click the **Properties** tab.
2. Select a string property in the table for which you want to change the display name, for example, **object_name**.
3. Click the **Override**, **Edit**, or **Add** buttons to the right of the **Localization** table to change the text.

3.6 Add or change a business object icon

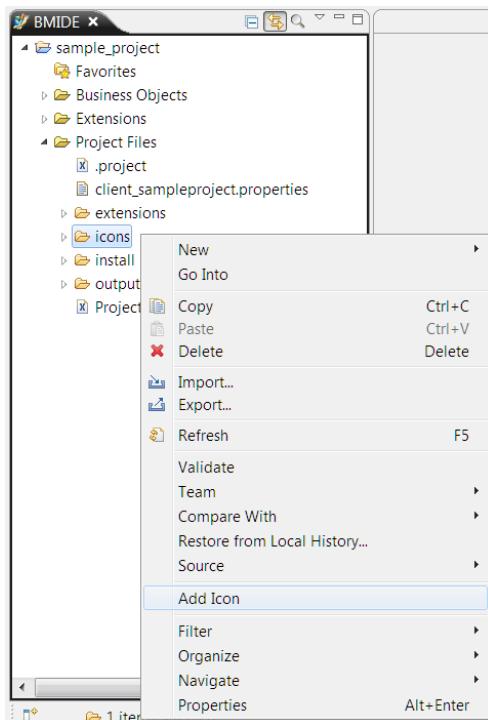
You can change the default icon for an existing business object or add an icon to a newly created business object. You can also decorate the icon with images to designate the business object's state (for example, status, remote, checked out, process, and so on).

Consider the examples:

- An icon of a custom pencil business object: 
- A pencil business object that is checked out: 

The icon has an X placed above the pencil.

You use the **Fnd0Icon** business object constant to define the icons to place on business objects. These icon definitions are placed on the server and used by the rich client.



Business object icon folder

The following procedure describes how to add an icon to a pencil business object and how to overlay images onto the icon.

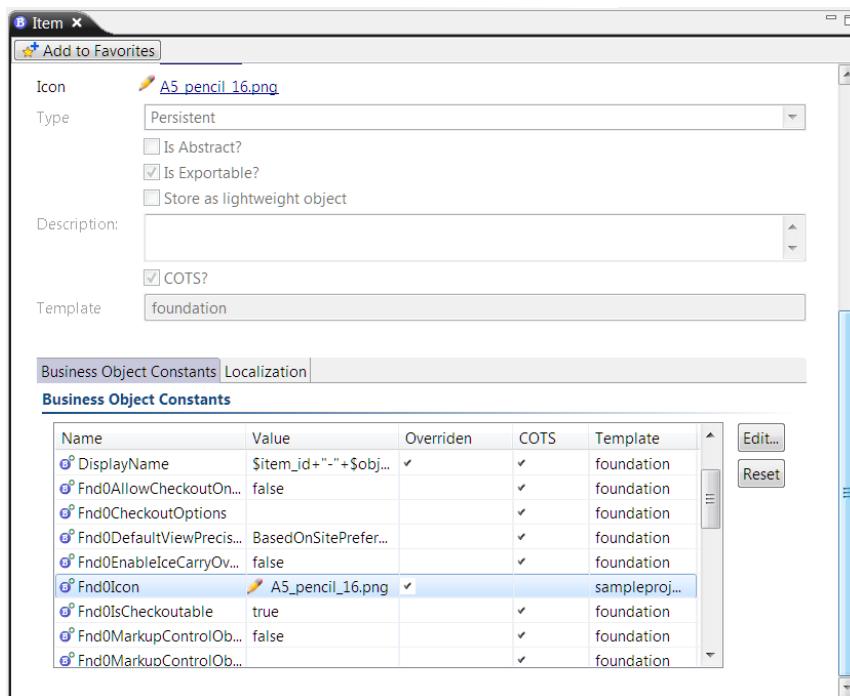
1. Create the icons.

Use a graphics editing tool to create a primary icon (16x16 pixels in size with a transparent background). For the business object's state, create icons with transparent backgrounds. These images are overlaid onto the business object icons.

2. Add the icons to your Business Modeler IDE project.

In the **Navigator** view, right-click your project and choose **Add Icon**. Your project's naming prefix is added to the icon file name, and the files are placed in the **icons** folder under the project.

3. Specify the primary icon to represent a business object on the **Main** tab in the **Business Object Constants** table, select the **Fnd0Icon** business object constant.



4. Deploy the icon to a server.

5. Verify the icon appears in the rich client.

3.6.1 Set up a business object icon overlay

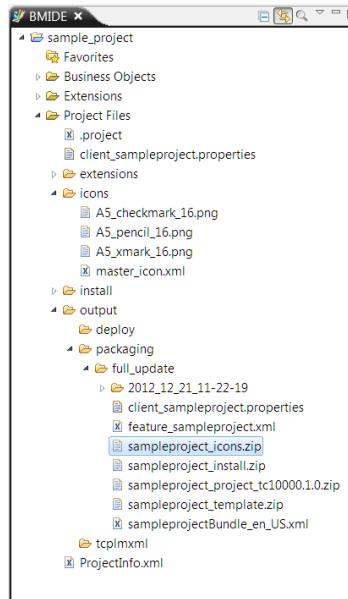
1. Create a property renderer to associate (overlay) icons based on property values.
 - a. Define the images to appear depending on the properties on the business object. For example, you can overlay an image on the primary icon if the **checked_out** property resolves to the **Y** value.
 - b. Create the property renderer. In the **Extensions** view, right-click the **Property Renderers** folder and choose **New Property Renderer**.
 - c. In the **Render Definition** box, type a well-formed XML definition for the rendering. For example, to place overlay icons on the primary icon that designate when a business object is checked out, use a definition similar to the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<icons Version="1.0">
  <primaryIcon source="A5_pencil_16.png"/>
  <overlayIcon source="checked_out" mapName="CheckedOutMap" />
  <propertyMap name="CheckedOutMap">
    <item key="Y" value="A5_xmark_16.png"/>
    <item key="N" value="A5_checkmark_16.png"/>
    <item key=" " value="A5_checkmark_16.png" />
  </propertyMap>
</icons>
```

- d. Attach the property renderer to the business object with the **Property Renderer Attachments** box.

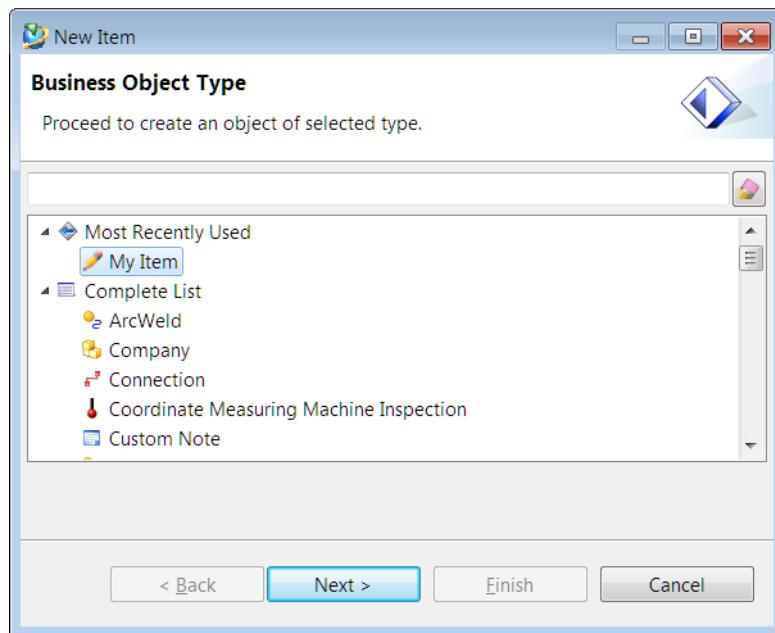
2. Deploy the icons to a server.

The icons are placed in a *project_icons.zip* file.



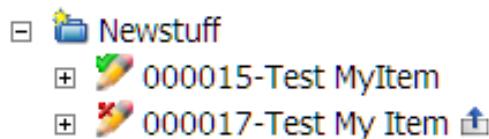
3.6.2 View the icons in the rich client

Create an instance of the business object in the rich client. For example, in the My Teamcenter application in the rich client, choose **File→New→Item** and choose the custom business object in the example. Note the new icon on the business object.



If any overlays are placed on the icon, change the state of the business object instance to view them.

For example, when the custom business object is checked in, a check mark appears on the icon. And when our custom business object is checked out, an X appears on the icon.



3.7 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

3.8 Summary

The following topics were taught in this lesson:

- Describe the recommendation for **Item** extension.
- Create **Item** business objects.
- Configure a variety of **Item** properties.
- Add an icon to a custom business object.

Lesson

4 Form business object configuration

Purpose

The purpose of this lesson is to extend the Business Modeler IDE with **Form** business objects and configure a variety of **Form** properties.

Objectives

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

- Create **Form** business objects.
- Configure a variety of **Form** properties.

Help topics

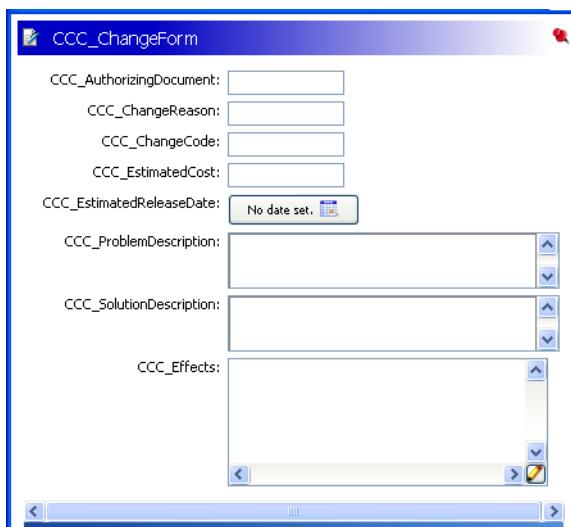
Additional information for this lesson can be found in:

- *Create a form business object*

4.1 Teamcenter forms

Forms are one of the various ways of storing attribute values in Teamcenter. By default, forms are displayed as a simple list of properties. The form is similar to the **Properties** dialog box. However, you can only use this technique if you are storing your information in a form definition class in the database. Otherwise, it is not possible to determine which properties to list.

Form business objects manage underlying product information and control how this information is displayed to users.



Key points

- Properties specific to each form can be displayed in the **Viewer** tab or in a separate dialog box.
- A Teamcenter form has properties that have a logical association (for example, employee data, change data), but the properties do not need multiple revisions (as Teamcenter items do).
- A Teamcenter form can store properties and be related to the Teamcenter item with relation (attachment) properties.
- An item can contain multiple forms that each serve a different purpose.
- Form display can be controlled by access rules.

4.1.1 End user form creation

End-users create forms in the Teamcenter rich client using one of the following methods:

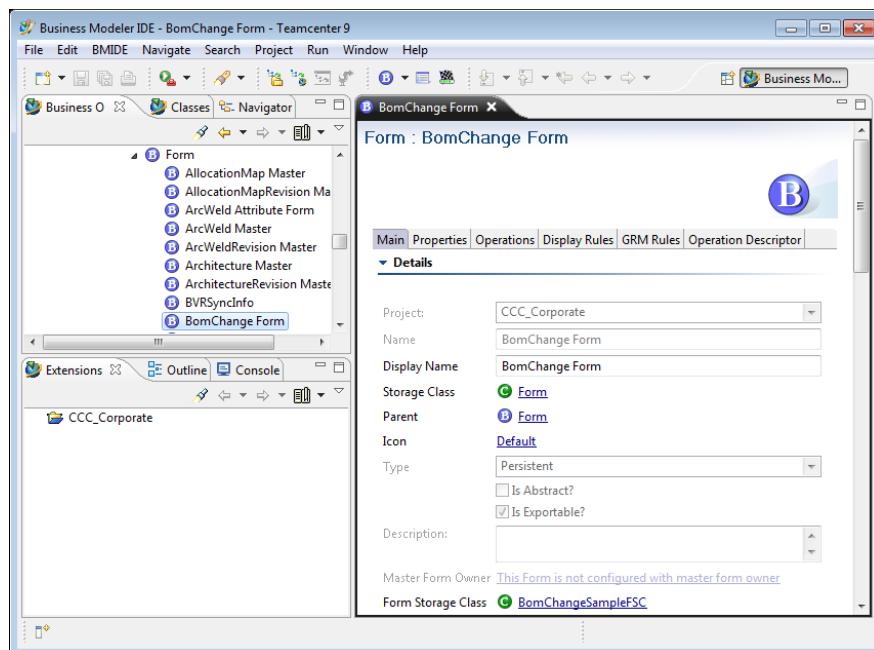
- Choose **File→New→Form**. This is used to create a stand-alone form object in a container (like a folder) or a form associated with an existing item or item revision object. End users select the type of form they are creating from the list of form types on the **New Form** dialog box.
- Choose **File→New→Item**. When an item is created, at least two form objects are also created: the item master and item revision master.
- Choose **File→New→Change**. When change objects are created, additional forms may be automatically created for the change revision object. How many forms and of what type are set when you create new change objects.
- Use a workflow handler. Form objects may be created automatically during a workflow process by using the **EPM-create-form** action handler.

To view a form in the Teamcenter rich client, select the form object and click the **Viewer** tab.

4.2 Defining forms

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.

Forms store their metadata in the **Storage Class** class (**Form**) and their customer property data in the **Form Storage Class**.



Key points

Form business object properties are stored in two locations and are automatically combined and displayed in the **Properties** table of the new form (**BOMChange Form**).

- **Storage Class**

Properties of the **BOMChange Form** business object are inherited from the **Form** business object.

- **Form Storage Class**

Properties specific for each form, for example, **erp_code** and **vendor**, are added as properties on the **BOMChange Form** business object, but the Business Modeler IDE creates them as *run time* properties. They are persistent properties on the **Form Storage Class**.

4.2.1 Forms data model

The data model for a new **Form** business object uses the secondary business object approach. The **MyERPform** uses the **Form** storage class. This differs from the **Item** data model, which uses the primary business object approach.

| Business objects | Classes |
|---|---|
| <p>⑤ Form</p> <p>⑤ MyERPform</p> <p>⑤ POM_object</p> <p>⑤ MyERPformStorage</p> <p>erp_code</p> <p>vendor</p> <p><i>Denotes properties</i></p> | <p>⑥ Form</p> <p>⑥ POM_object</p> <p>⑥ MyERPformStorage</p> <p>erp_code</p> <p>vendor</p> <p><i>Denotes attributes</i></p> |

Note

A primary business object has the same name as its associated storage class. A secondary business object uses the storage class of its parent business object. Properties for **Item** and **ItemRevision** are placed *directly* on your new **Item** and **ItemRevision** business objects.

When the **MyERPform** business object is created:

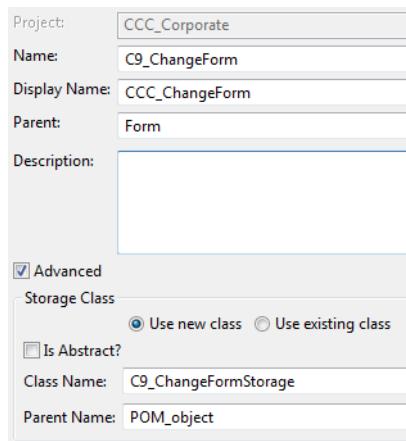
1. The **MyERPform** business object is created and displayed under **Form**. The class for the **MyERPform** business object is **Form**.
2. For customer specific properties, the ⑤ **MyERPformStorage** is created along with the ⑥ **MyERPformStorage** storage class under **POM_object**. Business Modeler IDE provides a default storage class name, but it can be overwritten while the new form is created.

Note

The **Form** class attribute *data_file* references the **MyERPformStorage** class in order to load the customer specific form properties.

4.2.2 Create a form business object

Creating a **Form** business object involves providing **Name** and **Display Name** values, confirming the **Storage Class** option, and defining the form **Properties**.

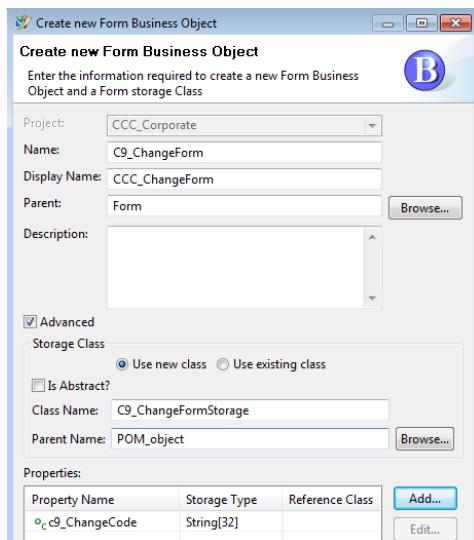


1. In the **Business Objects** view, right-click the **Form** business object (or one of its children) and choose **New Business Object**.
The Create New Form Business Object wizard runs.
2. In the **Create New Form Business Object** dialog box, enter the following information:
 - a. Type **Name**, **Display Name**, and **Description** values for the new business object.
 - b. Select the **Advanced** check box to view the **Form Storage Class** information.
 - The **Name** box automatically contains the new storage class, *which is automatically created*, when the **Form** business object is created.
 - The **Parent** box automatically contains the parent class.
 - c. Click the **Add** button to the right of the **Properties** table to add a property to the **Form** business object. Repeat as needed for more properties.
 - d. Click **Finish**.

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

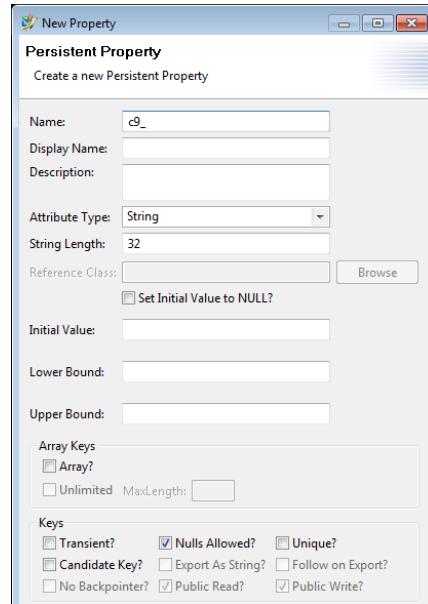
4.2.3 Create the form business object properties

While the **Form** business object is being created, the *properties* should be created. The Business Modeler IDE automatically creates the **Form Storage Class** with attributes that correspond to the business object properties. The form business object shows the properties as **Runtime** properties.



1. Click the **Add** button to the right of the **Properties** table to add a property to the new form business object.
2. In the **New Property** dialog box, enter the following:
 - a. **Name**
 - b. **Display Name**
 - c. **Attribute Type**
 - d. Depending on the **Attribute Type**, enter other property settings.
3. Repeat as needed for additional properties.

4.2.4 Create a persistent property



Use the following values in the **Persistent Property** dialog box for a **String** persistent property:

- **Name**

The property name as you want it to appear in the database. The name cannot contain spaces. A prefix from the template is automatically affixed to the name.

- **Display Name**

Type name as you want it to appear in the user interface.

- **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**

Selecting **Nulls Allowed** allows an empty value for the attribute.

4.2.5 Property constants

Property constants allow you to control whether a business object property is modifiable, visible, enabled, exportable, or required.

The screenshot shows the 'Property Constants' section of a configuration interface. It consists of two tables:

| Property Name | Type | Storage Type | Inherited | Source | COTS |
|-------------------------|-----------|--------------|-----------|---------------------|------|
| archive_date | Attribute | Date | ✓ | POM_application_... | ✓ |
| backup_date | Attribute | Date | ✓ | POM_application_... | ✓ |
| based_on | Runtime | String[0] | ✓ | WorkspaceObject | ✓ |
| c9_AuthorizingDocument | Runtime | String[32] | | C9_Change_Form | |
| c9_ChangeCode | Runtime | String[32] | | C9_Change_Form | |
| c9_ChangeReason | Runtime | String[32] | | C9_Change_Form | |
| c9_EstimatedCost | Runtime | Double | | C9_Change_Form | |
| c9_EstimatedReleaseDate | Runtime | Date | | C9_Change_Form | |

| Name | Value | Overridden | COTS | Template |
|-----------------|----------|------------|------|------------|
| ComplexProperty | | | ✓ | foundation |
| Enabled | false | | ✓ | foundation |
| Exportable | Optional | | ✓ | foundation |
| InitialValue | | | ✓ | foundation |
| Localizable | false | | ✓ | foundation |
| Modifiable | Write | ✓ | | cccddev |
| Required | false | | ✓ | foundation |
| StubProperty | false | | ✓ | foundation |
| Visible | true | | ✓ | foundation |

The following constants can be applied to business object properties:

| Property constants | Description |
|-------------------------|--|
| 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. |

4.2.6 Hide properties on a form business object

Forms in the Teamcenter rich client hold additional information about items. If you do not want a property on a form to be visible to end users, you can hide the property by overriding its **Visible** property constant. You can do this for properties on the **Form** business object and its children.

The screenshot shows the Teamcenter application interface. At the top, there's a toolbar with various icons. Below the toolbar, the main area has a title bar 'Properties' and a status bar at the bottom. The central part of the screen displays two tables:

- Properties** table (top): Shows a list of properties for a specific business object. One row is highlighted in yellow. The columns include: Property Name, Type, Storage Type, Inherited, Source, and COTS. Some properties listed are 'archive_date', 'backup_date', 'based_on', and several starting with 'c9_'. The 'Source' column for these properties points to 'C9_Change_Form'.
- Property Constants** table (bottom): A separate table titled 'Property Constants' with columns: Name, Value, Overridden, COTS, and Template. It lists various properties like 'Enabled', 'Exportable', 'InitialValue', etc., with their corresponding values and overridden status. The 'Visible' property is also listed with its value set to 'false'.

Use the following procedure:

1. Browse to the child of the **Form** business object that has the property you want to hide. To search for a form business object, you can click the **Find** button at the top of the view.
2. Right-click the form business object, choose **Open**, and click the **Properties** tab in the resulting view.
The properties of the form appear in a table.
3. On the **Properties** pane, select the property to hide.
4. In the **Property Constants** table, select **Visible**.
5. Click the **Edit** button to the right of the **Property Constants** table.
6. Clear the **Value** check box to clear the box .
7. Click **Finish**.
8. After deployment, test to ensure the property is hidden in the Teamcenter rich client. Open the form and verify that the property is now hidden.

Note

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

4.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

4.4 Summary

The following topics were taught in this lesson:

- Create **Form** business objects.
- Configure a variety of **Form** properties.

Lesson

5 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 characteristics for list of values (LOV).
- Create a LOV and attach the LOV to a property.
- Create a cascading LOV.
- Create a dynamic LOV.

Help topics

Additional information for this lesson can be found in:

- *Creating lists of values (LOVs)*

5.1

Introduction to lists of values (LOVs)

The **LOV** folder in the **Extensions** folder 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.

| Safety Code | Value | Description |
|-------------|---------------------------------|-------------|
| 1 | Does not meet safety standards. | |
| 2 | Does not meet safety standards. | |
| 35 | Under review for safety | |
| 40 | Safety approval pending | |

There are three main types of lists of values:

- Batch

Store the LOV values in the Teamcenter database rather than storing them in the template. From XML files, you can use the **bmide_manage_batch_lovs** utility to update the LOV values in the Teamcenter database.

- Classic

Store the LOV values in the template. There are three variations:

- Filter LOV
- Cascading LOV (Hierarchical)
- Interdependent cascading LOV

- Dynamic

Read the LOV values dynamically by querying the database.

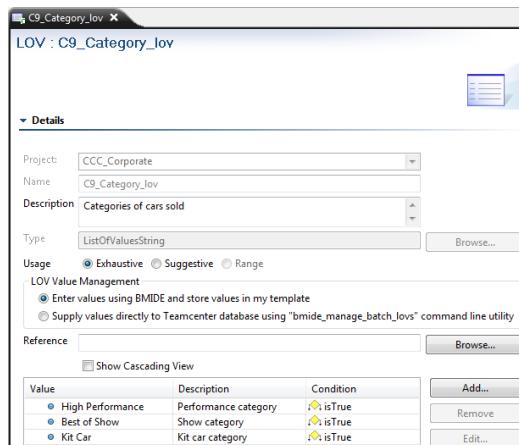
LOV attachments

After you create a list of values, you must attach it to a property on a business object.

| LOV Attachments | | | | |
|------------------------------------|-----------|-----------|----------|------|
| Business Object.Property(Property) | Condition | Inherited | Override | COTS |
| | isTrue | | | |
| | | | | |
| | | | | |

5.1.1 List of values (LOV) interface

In the **Extensions** view, expand the project folder until you see the **LOV** folder. Expand the **LOV** folder to see existing COTS list of values. Double-click a LOV to open it into an editor view.



Type – the LOV type and must match the class attribute type.

Usage choices:

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 Value Management choices:

Enter values using BMIDE and store values in my template stores the LOV values in the template as *static* values. All management of the values is done in the Business Modeler IDE.

Supply values directly to Teamcenter database using "bmide_manage_batch_lovs" command line utility stores the LOV values in the database as *batch* values. Values are supplied to the database using the **bmide_manage_batch_lovs** utility.

LOV Attachments – 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.

| LOV Attachments | | | | |
|------------------------------------|-----------|-----------|----------|------|
| Business Object.Property(Property) | Condition | Inherited | Override | COTS |
| C9_Item.C9_Category | .isTrue | | | |
| | | | | |
| | | | | |

5.1.2 LOV types

An LOV type restricts LOV values to certain LOV value options. For example, if an LOV type is **ListOfValuesInteger**, you can enter only integer values in the LOV.

When you create an LOV, use the **Type** box to select the type. The following table describes the general types of LOV.

| General LOV types | Description |
|-------------------|--|
| <i>type</i> | <p>Users can load the values from the database or enter custom values.</p> <p>The value type is the <i>type</i> selected. The value type can only be attached to a <i>type</i> attribute.</p> <p>For example, the <i>type</i> selected is String. This LOV type can only be attached to a string attribute of a class. If a user adds custom values, only String values can be added.</p> |
| typeExtent | <p>Values are loaded from the database after setting the reference class and reference attribute. Values contain all instances of the class attribute.</p> <p>Users cannot enter custom values.</p> <p>For example, the LOV type is ListOfValuesStringExtent. If the reference class is POM_user and the reference attribute is user_name, the values of the LOV are all the existing user names listed in the database.</p> <p>Or, for example, if the LOV type is ListOfValuesTagExtent, the values are loaded from the database after setting the reference class and reference attribute. Values contain all the instances of the class attribute.</p> |

5.1.3 LOV value types

There are six valid value types used to construct LOVs. Each LOV can contain only one of these value types (that is, value types cannot be mixed in the same LOV).

| Value | Definition |
|------------------|---|
| Integer | Whole number. |
| Double | Double-precision floating point decimal number (sometimes called a <i>real</i>). |
| Char | Single ASCII character. |
| String | String of ASCII characters. |
| Date | Date and time in the format used at your site. |
| Reference | Reference to a list of unique tags in the database (for example, item ID). |

5.1.4 LOV usage types

Each LOV must also be assigned one of three usage types.

| Usage | Definition |
|-------------------|--|
| Exhaustive | Used to define all allowable entries. A user is prevented from specifying a value that is not contained in an exhaustive LOV. |
| Suggestive | Used to provide a suggested list of allowable values. For example, a suggestion LOV could be used to list commonly used description strings. Because description boxes typically accept any user-defined string, the user can select one of the suggested description strings from the LOV or enter another user-defined string. |
| Range | Used to provide the user with a constrained subset of allowable entries. For example, a range LOV could be used to construct a small consecutive list of serial numbers. A user is prevented from specifying a value not contained within a range LOV. |

5.1.5 List of values (LOV) types

Mainstream types of lists of values (LOVs).

- **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).
- **ListOfValuesFilter** – Reference to another nonfiltered LOV with the capability to filter the referenced LOV's values.
- **ListOfValuesInteger** – Whole number.
- **ListOfValuesString** – String of ASCII characters.

Note

The ability to store LOV values in the database as batch LOVs is available only on the following LOV types: **ListOfValuesChar**, **ListOfValuesDate**, **ListofValuesDouble**, **ListOfValuesInteger**, and **ListOfValuesString**.

Other types of lists of values (LOVs). Entries containing **Extent** must match an existing value in the database.

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.

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

ListOfValuesIpClassification – An intellectual property classification.

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.

Special conditions for the following LOVs

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.

5.1.6 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. Use **BMIDE**→**Organize Extensions**→**Set active extension file** to select the file where you want to save the data model changes, for example, **lovs.xml**.
2. Start the **New LOV** wizard.
 - On the menu bar, choose **BMIDE**→**New Model Element**, type **LOV** in the **Wizards** box.
 - Open the **Extensions** folder, right-click the **LOV** folder, and choose **New LOV**.
3. Choose the **LOV** type: Classic LOV, Dynamic LOV, or Batch LOV.

The **New LOV** responds based on your selection.

In the **New LOV** wizard, enter the following information in the **LOV** dialog box:

1. Type the **Name** and **Description** you want to assign to the new LOV.
2. Click the **Browse** button to the right of the **Type** box to select the LOV business object for the LOV you want to create.
3. In the **Usage** section, click one of **Exhaustive**, **Suggestive**, or **Range**.
4. Click the **Add** button to the right of the **LOV Values** table to display the **Create** dialog box.
 - In the **Value** box, type the value of the LOV.
 - In the **Description** box, type a brief description of the LOV value.
 - In the **Value Display Name** box, type the value name as you want it to appear in the user interface.
 - Click **Finish**. The value is added to the **LOV Values** table.
5. Change the LOVs as desired by using the **Remove**, **Edit**, **Move Up**, **Move Down**, and **Clear** buttons.
6. Click **Finish**. The new LOV appears in the **LOV** folder.

Note

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.

5.1.7 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. Use **BMIDE**→**Organize Extensions**→**Set active extension file** to select the file where you want to save the data model changes, for example, **lovs.xml**.
2. Open the **Extensions\LOV** folder and expand the folder: **Classic LOV**, **Dynamic LOV**, or **Batch LOV**.
3. Right-click the LOV you want to attach and choose **Open**. The LOV details appear in a new view.
4. Scroll to the bottom of the view and click the **Attach** button under the **LOV Attachments** heading.
5. Perform the 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.

Note

Only those conditions appear that have valid signatures. For LOV attachments, the valid condition signature is as follows:

condition-name(UserSession)

- 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.

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.

5.1.8 Add, remove, or clear a value to an LOV

1. Use **BMIDE**→**Organize Extensions**→**Set active extension file** to select the file where you want to save the data model changes, for example, **lovs.xml**.
2. Open the **Extensions\LOV** folder and expand the folder: **Classic LOV**, **Dynamic LOV**, or **Batch LOV**.
3. Select the LOV to modify in the **LOV** folder. The LOV values appear.
4. Make the desired changes in the **LOV** dialog box.
 - Click **Add** to add a value. Then type the **Value**, **Description**, and **Value Display Name**.
 - Select an existing value and click **Edit** to edit a value. Then modify the **Value**, **Description**, and **Value Display Name**.
 - Click **Remove**, **Move Up**, **Move Down**, or **Clear** to change the **LOV**.
 - If you chose the **Range** usage, fill in the high and low values for the range in the **Upper** and **Lower** boxes.

Note

If the **LOV** is a **Range**, fill in the high and low values for the range in the **Upper** and **Lower** boxes.

5. When your changes are finished, click **Finish**.
6. Choose **BMIDE**→**Save Data Model**, or click the **Save Data Model** button on the main toolbar.

5.1.9 Set display names for lists of values (LOVs)

Lists of values (LOVs) are the pick lists displayed in Teamcenter when end users click an arrow in a data entry box. You can define the text that displays for string LOVs in the Teamcenter user interface.

1. In the **Extensions** view, open the **LOV** folder.
2. Open an LOV and select a value in the LOV table.
3. Click the **Localization** button to the right of the table.
The **LOV Value Localization** dialog box is displayed.
4. In the **LOV Value Localization** dialog box, click the **Add**, **Edit**, or **Remove** button. Select an existing value to click the **Override** button.
The **Localization** dialog box is displayed.
5. Perform the following in the **Localization** dialog box:
 - a. In the **Value Localization** box, type the value as you want it to display in the user interface.
 - b. If a description was previously entered for the LOV value, in the **Description Localization** box, type a description for the display text.
 - c. Click the arrow in the **Locale** box to select the language locale where the text is used.
 - d. In the **Status** box, select the status of the text change in the approval life cycle.
 - e. Click **Finish**.
6. Click **Finish** in the **LOV Value Localization** dialog box.

5.2 Classic LOVs

Filter LOVs

A LOV created by a Teamcenter administrator from another existing LOV. The values would typically be a subset of the existing LOV, but the values can be the same.

Cascading LOVs (Hierarchical)

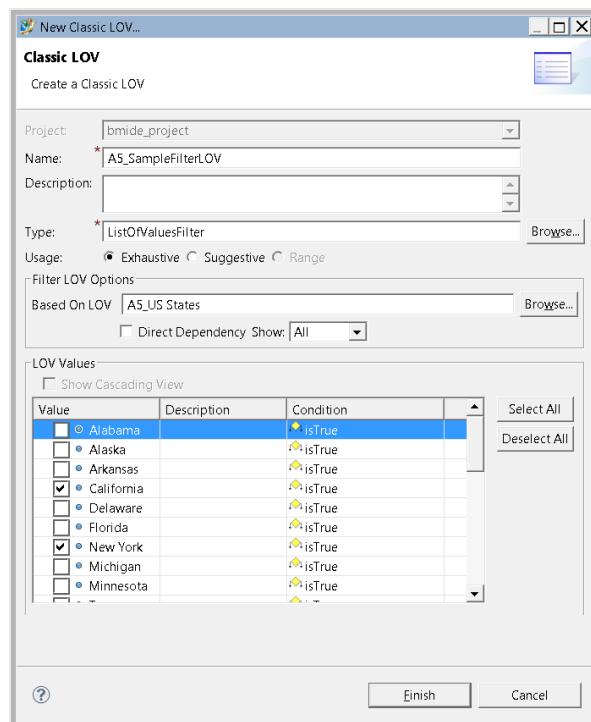
Cascading LOVs allow you to organize and present your lists in a hierarchy. This is especially beneficial for long lists of organized values, and allows for end users to choose a value in the list in an organized manner.

Interdependent cascading LOVs

Interdependent LOVs are like cascading LOVs in that they allow you to organize and present your lists in a hierarchy. Interdependent LOVs have the extra feature of saving values into multiple property fields (for example, saving each value selected while stepping down the hierarchy).

5.2.1 Filter LOV

A **Filter LOV** is created from another existing LOV – the **Based On LOV**. The **Filter LOV** may contain all or a portion of the values from the **Based On LOV**. When creating the **Filter LOV**, the administrator will select all (**Direct Dependency**) or only the values you want to use. In the **Show** box, select **All** to show all the values of the original LOV or **Selected** to show only selected values. Then in the **LOV Values** pane, select only the values you want to use.



Creating a filter LOV

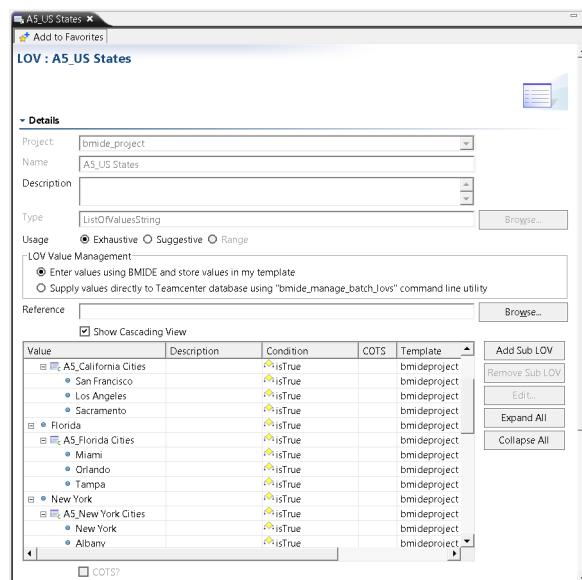
5.2.2 Cascading LOV

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. A cascading LOV allows you to organize and present your lists in a hierarchy, which is especially beneficial for long lists.

The cascading LOV consists of two LOVs.

- A main classic LOV to hold the sub-LOVs.
- The sub-LOVs that can be used under the main LOV.

For example, a LOV that contains a list of states, and another LOV for each state that contains a list of cities.



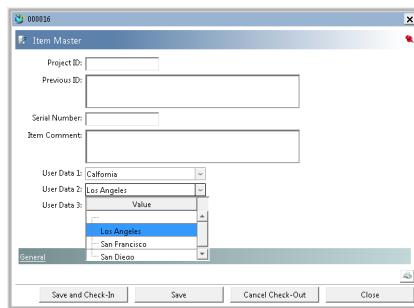
Creating a cascading LOV

5.2.3 Interdependent LOV

An interdependent attachment to a cascading LOV lists the appropriate property values when you select a value higher in the LOV tree. For example, if you have a cascading LOV of states that contains cities sub-LOVs, you can use this so that the end user selects the state and then is prompted to select the city.

Example

In a States LOV, select the California value and then select the California city



Interdependent LOV in the rich client user interface

Note

The interdependent LOVs must be balanced to work. If you set up an LOV that is interdependent, each branch must have an LOV attached. If you add an attachment to one level, all in that level must have one. Otherwise, the **Attach** button is not available.

5.2.4 LOV loaded from the database

A LOV can be created with values that are loaded from the Teamcenter database. The LOV is created and attached to a property as described earlier with a basic LOV.

To load values from the Teamcenter database, the **Reference** box is used and will contain a Teamcenter *class.attribute* value. Depending on the values you want in the LOV, you choose the Teamcenter *class.attribute*.

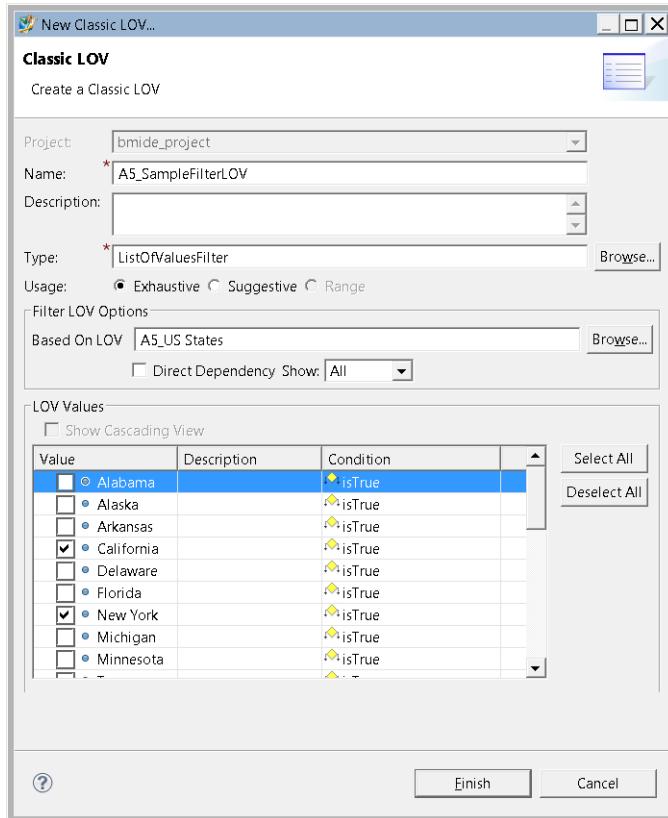
Note

The LOV will show the same values until you perform another **Load** from the Teamcenter database. A *dynamic* LOV can be set up, but it requires a Teamcenter program to keep the LOV up-to-date.

With a value in the **Reference** box, the **Load** button becomes active and when pressed displays the **Teamcenter Repository Connection**. As with a hot deploy, you enter the **Teamcenter Repository Connection** login information and connect to the database. Values from the Teamcenter database, specifically from the *class.attribute* instances, are loaded into the LOV.

5.2.5 Create a filter LOV

Right-click the **LOV→Classic LOV** folder and choose **New Classic LOV**.

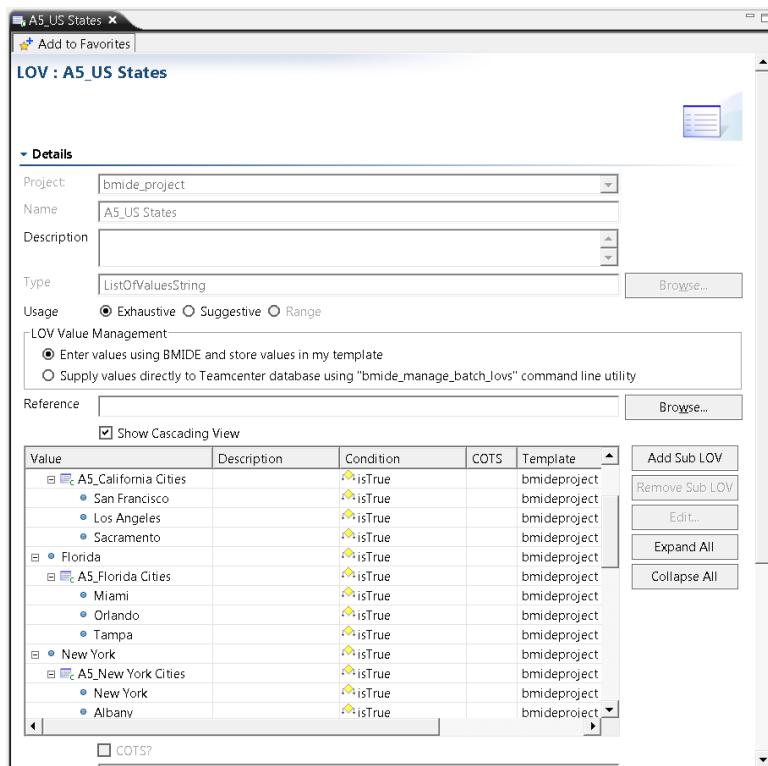


New LOV dialog box information.

- **Name** box – the name you want to assign to the new LOV.
- **Type** box – **ListOfValuesFilter**.
- **Based on LOV** box – click the **Browse** button to choose the LOV you want to base the filter LOV on.
- **Direct Dependency** check box – new LOV values to be derived directly from the original LOV.
- **Show** box – use **All** to show all the values of the original LOV or **Selected** to show only selected values.
 - o Select **All** to show all the values of the original LOV.
 - o Select **Selected** to show only selected values of the original LOV.
- **LOV Values** pane – select only the values you want to use.

5.2.6 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.



Follow these steps:

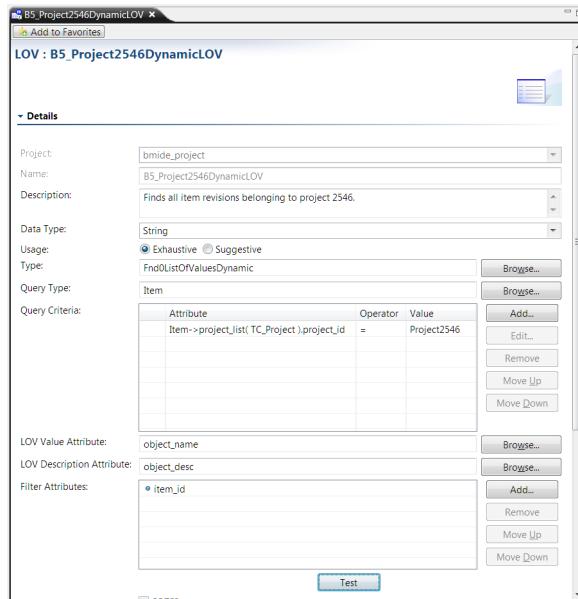
1. Create a main LOV to hold the sub-LOVs. Then create sub-LOVs that can be used under the main LOV. For example, create an LOV that contains a list of states, and then create LOVs for each state that contain a list of cities.
2. Right-click the main LOV in the **Extensions** view, choose **Open**, and select the **Show Cascading View** check box on the LOV.
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. After you add the sub-LOVs, they appear in a cascade format in the main LOV.
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.
4. Attach the main LOV to a property to the **LOV Attachments** table and choosing **Attach**.

5.3 Dynamic LOVs

Lists of values (LOVs) are lists on property boxes in the user interface. Dynamic LOVs show a list of values obtained dynamically by querying the database.

For example, if you want a list of values of all aluminum widgets in the database that have a blue glossy finish, you can create a dynamic LOV to query the database for them. Or if you want a list of values of all the members of the widget development project, you can create a dynamic list of values to find them. Dynamic lists of values are not static but change as the data in the database changes. They allow end users to select from lists composed of data that is active in the database.

The following sample dynamic LOV queries for all items belonging to a certain project.



When you attach the dynamic LOV to a property and install the template to the server, the query results are displayed in the end user interface as a list of values.

Dynamic list of values in the end user interface

| Affected item in project: | | | |
|---------------------------|----------------------|--------|--|
| Name | Description | ID | |
| Housing seats | The plastic seats on | 000019 | |
| Housing base | Aluminum base to | 000021 | |
| Mounting brackets | Brackets used to | 000020 | |

5.3.1 Create dynamic lists of values

Dynamic lists of values (LOVs) show a list of values obtained dynamically by querying the database. You can query for product data from objects such as items, parts, and datasets, or query for administrative data from categories such as users, groups, and roles. For example, you could create a list of values that queries for all items supplied by a particular vendor, or all users in a certain group.

1. Start the New Dynamic LOV wizard.
 - On the menu bar, choose **BMIDE→New Model Element**, type **Dynamic LOV** in the **Wizards** box, and click **Next**.
 - Open the **Extensions\LOV** folders, right-click the **Dynamic LOV** folder, and choose **New Dynamic LOV**.
2. Perform the following steps in the **New Dynamic LOV** dialog box:
 - a. Type the **Name** and **Description** you want to assign to the new LOV.
 - b. Click the arrow in the **Data Type** box to select the data type of the LOV values: **String**, **Integer**, **Date**, **Double**, or **Tag** (reference to a unique tag in the database, for example, item ID).
 - c. In the **Usage** section, click one of the following buttons: **Exhaustive** or **Suggestive**.
 - d. The **Type** box defaults to the **Fnd0ListOfValuesDynamic** business object, which is the business object used to define dynamic lists of values.
3. Define the **Query Type** for the new dynamic LOV. You will also test the **Query Criteria** from the new dynamic LOV.
4. Attach the new dynamic LOV to a property.

When an end user clicks the arrow on the property in the user interface, the LOV table displays the query results.

5.3.2 Create dynamic LOV query

Perform the following steps in the **New Dynamic LOV** dialog box to define the **Query Type** and **Query Criteria**:

1. Click the **Browse** button to the right of the **Query Type** box to select the first business object type to query.

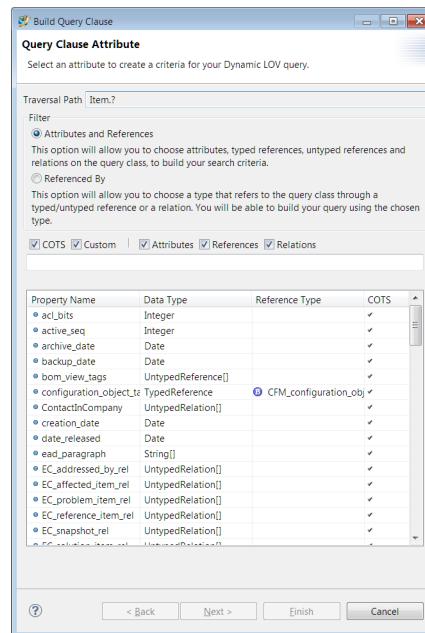
For example, if you want to query for properties on item business objects, select **Item**. If you want to query for system users, select **User**.

2. Click the **Add** button to the right of the **Query Criteria** table to select the attributes to include in the query.

- a. Select one of the **Filter** options:

- **Attributes and References** – Displays attributes, typed references, untyped references, and relations on the query class.
- **Referenced By** – Displays typed and untyped references on a relation. When you select this and then select an attribute in the table below, you must click **Next** to select the referenced attribute.

- b. Select an attribute in the table.



Building the query clause

- c. After you add an attribute to the **Query Criteria** table, add the following to complete the criteria:

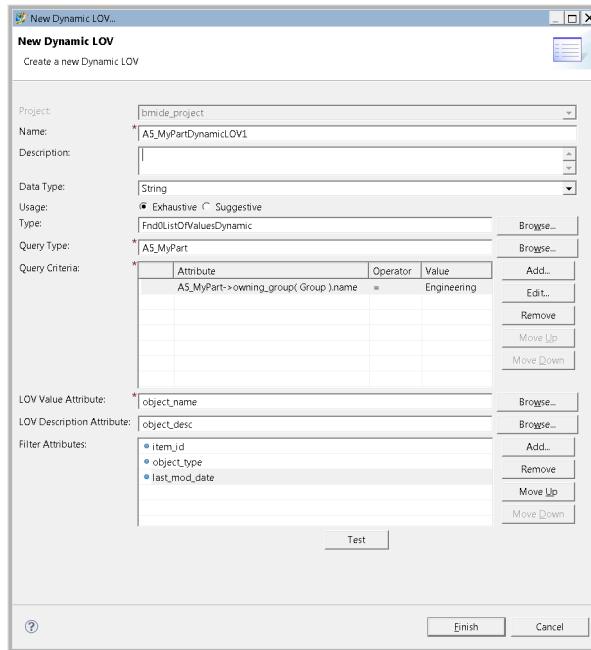
Click the arrow in the **Operator** box to select the operator to evaluate the attribute.

| Operator | Description |
|--------------------|--------------------------|
| = | Equal to |
| > | Greater than |
| < | Less than |
| != | Not equal to |
| >= | Greater than or equal to |
| <= | Less than or equal to |
| IS_NULL | Empty |
| IS_NOT_NULL | Not empty |

In the **Value** box, type the value against which the selected attribute has to be evaluated.

3. In the **LOV Value Attribute** box, select the property to be used as the value of the LOV displayed in the end user interface. Click the **Browse** button to the right to make the selection.
4. In the **LOV Description Attribute** box to select the property to be used as the description of the LOV displayed in the end user interface. Click the **Browse** button to the right to make the selection.

- Add properties in the **Filter Attributes** box to view as columns in the list of values table in the end user interface. These provide more information about each value to help the end user. Also, the end user can click the column headings to sort (filter) the lists of values.



Sample dynamic LOV

- Click the **Test** button to send the query to the database and view the returned results. The results of the query are displayed as the LOV table that appears in the end user interface.

| LOV Value (object_name) | LOV Description (object_desc) | item_id | object_type | last_mod_date |
|-------------------------|---|---------|-------------|-------------------|
| 1.5 inch lug nut | Main drive shaft lug nut | 000020 | A5_MyPart | 01-Nov-2012 12:00 |
| Cotter pin | Cotter pin to secure lug nut placement | 000021 | A5_MyPart | 01-Nov-2012 12:01 |
| Front drive shaft | Drive shaft for the front sprocket assembly | 000019 | A5_MyPart | 01-Nov-2012 11:58 |
| Sprocket | Primary sprocket in front assembly | 000018 | A5_MyPart | 01-Nov-2012 11:43 |

Results of testing a dynamic list of values

5.3.3 Considerations for creating dynamic LOVs

Following are considerations to keep in mind when you create dynamic lists of values:

- If the business object in the **Query Type** box is revisable (like **ItemRevision**), the values for all queried instances are displayed in separate rows when you click the **Test** button. If you want only the values from the latest revision to be displayed, you must add the following line in the **Query Criteria** table:

```
AND query-type.active_seq != 0
```

- If the properties in the **LOV Value Attribute**, **LOV Description Attribute**, or **Filter Attributes** boxes are variable length arrays (VLAs), all the values for the properties in the queried instances are displayed in separate rows when you click the **Test** button. In addition, the results are not sortable if you click the column header; the results are displayed unsorted as they are retrieved from the server.
- If the property in the **LOV Description Attribute** box is a variable length array (VLA), when you attempt to create an interdependent LOV using that dynamic LOV, the **Attach Description** button in the **Interdependent LOV** dialog box is disabled. Attaching a description in interdependent LOVs is not supported for VLA properties.
- If a variable length array (VLA) property is selected as an LOV value, all the values from the array of the selected instance are displayed. If any of the values from the array are invalid per the query condition, and that value is selected in the rich client, a error message is displayed, for example:

```
The Value "80.5" is not valid for the property "a2AttachDouble" of type "Double".
```

- When you create a dynamic LOV and select a date attribute in the **Query Criteria** table, if you use the **=** operator to find items with a particular date, you do not get the expected results when you click the **Test** button. That is because hours and minutes are saved on date attributes, and you must enter the exact hour and minute to query data attributes using the **=** operator. Instead, use the **>=** or the **<=** operators.

Also keep in mind that the Business Modeler IDE displays date and time in Greenwich Mean Time (GMT), and the rich client displays the date and time in the end user's local time. As a result, there can be a mismatch between the date attribute values displayed in the Business Modeler IDE and those retrieved from the database.

5.4 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

5.5 Summary

The following topics were taught in this lesson:

- Describe characteristics for list of values (LOV).
- Create a LOV and attach the LOV to a property.
- Create a cascading LOV.
- Create a dynamic LOV.

Lesson

6 *Relation business object configuration*

Purpose

The purpose of this lesson is to extend Teamcenter with a **Relation** business object and configure properties on the relation.

Objectives

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

- Create a **Relation** business object.
- Configure properties on the **Relation** business object.

Help topics

Additional information for this lesson can be found in:

- *Create a relation business object*

6.1

Introduction to relation business objects

Following are some common types of relation business objects. These are all children of the **IManRelation** business object. To see the available relations that can be used to relate source and target objects, in the rich client My Teamcenter application, copy a source object, select a target object, and choose **Edit→Paste Special**. The **business_object_default_relation** preference specifies the default relation that is created when an object is pasted under an instance of the business object.

- Specification relations

The **IMAN_specification** business object defines this relation.

Specification relations are detailed methods, designs, processes, and procedures used to satisfy requirements. A specification relationship can only be established with an item revision, not an item. Although requirements may remain fairly constant for a product (item), actual manufacturing methods, designs, processes, and procedures may change drastically from model to model (item revisions).

- Requirement relations

The **IMAN_requirement** business object defines this relation.

Requirement relations are criteria that must be satisfied by this item or item revision. However, requirements often do not specify how this criteria should be satisfied. For example, a requirements relation may specify maximum weight for an item revision but not how to construct it. Extend the business object to create your own specification relations.

- Manifestation relations

The **IMAN_manifestation** business object defines this relation.

Manifestation relations are nondefining snapshots of a particular aspect of an item or item revision at a particular moment in time. For example, numerically controlled (NC) program files are a common manifestation. Consider that they represent one aspect of an item revision (that is, machining information) and that this information is only accurate as long as the item revision does not change. If the item revision does change, the NC program files may no longer be accurate and may need to be re-created.

- Reference relations

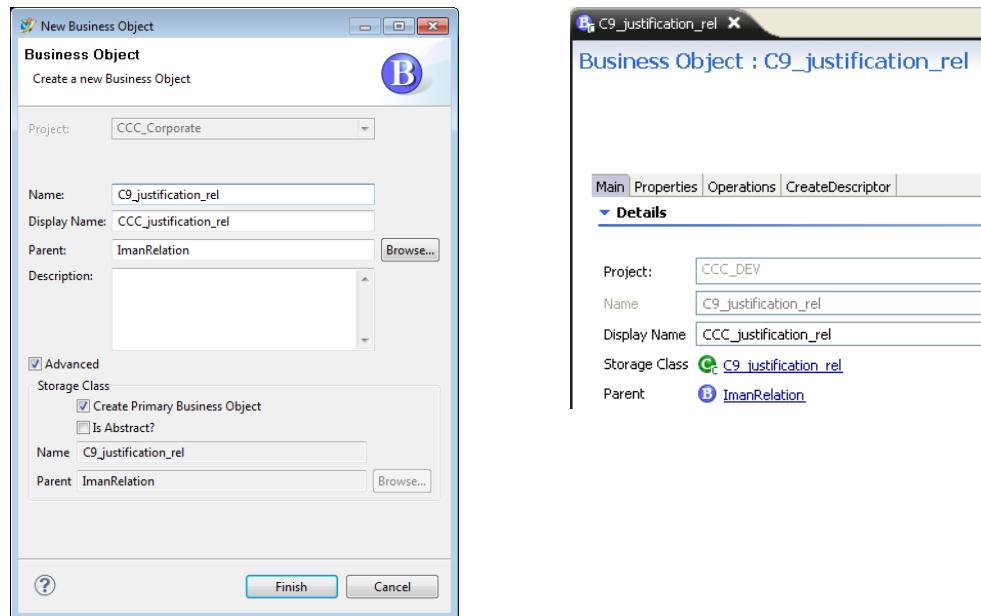
The **IMAN_reference** business object defines this relation. Reference relations describe a general nondefining relationship of a workspace object to an item or item revision. This relation type can be thought of as a miscellaneous relation type. Typical examples of reference relations are white papers, field reports, trade articles, customer letters, lab notes, and so on.

6.2

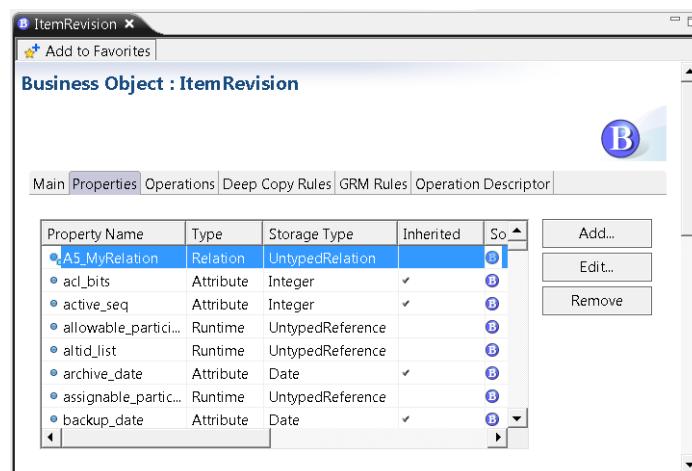
Defining a new relation

Data objects created by users in Teamcenter are related to one another using a relation, which are defined in the Business Modeler IDE as *relation business objects*.

- To create a new relation business object, right-click the **ImanRelation** business object or one of its children and choose **New Business Object**.



- A relation property must be added to the business object for the new relation.



Additional points when defining a new relation

- Available relationships are children of the **ImanRelation** business object.
- To make attached objects of the relation property display under the target business object, you must modify the **<Type_Name>_DefaultChildProperties** preference to include your new relation property.
- To set the relation property as a default paste relation, you must add it to the *business-object-name_default_relation*.
- After a new relation business object is added, create a GRM rule to relate items using the new relation.
- You can use Generic Relationship Management (GRM) rules to limit what objects can be pasted to other objects.
- The new relation can be used to define a compound property for a path that exists between the two objects.
- Deep copy rules can be defined to include rules for the new relation.

6.2.1 New relation properties

Relation properties are properties that define the relationship between objects. For example, a dataset can be attached to an item revision with a specification, requirement, or a new relation.

| Property Name | Type | Storage Type |
|-------------------|-----------|------------------|
| c9_Attach_Date | Attribute | Date |
| c9_Attaching_User | Attribute | String[32] |
| lsd | Attribute | Date |
| object_properties | Attribute | Short |
| owning_site | Reference | TypedReference |
| pid | Attribute | Integer |
| primary_object | Reference | UntypedReference |
| relation_type | Reference | TypedReference |

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

6.2.2 New relation property constants

Property constants apply to relation properties as they did with previous business object properties. Because these constants are attached to properties, they are inherited, just like the properties themselves.

| ▼ Property Constants | |
|----------------------|----------|
| Name | Value |
| ComplexProperty | |
| Enabled | false |
| Exportable | Optional |
| InitialValue | |
| Localizable | false |
| Modifiable | Write |
| Required | false |
| StubProperty | false |
| Visible | true |

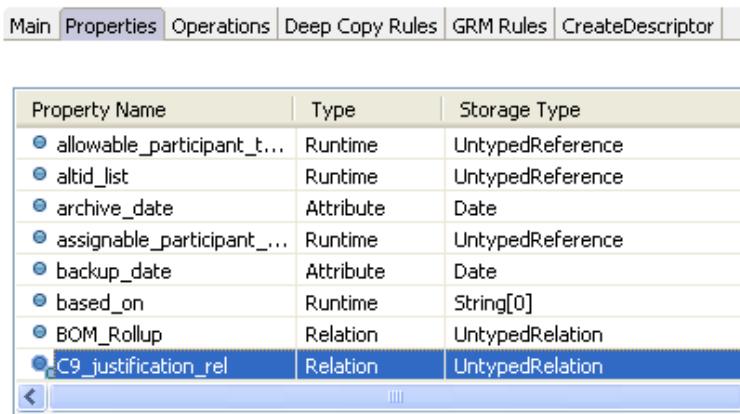
You can create property constants for a number of situations. Some examples are:

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

6.2.3 Business object property of new relation

Relation properties on business objects are properties that define the relationship between objects. For example, a dataset can be attached to an item revision with a specification, requirement, or a new relation.

In this example, **C9_justification_rel** is a **Relation** property on the **C9_CommonItemRevision** business object.



| Property Name | Type | Storage Type |
|-----------------------------|-----------------|------------------------|
| allowable_participant_t... | Runtime | UntypedReference |
| altid_list | Runtime | UntypedReference |
| archive_date | Attribute | Date |
| assignable_participant_... | Runtime | UntypedReference |
| backup_date | Attribute | Date |
| based_on | Runtime | String[0] |
| BOM_Rollup | Relation | UntypedRelation |
| C9_justification_rel | Relation | UntypedRelation |

Key points

- Before the new relation can be used with a business object, the relation must be added.
- Other properties are inherited, the relation property is inherited to business objects lower in the business object view.

6.2.4 Create a new relation business object

Use the **ImanRelation** business object or its children to create a new relation business object.

The **New Business Object** wizard is used to create a new relation business object.

Use these steps to create additional **ImanRelation** business objects to use with other business objects.

1. Locate the **ImanRelation** business object:
 - a. In the **Business Objects** view, click the **Find Business Object** button.
 - b. In the filter string box, type **ImanRelation**.
 - c. Double-click **ImanRelation**.

The **Business Objects** view expands to the **ImanRelation** folder.

2. Right-click the **ImanRelation** business object and choose **New Business Object**.
3. In the **New Business Object** dialog box, enter the following information:
 - a. Type **Name**, **Display Name** and **Description** values for the new relation business object.
 - b. Use the following default settings:
 - Create primary Business Object**.
 - Is Abstract?**
 - c. 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.

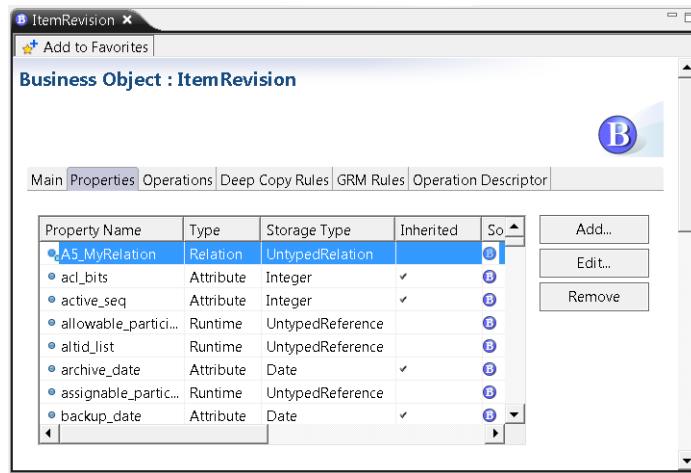
4. Open the new relation into the editor view to add properties.

6.2.5 Add a relation property

Relation properties are properties that define the relationship between objects. For example, a dataset can be attached to a custom item revision with relations such as a specification, requirement, or reference. You can also create your own custom relation business objects.

1. In the **Business Objects** folder, right-click the custom business object to which you want to add the property, choose **Open**, and click the **Properties** tab in the resulting view.
2. Click the **Add** button to the right of the properties table.
3. Under **Property Types**, select **Relation**. Click **Next**.
4. Perform the following steps in the **Relation Property** dialog box:
 - a. Click the **Browse** button to the right of the **Relation Business Object** box and select the relation business object you want to use, for example, **IMAN_specification**. The available business objects are children of the **ImanRelation** business object.
 - b. In the **Description** box, type a description of the new relation property.
 - c. Click **Finish**.

The new relation property is added to the property table.



5. After deployment, test your new property in the Teamcenter rich client.

For example, if you added a relation property to a custom child of the **ItemRevision** business object, in the My Teamcenter application, select an instance of that kind of item revision and attach a dataset to it using the new relation property by using the **Edit→Copy** and **Edit→Paste...** option.

6.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

6.4 Summary

The following topics were taught in this lesson:

- Create a **Relation** business object.
- Configure properties on the **Relation** business object.

Lesson

7 *Dataset business object configuration*

Purpose

The purpose of this lesson is to extend Teamcenter with a **Dataset** business objects and a **Tool** extension.

Objectives

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

- Create a **Dataset** business object.
- Configure the new dataset business object to use an existing Teamcenter tool.
- Create a **Tool** extension.

Help topics

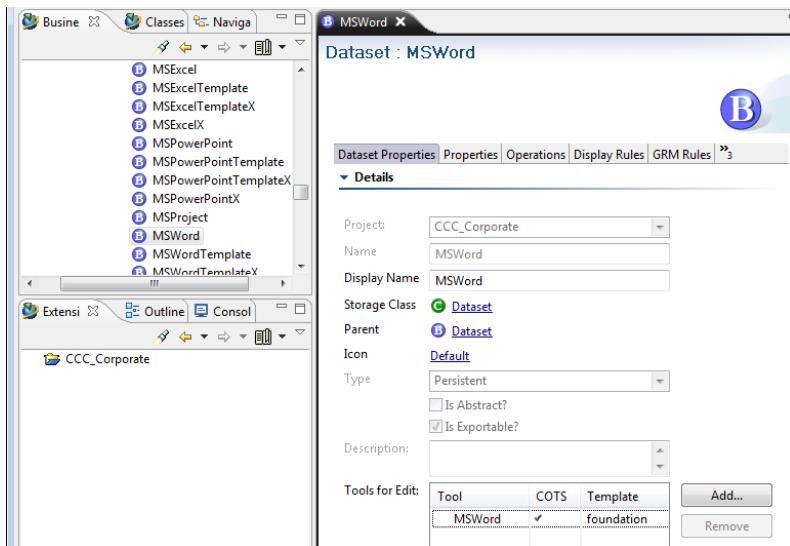
Additional information for this lesson can be found in:

- [*Create a dataset business object*](#)
- [*Add a tool*](#)

7.1

Introduction to dataset business objects

Datasets are objects used to manage file data associated with external software applications.

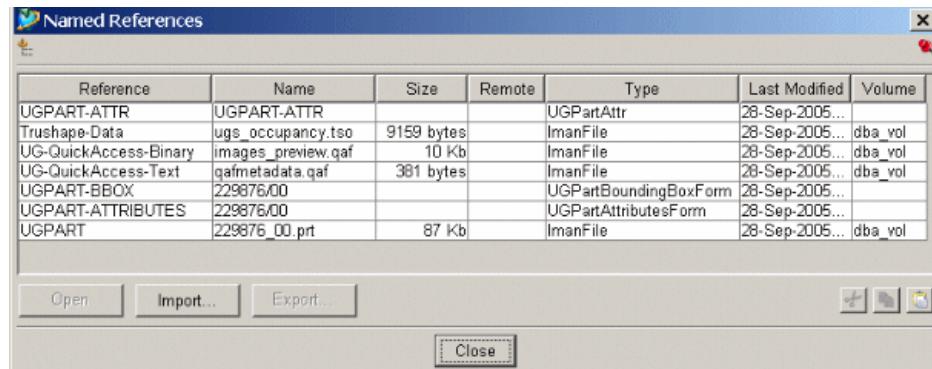


Datasets typically consist of a single application data file or logical groupings of application data files. There are numerous types of datasets predefined in Teamcenter. However, your site may need to add more types to be able to manage your site's specific application data files and the viewing/editing software applications associated with these files.

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.

7.1.1 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 type of applications associated with a dataset type.

7.1.2 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:

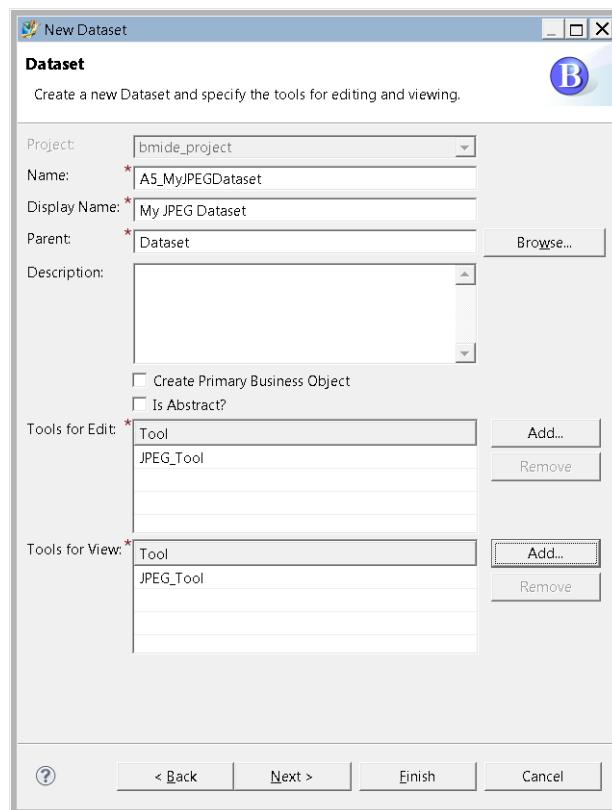
- Importing document, images, or other data created outside the database
- Migrating legacy data

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

- Rich client interface
- The **import_file** utility for batch import

7.1.3 Create a dataset business object

1. Set the active extension file to where you want to save the data model changes.
2. Open the New Dataset wizard with one of these methods:
 - Choose **BMIDE**→**New Model Element**, type **Dataset** in the **Wizards** box, and click **Next**.
 - In the **Business Objects** folder, right-click the **Dataset** business object and choose **New Business Object**.



Creating a custom dataset business object

Note

The **Tools for Edit** pane and **Tools for View** pane require an existing tool definition.

3. In the **Dataset** dialog box, enter the following information:

- **Name**

The name cannot contain spaces. A prefix from the template is automatically affixed to the name.

- **Display Name**

Type the name as you want it to appear in the user interface.

- **Description**

Type a description of the new business object.

- Use these default settings:

- Create primary Business Object.**

- Is Abstract?**

- **Tools for Edit**

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.

- **Tools for View**

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.

Click **Next** if you want to add references and parameters to the dataset. Otherwise, click **Finish** to complete the dataset creation.

7.1.4 Set up the dataset named references

The file name references to associate with the dataset must be defined to complete the **Dataset** business object creation. **Tool** parameters must be associated with the file name reference(s) for the possible actions the user may perform on the dataset, for example view, edit and print.

Provide the **References** information to add file name references to associate with the dataset in the **Dataset References** dialog box:

- **Reference**
Type unique name for this file reference.
- **File Type**
Specify the file name extension (for example, txt, pdf, doc, and so on).
- **Format**
Choose whether the files are binary, object, or text.

Provide the **Tool Action** information to modify the action that a software application takes when a dataset is launched in the **Add/Modify Dataset Tool Action** dialog box:

- **Tools**
Select the tool you previously chose to use for viewing or editing this dataset. The only available tool is the one you previously selected.
- **Operations**
Choose the operation to use with the dataset. For example, **Open**, **OpenUsing**, **Print**, **PrintUsing**, or **Send**.

Note

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.

- **References**
The file name extensions to associate with the action. 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.
- **Parameters**
Parameters to be passed with the action.

7.1.5 Configure dataset view and open

In the Teamcenter rich client, you can select a dataset, and choose **File→Open With** to choose the tool to open the dataset or choose **File→View With** to choose the tool to view the dataset.

By default, a set of tools are assigned to each kind of dataset. These tools are initially set up when the dataset is created.

If you want to be able to use a different tool to open or view a dataset, you can add it to the set of available tools for that dataset type. For example, if you want to use Microsoft Word to open text datasets, you can add it to the tool set.

Using this example, *provide this information in the Business Modeler IDE:*

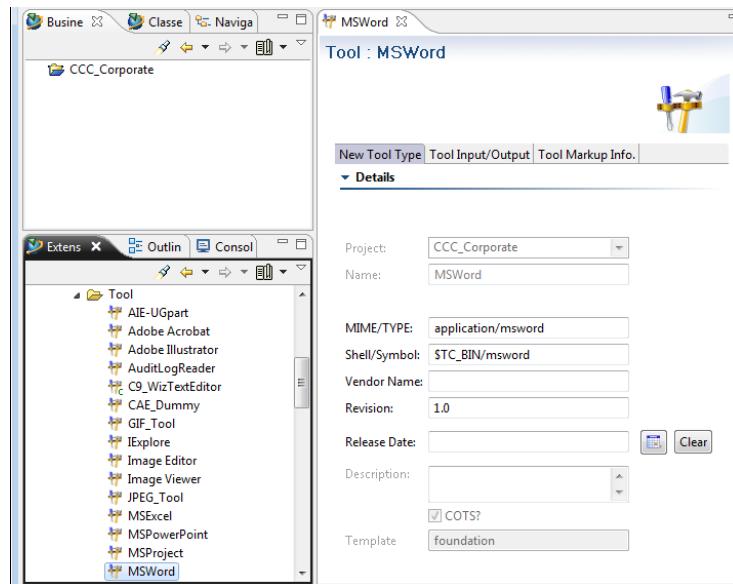
1. The program executable for the tool:
 - Open the **MSWord** tool.
 - **Shell/Symbol** – set the executable as **winword.exe**.
2. The tool to be used to view and open the dataset:
 - In the **Business Objects** view, open the **Text** dataset business object.
 - **Tools for Edit** – choose the **MSWord** tool.
 - **Tools for View** – choose the **MSWord** tool.
 - In the **Add/Modify Dataset Tool Action** dialog box, for the **Tools** box choose the **MSWord** tool.
 - o **Open** – choose the **MSWord** tool.
 - o **OpenUsing** – choose the **MSWord** tool.

From the example, *set up the MIME types so that the tool can be used to open the dataset.:*

1. Create new action for the file type:
 - In **Folder Options** choose **TXT**, and in the **Edit File Type** dialog box, click **New**.
 - In the **New Action** dialog box, type **Open with MSWord** in the **Action** box and located the **winword.exe** file.
2. Modify the dataset preferences using the My Teamcenter application in the rich client:
 - With **Edit→Options**, select **Dataset**.
 - **Dataset Type** – select **Text**.
 - **Default Tool** – select **MSWord**.
 - Select the **Use MIME Type to Search Application for Default Tool** check box.

7.2 Introduction to tools

Tool describes a software application behaving in a specific manner.



The Teamcenter tool definition relates to the operating system application by specifying the MIME/type definition for the software application on the workstation. Tools are used by dataset business objects.

7.2.1 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.

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.

1. Prepare to define the tool.
 - Set the active extension file to where you want to save the data model changes, for example, **options.xml**.
 - Expand the project and the **Options→Tool** folders.
 - Right-click the **Tool** folder and choose **New Tool**.
2. Provide this information to define the tool.
 - **Name**
 - **Description**
 - **MIME/TYPE** (for 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.
 - **Shell/Symbol**
The full path and program name to be run in a shell. For example, for Microsoft Word, type **\$TC_BIN/msword**.
 - **Vendor Name**
 - **Revision**
 - **Release Date** (the release date of the application)
3. Provide this information in the **New Tool Input/Output** dialog box:
 - **Input**
Type of data the tool *accepts*, for example, **ASCII** or **Binary**.
 - **Output**
Type of data the tool *outputs*, for example, **ASCII** or **Binary**.

4. Provide this information in the **New Tool Markup Information Page** dialog box to define view and markup capabilities:
 - In the **Mac Launch Command** box, type the command to launch the tool in the Macintosh operating system, for example, **sample.app**.
 - In the **Win Launch Command** box, type the command to launch the tool in the Windows operating system, for example, **sample.exe**.
 - Select the **Download Required?** check box to indicate that the application launcher must download any files before launching the application.
 - Select the **Callback Required?** check box to enable callback through the application launcher for a non-Teamcenter application.
 - Select the **View Capable?** check box to indicate whether the defined application can view files.
 - Select the **Markup Capable?** check box to indicate whether the defined application can perform markups.
 - Select the **Embed Application?** check box to indicate the defined application is an embedded rich client tool. For these tools, the **Shell/Symbol** box contains the command to launch the embedded tool.
 - Select the **VVI Required?** check box to indicate that the defined application accepts Velocity Vector Imaging (VVI).
 - Select the **Digital Signature Capable?** check box to indicate whether the defined application can do digital signing.

Note

The digital signature capability is currently not used.

7.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

7.4 Summary

The following topics were taught in this lesson:

- Create a **Dataset** business object.
- Configure the new dataset business object to use an existing Teamcenter tool.
- Create a **Tool** extension.

Lesson

8 *Option extensions and BMIDE reports*

Purpose

The purpose of this lesson is to extend Teamcenter with options that work with Teamcenter business objects. In this lesson you will also use the Business Modeler IDE reports.

Objectives

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

- Define and create a note type option.
- Define and create status type options.
- Define and create units of measure.
- Compare data model from two different sources with the Business Modeler IDE reports.

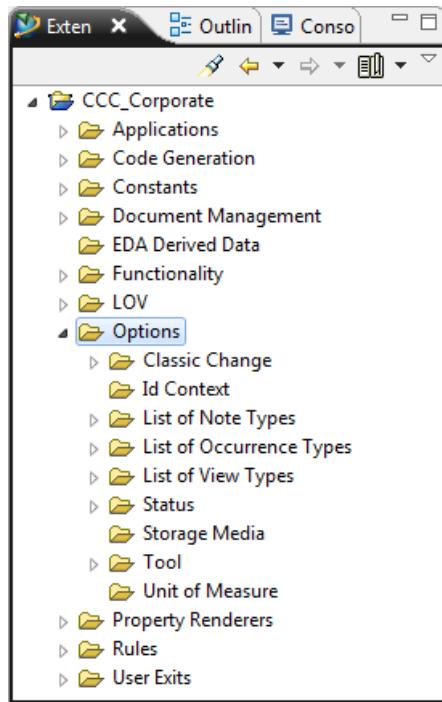
Help topics

Additional information for this lesson can be found in:

- *Creating options*

8.1 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.

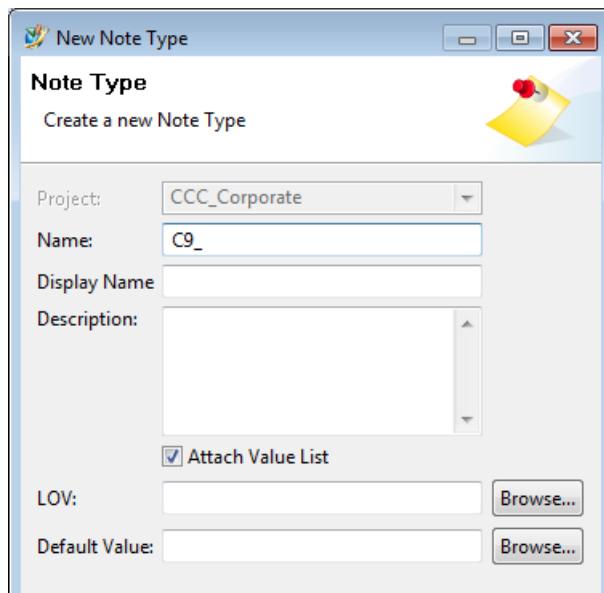


For example:

- A **Note Type** object stores an attribute on a Structure Manager BOM occurrence (component).
- A **Status** object designates the status of a business object in a workflow.
- A **Unit of Measure** object designates an item's units needed in a Structure Manager BOM.

8.2 Introduction to notes

A note type is an object associated with a product structure occurrence in a Structure Manager bill of materials (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 dialog boxes 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 manufacturing process management and for synchronizing object attributes from NX. These should not be deleted.

8.2.1 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. Select the file where you want to save the data model changes, for example, **options.xml**.
2. Expand the project and the **Options→List of Note Types** folders.
3. Right-click the **List of Note Types** folder and choose **New Note Type**.
The New Note Type wizard runs.
4. Perform the remaining steps in the **Note Type** dialog box.

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

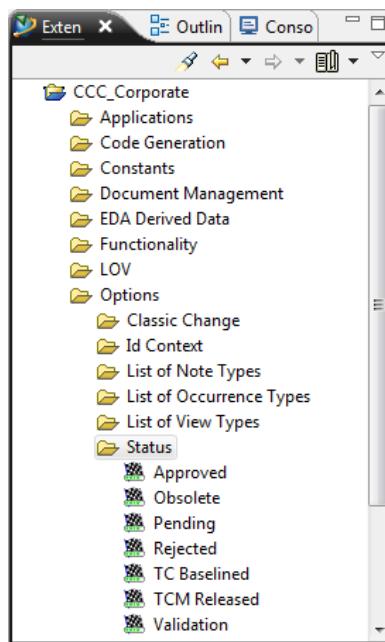
1. In the **Name** box, type the name you want to assign to the new note object.
2. In the **Display Name** box, type the name as you want it to appear in the user interface.
3. In the **Description** box, type a description of the new note object.
4. If you want to attach a value to the note from a list of values (LOV), select the **Attach Value List** check box **LOV** and **Default Value** boxes are displayed.
 - Click the **Browse** button to the right of the **LOV** box to locate the list of values to attach to the note. The LOV must be a **String** type because notes are string types.
 - 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.
5. 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** runtime business object. If you specified an LOV with the note, the new property automatically has the LOV attached to it.

8.3 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.

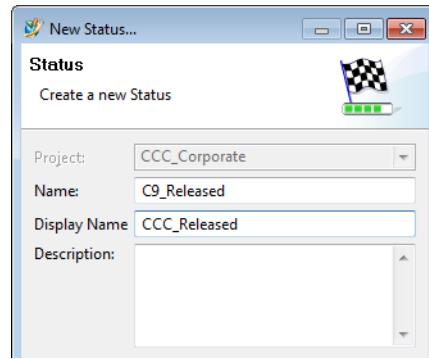


Key points

- After parts are released, Structure Manager can be used to display different bills of materials 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.
- To see the new status, use the **Modify Revision Rule** dialog box in Structure Manager.
 - In Structure Manager, choose **Tools**→**Revision Rule**→**Modify Current**.
 - In the **Modify Revision Rule** dialog box, click **Working** in the lower left corner and select **Status**.
 - Click the button to the right of the **Status** box.

8.3.1 Add a status

A status is applied to an object after it goes through a workflow. Typical statuses are *pending*, *approved*, *released* (in the following case, **CCC_Released** appears in the user interface).



1. Select the file where you want to save the data model changes, for example, **options.xml**.
2. Expand the project and the **Options→Status** folders.
3. Right-click the **Status** folder and choose **New Status**.
The New Status wizard runs.
4. 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 as you want it to appear in the database. The name cannot contain spaces. A prefix from the template is automatically affixed to the name.
 - c. In the **Display Name** box, type the name as you want it to appear in the user interface.
 - d. In the **Description** box, type a description of the new status object.
 - e. Click **Finish**.

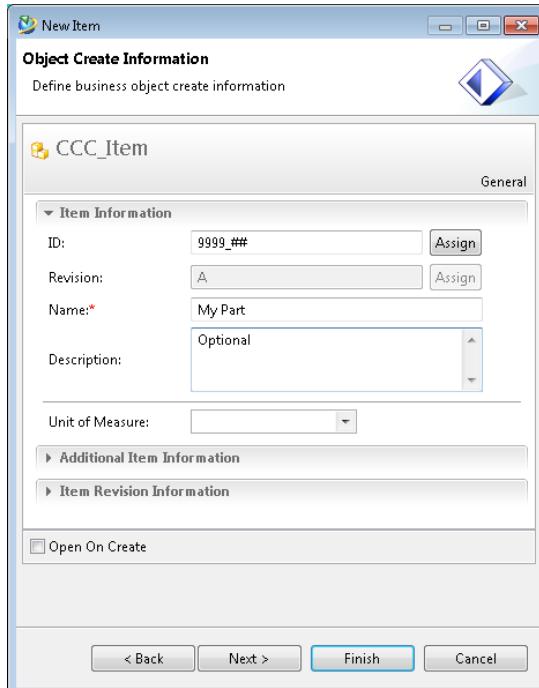
The new status object appears under the **Status** folder in the **Extensions** view.

8.4

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 revisions 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.

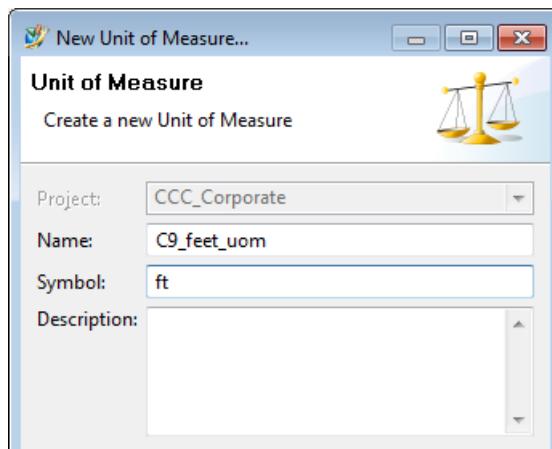
In **Structure Manager**, if no specific quantity value is associated with Items, the default quantity is each (one component). Also in Structure Manager, if UOM is anything other than null, the component does not open in NX.

8.4.1 Add a unit of measure

Create a units of measure (UOMs) when you need a new measurement for users.

1. Select the file where you want to save the data model changes, for example, **options.xml**.
2. Expand the project and the **Options→Unit of Measure** folders.
3. Right-click the **Unit of Measure** folder and choose **New Unit of Measure**.

The **New Unit of Measure** dialog box appears.



Define the unit of measure.

- 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. A prefix from the template is automatically affixed to the name.
- c. In the **Symbol** box, enter the unit (for example, **in** for inches, **oz** for ounces, and so on). 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.

8.5 Data model reports

Create reports by choosing **BMIDE→Reports**. You can create the following kinds of reports:

Compare Two Data Models

This report provides the data model differences between two data models. Often when working with the Business Modeler IDE, you want to compare the data model between two models. You can also generate this report using the **bmide_generate_compare_report** utility.

Condition Usage

This report provides the details of a condition and all model elements that use the condition. In the Business Modeler IDE, users can create conditions and attach these conditions to various business rules, LOVs, and so on, to define the behavior of Teamcenter.

Create Operation Override Report

This report provides the details of custom business objects where the create operations are overridden. In the Business Modeler IDE, you can override the create operations on the business objects.

Data Model

This report provides the details of a given category of model elements. Business Modeler IDE users can define a number of model elements and store them in a Business Modeler IDE template. These templates can be deployed to any Teamcenter database. You may be interested in generating a report all LOVs or GRM rules, or a report that shows a combination of multiple model element categories such as all deep copy rules and naming rules. You can also generate this report using the **bmide_generate_datamodel_report** utility.

Data Model Documentation

This report generates a multiple HTML-paged report of all data model elements for all templates in the data model. This report generates a comprehensive report of each element within the data model.

8.5.1 Compare Two Data Models report

Compare Two Data Models

This report provides the data model differences between two data models.

For example, you may be working in your Business Modeler IDE adding new elements to your template. Some time may have passed since your last deploy and you want to know the differences between the data model in your Business Modeler IDE and the data model that has been deployed to a Teamcenter database.

Another scenario is that as an administrator you have two Teamcenter database sites, and you are not certain if the data model is exactly the same in both sites or if there are differences. You may also want to see the differences between Business Modeler IDE template projects within your Business Modeler IDE client.

In all of these cases this report shows you the differences between the two data models.

This report gives you the ability to generate a report and choose the two data model input sources that are used to generate the report. The input may come from any of the following sources:

- Data model of a Business Modeler IDE template project
- Data model from a consolidated model file (**TC_DATA/model/model.xml**)
- Data model deployed to a Teamcenter database
- Business Modeler IDE template package

Note

You can also generate this report using the **bmide_generate_compare_report** utility.

8.5.2 Data Model report

Data Model

This report provides the details of a given category of model elements. Business Modeler IDE users can define a number of model elements and store them in a Business Modeler IDE template. These templates can be deployed to any Teamcenter database. You may be interested in generating a report all LOVs or GRM rules, or a report that shows a combination of multiple model element categories such as all deep copy rules and naming rules. A user can use the Business Modeler IDE client to examine all of this information; however, this report offers an easier means to examine all elements within a given category by generating this information into a single HTML page.

This report gives you the ability to generate a report like this and choose the data model input that is used to generate the report. The input may come from any of the following sources:

- Data model of a Business Modeler IDE template project
- Data model from a consolidated model file (*TC_DATA/model/model.xml* file)
- Data model deployed to a Teamcenter database
- Business Modeler IDE template package

Because all of the elements are generated into a single file, this report has limitations. As the number of categories and elements within a category grows, so does the resulting file size. In cases where you want to generate a report of multiple categories try using the **Data Model Documentation** report.

Note

You can also generate this report using the **bmide_generate_datamodel_report** utility.

8.5.3 Other Business Modeler IDE reports

Condition Usage

This report provides the details of a condition and all model elements that use the condition. In the Business Modeler IDE, users can create conditions and attach these conditions to various business rules, LOVs, and so on, to define the behavior of Teamcenter. This report generates a single HTML page report of the condition details and all of the model elements that refer to the condition. To generate this report you select a template project and the condition.

You can also generate this report using the **bmide_generate_condition_report** utility.

Create Operation Override Report

This report provides the details of custom business objects where the create operations are overridden. In the Business Modeler IDE, you can override the create operations on the business objects. This report generates a single page HTML report of all the custom objects that are subtypes of **Item**, **ItemRevision**, **Dataset**, **Form**, and **Relation** business objects that override the create operations, such as the **Create**, **SetPropertiesFromCreateInput**, **ValidateCreateInput**, **FinalizeCreateInput**, and **CreatePost** operations. Select a template project to run the report.

Data Model Documentation

This report generates a multiple HTML-paged report of all data model elements for all templates in the data model. This report generates a comprehensive report of each element within the data model. The report has links to view all elements within a category such as all LOVs, all naming rules, or all business objects. Additional links are provided so that you can view all of the elements for any Teamcenter template. A **What's New** section lists all the elements added, deleted, or changed since a previous release. This report is useful when you need a complete picture of the data model and interaction of various elements in the data model.

Due to the number of files generated for this report, report generation may take approximately 20 minutes.

8.5.4 Run the Compare Two Data Models report

1. Choose **BMIDE→Reports** and click **Next**.
2. Select **Compare Two Data Models** and click **Next**.
3. In the **Input Data Model 1** box, select the source of the first data model to compare and click **Next**:
 - **Template Project**
 - **Teamcenter Server**
 - **Template Package**
Browse for a template package.
 - **Model File**
4. Locate the first data model to compare. The dialog box that appears next is dependent upon the input source you selected:
 - **Template Project**
Select the project and templates to include in the report.
 - **Teamcenter Server**
 - **Template Package**
 - **Model File**
Select the **TC_DATA\model\model.xml** model file.
5. In the **Input Data Model 2** box, select the source of the second data model to compare and click **Next**.
6. Locate the second data model and click **Next**.
7. On the **Compare Two Data Model Options Page**, choose the value for **Show Equal Attributes** and click **Next**.
8. In the **Report File Selection** dialog box, select the folder location where to write the report, and type a name for the report in the **File Name** box.
9. Click **Finish**.

8.5.5 Run other Business Modeler IDE reports

Condition Usage report

1. Choose **BMIDE→Reports** and click **Next**.
2. Select **Condition Usage** and click **Next**.
3. In the **Condition Usage Report** dialog box, select the condition to report, and the project it resides in. Click **Next**.
4. In the **Report File Selection** dialog box, select the *project\output\reports* folder as the location where to write the report, and in the **File Name** box, type a name for the report.
5. Click **Finish**.

Create Operation Override report

1. Choose **BMIDE→Reports**.
2. Select **Create Operation Override Report** and click **Next**.
3. In the **Project** box, select the project on which you want to run the report and click **Next**.
4. In the **Report File Selection** dialog box, select the *project\output\reports* folder as the location where to write the report, and in the **File Name** box, type a name for the report.
5. Click **Finish**.

The report is generated at the specified location.

8.6 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

8.7 Summary

The following topics were taught in this lesson:

- Define and create a note type option.
- Define and create status type options.
- Define and create units of measure.
- Compare data model from two different sources with the Business Modeler IDE reports.

Lesson

9 Rule extensions

Purpose

The purpose of this lesson is to extend Teamcenter with a variety of rules that work with the Teamcenter business objects.

Objectives

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

- Create a naming rule (on item business object).
- Create a revision naming rule (on item revision business object).
- Modify a display rule to control who can create an item.
- Add deep copy rules to item revision save-as and revise operations.
- Create a condition on a item naming rule pattern.

Help topics

Additional information for this lesson can be found in:

- *Creating business rules*

9.1 Introduction to rules

The **Rules** folder in the **Extensions** view 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:

- Naming rules
- Revision naming rules
- Business object display rules
- Deep copy rules
- Generic Relationship Management (GRM) rules.

Rule evaluation

- 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.
- The first business rule found is executed.

Example

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

9.2 Naming rules introduction

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.

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, all business objects use the rule on that **item_id** property.

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**.

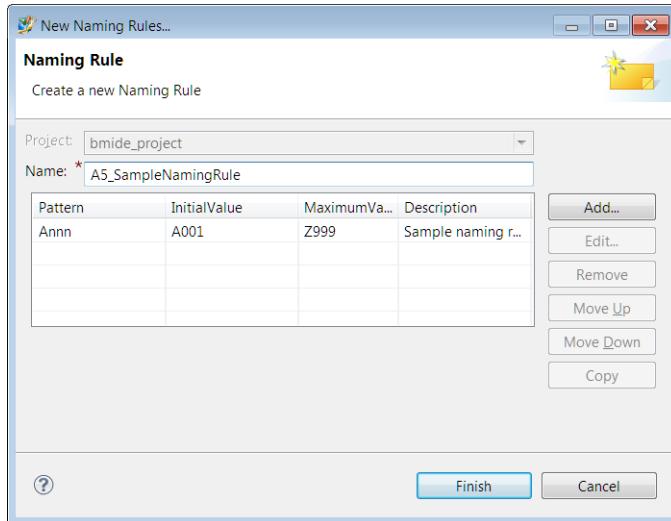
First enter a pattern of "**CCC**". Then enter a second pattern of **NNNNNN** (for numbers) with an initial value of **CCC000001** and a maximum value of **CCC999999**. Then attach it to the **item_id** property of the item business object you want.

The following table describes the properties to which naming rules can be attached.

| Business object | Property name |
|-----------------|-------------------------|
| Dataset | object_name |
| | pubr_object_id |
| | rev |
| Form | object_name |
| Identifier | idfr_id |
| Item | item_id |
| | object_name |
| ItemRevision | item_revision_id |
| PSOccurrence | occurrence_name |
| TC_Project | project_id |
| | project_name |
| TC_WorkContext | workcontext_name |

9.2.1 Creating naming rules

The **New Naming Rule** dialog box is used to define the naming rule patterns.



Creating a naming rule

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.

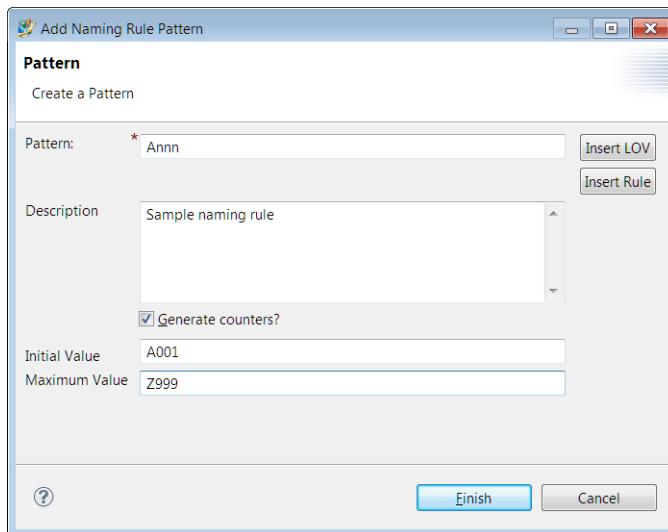
Each pattern can consist of combinations of the characters shown in the following table.

| 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.

9.2.2 Create and attach a naming rule



Creating a naming rule pattern

1. Set the active extension file to **rules.xml**.
2. Open the New Naming Rules wizard with one of these methods:
 - Choose **BMIDE→New Model Element**, type **Naming Rules** in the **Wizards** box, and click **Next**.
 - Open the **Extensions\Rules** folders, right-click the **Naming Rules** folder, 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. The name cannot contain spaces.
 - 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**.

9.2.3 Add a naming rule

You can create a naming rule for an item ID that starts with **CCC** plus a three digit number so that it generates numbers from **CCC001** to **CCC999**.

First enter a pattern of "**CCC**" . Then enter a second pattern of **nnn** (for numbers) with an initial value of **001** and a maximum value of **999**. Then attach it to the **item_id** property of the item business object you want.

1. Set the active extension file to **rules.xml**.
2. In the **Extensions** view, expand the project and the **Rules** folder.
3. Right-click the **Naming Rules** folder and choose **New Naming Rules**.
4. In the **Name** box, type the name for the new naming rule. *The name cannot contain spaces.*
5. Click the **Add** button in the **Patterns** pane, and enter the following information for each naming rule.
 - a. In the **Pattern** box, type "**CCC**"**nnn** for the pattern.
 - b. In the **Description** box, type a brief description.
 - c. Select the **Generate Counters** check box.
 - d. Type **CCC000** in the **Initial Value** box.
 - e. Type **CCC999** in the **Maximum Value** box.

Note

By default, the first pattern in the list is used to assign IDs for objects.

9.2.4 Attach a naming rule to a property

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 **Itemitem_id** property of the **Item** business object.

The following procedure describes how to 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. Set the active extension file to **rules.xml**.
2. Find the item business object and open it.
3. In the business object editor view, click the **Properties** tab.
4. In the **Properties** tab, locate and select the property.
5. Locate the **Naming Rule Attaches** section and click **Add**.
6. To the right of the **Naming Rule** box, click **Browse**.
7. Select the naming rule and click **OK**.

The **Naming Rule Attaches** section displays the naming rule.

9.2.5 Using system variables in patterns

If the pattern contains a system variable inside quotation marks, for example, "\${GROUP}-AD-"NNN, you can select the **Generate counters?** check box when creating the naming rule.

When generating counters for the "\${GROUP}-AD-"NNN example, the **Initial Value** and **Maximum Value** boxes can contain values similar to the following:

| | |
|----------------------|--------------------|
| Pattern | "\${GROUP}-AD-"NNN |
| Initial Value | \${GROUP}-AD-111 |
| Maximum Value | \${GROUP}-AD-999 |

Another similar example is:

| | |
|----------------------|----------------------|
| Pattern | "AA-\${GROUP}-BB-"NN |
| Initial Value | AA-\${GROUP}-BB-11 |
| Maximum Value | AA-\${GROUP}-BB-99 |

The following values are valid for system variable:

| | |
|--------------------------|---|
| GROUP | User's current group. |
| ROLE | User's current role. |
| USERID | User's ID. |
| USERNAME | User's name. |
| SITENAME | Site name. |
| All:<i>pref</i> | Specifies a preference variable. |
| GROUP:<i>pref</i> | Specifies a group protection scope preference variable. |
| ROLE:<i>pref</i> | Specifies a role protection scope preference variable. |
| SITE:<i>pref</i> | Specifies a site protection scope preference variable. |
| USER:<i>pref</i> | Specifies a user protection scope preference. |

9.2.6 Other naming rules pattern control

Using counters with naming rules

The first pattern in the naming rule is used with counters. If there are more patterns, the client prompts you to choose the pattern you want to use.

Any number of counters can be used in a pattern, for example, **nnn"."a**. With counters activated for this pattern, the **Assign** button allocates IDs as follows:

```
000.a
000.b
000.c
:
000.z
```

The right-side counter range completes first and then moves to the next range from the right, as follows:

```
001.a
:
001.z
002.a
:
002.z
003.a
:
003.z
```

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.

9.2.7 Revision naming rule

A *revision naming rule* is a business rule that defines the naming convention and sequence for a revision property. This functionality is required for the strict naming requirements needed for some industries. To put the naming rule into effect, you must attach it to a property on a business object.

- **Name** is the name you want to assign to the new naming rule. The name cannot contain spaces.
- **Exclude I, O, Q, S, X, Z ?** is used to exclude these characters from use in the rule. The characters are set in the **TcRevisionSkipLetters** LOV.
- **Initial Revision** information
 - **Initial Revision Type** is the revision name scheme: **Alphabetic** (only alphabet characters), **Numeric** (only numbers), or **Alphanumeric** (a mix of letters and numbers).
 - **Initial Revision Start** is the characters to start the revision naming scheme.
 - **Initial Revision Description** is a description of the first portion of the revision naming scheme.
- **Secondary Revision** information
 - **Secondary Revision Type** is the revision name scheme: **Alphabetic** (only alphabet characters), **Numeric** (only numbers), or **Alphanumeric** (a mix of letters and numbers).
 - **Secondary Revision Start** is the characters to continue the revision numbering.
 - **Secondary Revision Description** is the description of the next portion of the revision naming scheme.

9.2.8 Supplemental revision types

Allowable supplemental revision types:

NumericNoZeroFill

Assigns numbers with no zeroes, for example, 1, 2, 3, up to 9999.

FixedTwoDigitsZeroFill

Assigns two-digit numbers using a zero fill-in, for example, 01, 02, 03, up to 99.

FixedThreeDigitsZeroFill

Assigns three-digit numbers using a zero fill-in, for example, 001, 002, 003, up to 999.

FixedFourDigitsZeroFill

Assigns four-digit numbers with a zero fill-in, for example, 0001, 0002, 0003, up to 9999.

CurrentRevLetterNumericZeroFill

Assigns numbers to the current revision letter with zeroes, for example, A1, A2, A3, up to A99.

CurrentRevLetterFixedOneDigit

Assigns single-digit numbers to the current revision letter, for example, A1, A2, A3, up to A9.

CurrentRevLetterFixedTwoDigitsZeroFill

Assigns two-digit numbers to the current revision letter, for example, A01, A02, A03, up to A99.

NextRevLetterNumericNoZeroFill

Assigns numbers to the next revision letter, for example, B1, B2, B3, up to B99.

NextRevLetterNumericFixedOneDigit

Assigns single-digit numbers to the next revision letter, for example, B1, B2, B3, up to B9.

NextRevLetterFixedTwoDigitsZeroFill

Assigns two-digit numbers to the next revision letter, for example, B01, B02, B03, up to B99.

9.2.9 Add a revision naming rule

1. Set the active extension file to **rules.xml**.
2. In the **Extensions** view, expand the project and the **Rules** folder.
3. Right-click the **Revision Naming Rules** folder and choose **New Revision Naming Rules**.
4. Type the **Name** for the new naming rule. *The name cannot contain spaces.*
5. Select the **Exclude I, O, Q, S, X, Z ?** check box to exclude these characters from use in the rule. The characters are set in the **TcRevisionSkipLetters** LOV.
6. Select the following information for the **Initial Revision**.
 - a. Click the arrow in the **Initial Revision Type** box to choose the revision name scheme: **Alphabetic** (only alphabet characters), **Numeric** (only numbers), or **Alphanumeric** (a mix of letters and numbers).
 - b. In the **Initial Revision Start** box, type the characters to start the revision naming scheme.
 - c. In the **Initial Revision Description** box, type a description of the first portion of the revision naming scheme.
7. Click **Finish**.

The naming rule is added to the **Naming Rules** folder.

Secondary Revision and **Supplemental Revision Type** settings:

Note

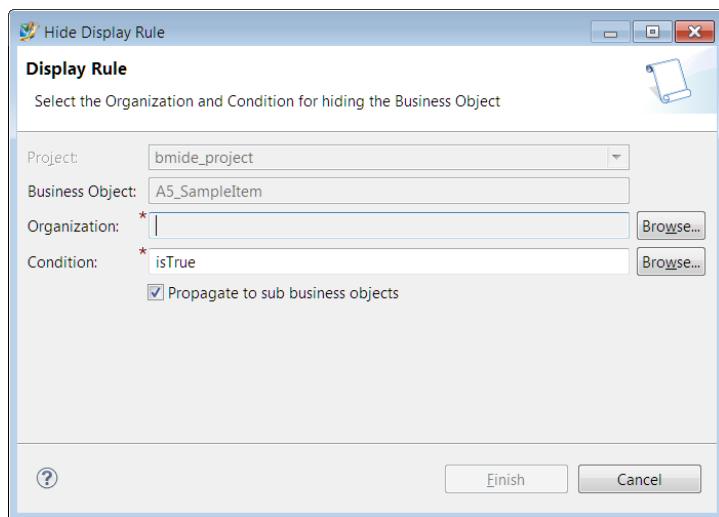
The **Secondary Revision Type** revision naming scheme must be a different type than the **Initial Revision Type**.

- Optionally select the information for **Secondary Revision**. Essentially repeat as with the **Initial Revision**.
- Optionally select the information for **Supplemental Revision Type**.

9.3

Working with business object display rules

Business object display rules determine the members of the organization who cannot view a business object type in menus in the Teamcenter user interface. The **Display Rules** editor displays the groups and roles that are not allowed to see the selected type of business object in menus. This rule is primarily used to hide business objects from creation (**File→New**) menus, thereby restricting those who can create the business object type.



Hide Display Rule wizard

Business object display rules are subject to the principles of group hierarchy and inheritance and inheritance. By default, all business objects are available for creation by all users.

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

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

Note

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

9.3.1 Add a business object display rule

The **Display Rules** editor displays the groups and roles that can view a business object in the create menus in the Teamcenter user interface.

1. Select the file where you want to save the data model changes, for example, **rules.xml**.
2. In the **Business Objects** view, right-click the relevant business object or one of its children, choose **Open**, and click the **Display Rules** tab.
3. Click the **Add** button to the right of the **Hide Business Object Rules** table.
4. Perform the remaining steps in the **Display Rule** dialog box:

Perform the following steps in the **Display Rule** dialog box:

1. Click the **Browse** button to the right of the **Organization** box.

The Teamcenter Repository Connection wizard displays the groups and roles from the server 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.

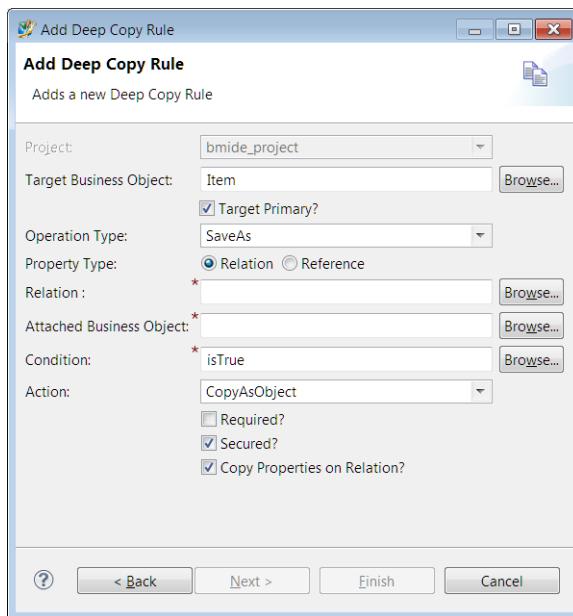
2. Select the **Propagate to sub Business Objects** check box if all children of the business object inherit the display rule.
3. As needed, click the **Browse** button to the right of the **Condition** box to select the condition for which this display rule runs. If you select **isTrue** as the condition, the rule always applies.
4. 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.

9.4

Deep copy rules

Deep copy rules define whether objects belonging to a business object instance can be copied when a user performs a save as or revise operation on that instance. Deep copy rules can be applied to any business object type and are inherited by children business object types. Prior to Teamcenter 10, deep copy rules could only be applied to item revision business objects.



Deep Copy Rule wizard

Possible actions include: CopyAsObject, CopyAsReference, CopyAsReferenceOrReferenceNewCopy, NoCopy, RelateToLatest.

Key points

- Existing deep copy rules should be tested to validate that the behavior meets company procedures.
- New deep copy rules should be tested to be sure it performs as expected.
- 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.
- Although deep copy rules defined at a parent business object are inherited by the children, they are not editable on the children. The **Edit** and **Remove** buttons are unavailable on the **Deep Copy Rules** tab for the children business objects.

9.4.1 Deep copy rule definitions

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.

CopyAsReferenceOrReferenceNewCopy

Relates the current primary object to the original secondary object if the secondary object has not been copied or revised during the current operation. If a new copy of the secondary object is produced during the current operation, it relates the current primary object to the new copy of the secondary object. This action is available for the save as operation only.

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.

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.

Additional actions

Select

Indicates that the object to be copied is to be selected by the client. To use this option, you must create a custom user interface or a server-side customization.

SystemCopy

Indicates that the copy is to be performed by the system without any end user input. This is similar to the **CopyAsObject** action but without presenting a dialog box to the user.

9.4.2 Deep copy examples

This example shows the effect of different deep copy rules for the **Revise** operation.

Revision structure *before Revise*

| Object | Relation |
|---|--|
| 001 MyPart Revision 001FileABCD 001FileMNOP | IMAN_reference IMAN_specification |

Objects attached to the item revision by the **IMAN_reference** relation can only be subject to **IMAN_referenceCopyAsReference** or **NoCopy** rules. Reference attachments cannot be copied forward as new objects.

Results after revise with CopyAsObject

| Object | Relation |
|---|--|
| 002 MyPart Revision 001FileABCD 002FileMNOP | IMAN_reference IMAN_specification |

Results after revise with NoCopy

| Object | Relation |
|---|--|
| 002 MyPart Revision 001FileABCD 001FileMNOP | IMAN_reference IMAN_reference |

Results after revise with CopyAsReference

| Object | Relation |
|---------------------|----------|
| 002 MyPart Revision | |

9.4.3 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.

Perform these initial steps to create a deep copy rule on an item revision business object:

1. Select the file where you want to save the data model changes, for example, **rules.xml**.
2. In the **Business Objects** view, right-click the **ItemRevision** business object or one of its children, choose **Open**, and click the **Deep Copy Rules** tab. Deep copy rules are only allowed for item revisions or children of item revisions.

The **Deep Copy Rules** editor displays the active rules on the item revision business object.

3. 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.

4. Click the **Add** button.

The Add Deep Copy Rule wizard runs.

9.4.4 Add a deep copy rule (continued)

Provide the following information in the **Add Deep Copy Rules** dialog box:

- **Target Business Object**

The business object for the deep copy rule, for example, **ItemRevision** or one of its children.

- **Target Primary?**

- o When selected, the **Target Business Object** is the *primary* object of the relationship specified in the **Relation Type** box, and the *secondary* objects are carried forward and related via the relation in the **Relation Type** box.
 - o When clear, the **Target Business Object** is the *secondary* object of the relationship specified in the **Relation Type** box, and the *primary* objects are carried forward and related via the relation in the **Relation Type** box.

- **Operation Type**

Select **Save As** or **Revise**.

- **Relation Type**

- o **Relation**

Available relationships are children of the **ImanRelation** business object. Objects with a relationship that matches the selected relationship are only copied from the source business object instance to the destination business object instance. To match all relationships, select **Match All**.

- o **Reference**

Define a deep copy rule between an object and a referenced object through the reference property. For example, the **Item** business object has a **uom_tag** reference property that is a typed reference property to the **UnitOfMeasure** business object. When configuring a deep copy rule for an **Item** business object, if the reference property is selected, you can configure the deep copy rule on the **uom_tag** property and the **UnitOfMeasure** or any of its subtypes as the secondary object.

- **Attached Business Object**

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.

- **Condition**

The condition for which this deep copy rule runs. If you select **isTrue** as the condition, the deep copy rule always applies.

- **Action**

The type of copying to be allowed for the business object:

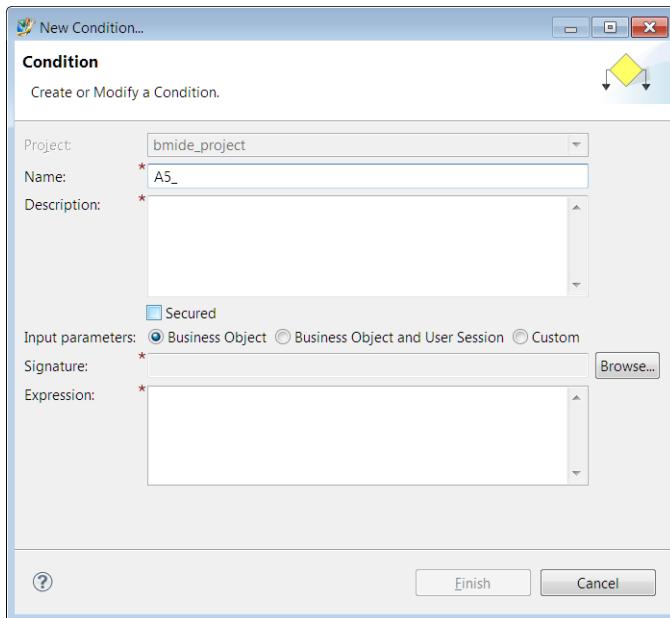
- o **CopyAsObject**
- o **CopyAsReference**
- o **CopyAsReferenceOrReferenceNewCopy**
- o **NoCopy**
- o **RelateToLatest** (Revise only)
- o **ReviseAndRelateToLatest** (Revise only)
- o **Select**
- o **SystemCopy**

- **Copy Properties on Relation**

- o 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.
- o If it is not selected, only the mandatory properties are carried forward.

9.5 Conditions introduction

Conditions are conditional statements that resolve to true or false based on the evaluation of an expression. Conditions can be used to evaluate objects or user sessions to deliver only certain results. Conditions are never used by themselves but are only used by other objects such as rules, LOVs, or IRDCs. The condition engine utilizes the CLIPS (C Language Integrated Production System) external rules engine to process condition data.



When you write a condition, you first choose the business objects against which you want to run the condition (the signature) and then you write the statement to evaluate the business objects (the expression).

Once you have created the condition, it can be used by other objects. Conditions are never used by themselves, but are only used by other objects. There are many kinds of objects that can use conditions, such as business object display rules, deep copy rules, LOVs, and naming rules, among others.

9.5.1 Condition example

Suppose you want to write a condition to evaluate item revision business objects for a particular group of users. Select **Business Object and User Session** and then browse for the **ItemRevision** business object. Notice that in the **Signature** box an **o** parameter is assigned to the business object, and a **u** parameter is assigned to the user session:

```
MyCondition ( ItemRevision o , UserSession u)
```

In the **Expression** box (the condition's evaluation formula), you use the **o** parameter to represent the business object and the **u** parameter to represent the user session.

If you want to evaluate the **ItemRevision (o)** name to be **Wheel** and the **UserSession (u)** group to be **Engineering**, you would write this expression.

```
o.object_name="Wheel" AND u.group_name="Engineering"
```

9.5.2 Conditions overview

Conditions are conditional statements that resolve to true or false based on the evaluation of an expression. A condition resolves to **TRUE** if the statement is valid or **FALSE** if it is not. Rules use conditions to describe the types of objects to which the rules apply.

When a rule that uses a condition is run against an object, it is divided into two parts, an **IF** clause and a **THEN** clause. The condition (**IF** clause) examines the object with Boolean logic, and the rule (**THEN** clause) describes an action or access permission on the object.

There are many types of rules that can run conditions, such as deep copy rules. For example, you can create a deep copy rule that states if (condition) a document business object has a specification relationship to an item revision, then (rule) that object is copied forward with the revised item revision.

The syntax for a condition is as follows:

```
condition-name(argument-list) :=  
    expression
```

In this syntax, everything to the left of the `:=` symbol is the signature, and everything to the right is the actual condition. (The `:=` symbol does not appear on the **Condition** dialog box when you create a condition.) Replace *condition-name* with the name of the condition, replace *argument-list* with the list of objects to be supplied by the calling program, and replace *expression* with the condition statement.

Following is an example condition:

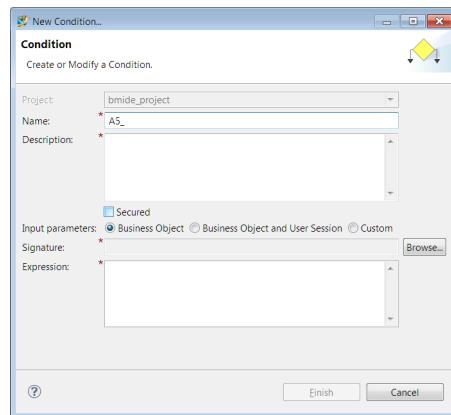
```
MyCondition( ItemRevision o) :=  
    o.color = red
```

In the signature (to the left of the `:=` symbol), **MyCondition** is the name of the condition and **ItemRevision** is the type of business object to run against. In the expression (to the right of the `:=` symbol), **o.color** is the **color** attribute on the business object, and **red** is the condition to be met. This condition says to examine the **color** attribute on all of the **ItemRevision** business objects and find those that have the **red** value.

You have a great deal of flexibility when creating the expression of a condition. You can write expressions that contain strings, logical statements, dates, tags, and even other conditions.

9.5.3 Add a condition

Defining a condition involves providing a name and description, choosing a signature, and writing the expression.



1. Start the **New Condition** wizard with one of these methods:
 - Choose **BMIDE→New Model Element**.
 - Expand **Extensions\Rules** and right-click the **Conditions** folder.
2. Type the values for **Name** and **Description**.
3. Select **Secured** if you want to prevent the condition from being modified or overridden by another template.
4. For **Input parameters**, select one of the following:
 - **Business Object**
 - **Business Object and User Session**
 - **Custom**
5. Choose a signature (with the **Browse** button). The signature depends on the **Input parameters** setting.
 - For a **Business Object** input parameter setting, select a business object.
 - For a **Business Object and User Session** input parameter setting, select a business object.
 - For a **Custom** input parameter setting, use the wizard to define a condition with two parameters where one of the parameters is not a **UserSession** business object or to define a condition with three or more parameters.

9.5.4 Condition input parameters and signature

Input parameters

Under **Input parameters**, select one of the following:

- Select the **Business Object** input parameter if you want the condition to be applied to a business object. You can also select this option if you want to use only the **UserSession** business object as a parameter.
- Select the **Business Object and User Session** input parameter if you want the condition to be applied to a business object in the context of a user's work session (using the **UserSession** object).
- Select the **Custom** input parameter if you want to define a condition with two parameters where one of the parameters is not a **UserSession** business object, or to define a condition with three or more parameters.

Signature

Click the **Browse** button to the right of the **Signature** box to select the business object for the condition.

- **Business Object**

Example

```
MyCondition (UserSession o)
```

- **Business Object and User Session**

Example

```
MyCondition ( ItemRevision o , UserSession u)
```

- **Custom**

The Custom Parameters wizard is used to define a condition with two parameters where one of the parameters is not a **UserSession** business object or to define a condition with three or more parameters.

In the **Parameter Name** box, type the name of the parameter to assign to the business object. You can type any character string for the parameter name, as long as it does not have spaces. Earlier you saw how by default the **o** parameter was chosen by the system to represent a business object, and **u** to represent a user session.

9.5.5 Condition operators

Binary conditions are constructed of expressions on the left side of the condition (left operand) compared to expressions on the right side (right operand). Binary conditions follow this syntax:

expression-value comparison-operator expression-value

The following Boolean operators are used to compare the left operand to the right operand.

| Operator | Description |
|--------------|---|
| ! | Not true (when used with a stand-alone operand) |
| != | Not equal to |
| < | Less than |
| <= | Less than or equal to |
| = | Equal to |
| > | Greater than |
| >= | Greater than or equal to |
| AND | Both operands are evaluated to true |
| NOT | The following operand is not true. |
| OR | Either operand is evaluated to true |

Note

Mathematical operators like **+**, **-**, **/**, *****, and **%** are not supported, nor are trigonometric functions like **sin**, **cos**, **tan**, and so on.

9.5.6 Condition examples

Examples provide a good way to understand how to structure conditions. The examples below are for discussion purposes only.

- Primitives

- String

```
isStringCondition1( Item o ) := Expression: "EqualString" =
"EqualString"
isStringCondition2( Item o ) := "UnequalString1" !=
"UnequalString2"
```

- Character

```
isCharCondition1( Item o ) := 'a' = 'a'
isCharCondition2( Item o ) := 'a' != 'b'
isCharCondition3( Item o ) := 'a' < 'b'
isCharCondition4( Item o ) := 'a' <= 'a'
isCharCondition5( Item o ) := 'b' > 'a'
isCharCondition6( Item o ) := 'a' >= 'a'
```

- Integer

```
isIntCondition1( Item o ) := 1 = 1
isIntCondition2( Item o ) := 0 != 1
isIntCondition3( Item o ) := 0 < 1
```

- Float

```
isFloatCondition1( Item o ) := 1.5 = 1.5
isFloatCondition2( Item o ) := 0.5 != 1.5
isFloatCondition3( Item o ) := 0.5 < 1.5
```

- Boolean

```
isBoolCondition1( Item o ) := true
isBoolCondition2( Item o ) := false
```

- Date

```
isDateCondition1( Item o ) := o.creation_date = "01-Jan-2009
00:00"
isDateCondition2( Item o ) := o.creation_date != "29-Feb-2008
23:59"
isDateCondition3( Item o ) := o.creation_date < "01-Jan-2009
00:00"
```

- Properties on business objects

- Properties on any business object

```

isPropCondition1( MyItem o ) := o.color = "red"
isPropCondition2( MyItem m, YourItem y ) := m.color = y.color
isPropCondition3( MyItem m, YourItem y ) :=
m.owning_project.project_id = y.owning_project.project_id

```

- o Properties on any business object when there are spaces in the property name

```

isCondition4( BOMLine bl1, BOMLine bl2 ) := bl1.`UG NAME` =
bl2.`UG NAME`

```

- o Properties on UserSession

```

isPropCondition5( MyItem o, UserSession u ) := o.owning_user =
u.user
isPropCondition6( MyItem o, UserSession u ) := o.owning_group =
u.group
isPropCondition7( MyItem o, UserSession u ) := o.owning_project =
u.project
isPropCondition8( MyItem o, UserSession u ) := u.user_id =
"infodba"
isPropCondition9( MyItem o, UserSession u ) := u.group_name =
"Engineering"
isPropCondition10( MyItem o, UserSession u ) := u.project_name =
"Concept"
isPropCondition11( MyItem o, UserSession u ) := u.role_name =
"DBA"

```

- Operations on business objects

```

isOperationCondition1( DeepCopyRule dr,
ItemRevision target, ItemRevision otherside ) :=
otherSide.items_tag.isLatestRevisionMature() = true
isOperationCondition2( DeepCopyRule dr, ItemRevision target,
POM_object otherside ) := otherSide.isReplica() = true
isOperationCondition3( DeepCopyRule dr, ItemRevision target,
ItemRevision otherside ) :=
target.checkUniqueItems(
otherSide,dr.relation,dr.is_target_primary,1,-1 ) = true

```

- Nested conditions

```

isNestCondition1( DeepCopyRule dr, ItemRevision target,
ItemRevision otherside ) :=
( Condition::checkOtherSideOneToOne( dr, target,otherside ) = true
) AND
( Condition::isOtherSideLatestMature( dr, target, otherside ) =
true )

```

Note

All nested conditions need to be prefixed with **Condition::**.

- **INLIST**

- o Search for a tag in an array of typed/untyped references

```
isNextCondition1( WorkspaceObject o, UserSession u ) :=
Function::INLIST( u.project, o.project_list )
```

Note

All **INLIST** conditions need to be prefixed with **Function::**.

- o Search for a primitive in an array of typed/untyped references

```
isNestCondition2( WorkspaceObject o, UserSession u ) :=
Function::INLIST( u.project_name, o.project_list, "project_name"
)
```

- String functions

- o **ToUpper**

```
isToUpperCondition1( Item o, UserSession u ) :=
Function::ToUpper( u.role_name ) = "DBA"
```

Note

All **ToUpper** conditions need to be prefixed with **Function::**.

- o **ToLower**

```
isToLowerCondition1( Item o, UserSession u ) :=
Function::ToLower( u.user_id ) = "mgr"
```

Note

All **ToLower** conditions need to be prefixed with **Function::**.

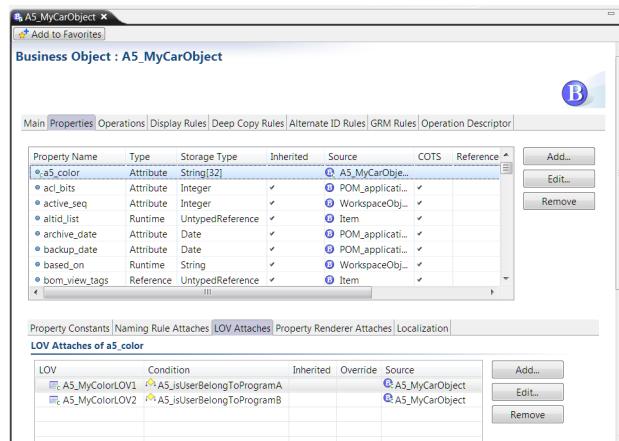
- Operators

Operators include **!**, **!=**, **<**, **<=**, **=**, **>**, **>=**, **AND**, **NOT**, and **OR**.

9.6 Attach LOVs with conditions

You can apply the conditions when attaching the LOVs to properties. In this example, you want to display different LOV values based on whether a user logs on to Teamcenter as a member of certain programs.

If the end user logs on as a member of the **Program A** program, you want to show the **Green** and **Red** values for the **color** property on the **MyCarObject** business object. But if the user logs on as a member of the **Program B** program, you want to show the **Gold** and **Silver** values on the same property.



1. Create the classic LOVs.
 - a. Create a **MyColorLOV1** LOV and add **Green** and **Red** values.
 - b. Create a **MyColorLOV2** condition with **Gold** and **Silver** values.
2. Create the conditions.
 - a. Create an **isUserBelongToProgramA** condition with an **isUserBelongToProgramA (UserSession o)** signature and an **o.project_name = "Program A"** expression.
 - b. Create an **isUserBelongToProgramB** condition with an **isUserBelongToProgramB (UserSession o)** signature and an **o.project_name = "Program B"** expression .
3. Attach the LOVs.
 - a. Attach the **MyColorLOV1** LOV to the **color** property on the **MyCarObject** business object with the **isUserBelongToProgramA** condition.
 - b. Attach the **MyColorLOV2** LOV to the **color** property on the **MyCarObject** business object with the **isUserBelongToProgramB** condition.
4. Verify the behavior.

9.7 Project LOV use case

In this use case, if an item is assigned to project A, the **Color** property shows one set of colors to choose from in the LOV; if the item is assigned to project B, the **Color** property shows another set of colors.

1. Create a custom business object and property.
 - a. Create a child of the **Item** business object: **B5_MyPart**.
 - b. Add a property **b5_MyColors** to designate the object's colors.
2. Create LOVs.
 - a. Create LOV **B5_Colors1**, values **Red** and **Blue**.
 - b. Create another LOV **B5_Colors2**, values **Green** and **Yellow**.
3. Create project-based conditions.
 - **B5_isProjectA**

This condition is evaluated to the **True** value when working with an object in Teamcenter whose **project_list** property has a project named **ProjectA**.

Signature:

```
B5_isProjectA ( WorkspaceObject o , UserSession u)
```

Expression:

```
(o!=null AND Function::INLIST("ProjectA", o.project_list, "project_name"))
OR (o=null AND u.project_name="ProjectA")
```

This expression says that if the workspace object exists (**o!=null**), and the **ProjectA** project exists, the condition is true.

- **B5_isProjectB**

This condition is evaluated to the **True** value when working with an object in Teamcenter whose **project_list** property has a project named **ProjectB**.

Signature:

```
B5_isProjectB ( WorkspaceObject o , UserSession u)
```

Expression:

```
(o!=null AND Function::INLIST("ProjectB", o.project_list, "project_name"))
OR (o=null AND u.project_name="ProjectB")
```

This expression says that if the workspace object exists (**o!=null**), and the **ProjectB** project exists, the condition is true.

4. Attach the LOVs to the property

- Attach **B5_Colors1** LOV to the **b5_MyColors** property and specify the condition as **B5_isProjectA**.

This means that when a **B5_MyPart** object is assigned to the **ProjectA** project, the **b5_MyColors** property displays the values on the **B5_Colors1** LOV.

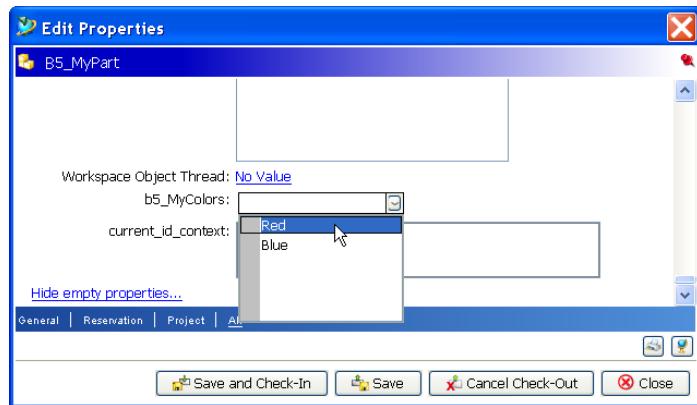
- Attach **B5_Colors2** LOV to the **b5_MyColors** property and specify the condition as **B5_isProjectB**.

This means that when a **B5_MyPart** object is assigned to the **ProjectB** project, the **b5_MyColors** property displays the values on the **B5_Colors2** LOV.

5. Test in the rich client.

- Deploy the template from the Business Modeler IDE to the server.
- Create a project whose name is **ProjectA** and another whose name is **ProjectB**.
- Create an instance of the **B5_MyPart** business object.
- Set the project on the business object instance to **ProjectA** by right-clicking the object and choosing **Project→Assign**.

On the **b5_MyColors** property, the available values are **Red** and **Blue**.



9.8 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

9.9 Summary

The following topics were taught in this lesson:

- Create a naming rule (on item business object).
- Create a revision naming rule (on item revision business object).
- Modify a display rule to control who can create an item.
- Add deep copy rules to item revision save-as and revise operations.
- Create a condition on a item naming rule pattern.

Lesson

10 Data model live updates

Purpose

The purpose of this lesson is to describe Teamcenter data model live updates.

Objectives

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

- Understand the live update process.
- Enable a template for live updates.
- Create a live update project.
- Deploy a live update project.

Help topics

Additional information for this lesson can be found in:

- *Working with live updates*

10.1 Introduction to live updates

Live updates is the revision on a live running system of nonschema data such as lists of values (LOVs) that requires frequent update. The live updates functionality in the Business Modeler IDE allows an administrator to revise data in the production database without shutting down the production server. It also provides tighter control on which elements get updated live in a production database.

This table lists elements that can be used in live updates. To select the elements to allow for live updates, use the **Live Update** preference.

| Data elements that can be updated live (short list) | |
|--|--|
| Display name (Internal name) | Description |
| Condition (Condition) | Defines conditional statements that resolve to true or false based on the evaluation of an expression. |
| Deep Copy Rule (TcDeepCopyRule) | Defines whether relational type objects can be copied as an object, copied as a reference, or not copied when the user performs a save as or revise operation. |
| Display Rule (TcTypeDisplayRule) | Determines the members of the organization who can view a business object in the create menus in the Teamcenter user interface. |
| GRM Rule (TcGRMRule) | Limits which objects can be pasted to other objects. Generic Relationship Management rules (GRMs) are defined using relationships. |
| Localization (Localization) | Defines localized display names for business objects, properties, legacy change, view, status, LOV values and descriptions, ID context, occurrence type, and note. |
| LOV (TcLOV) | Defines pick lists of values accessed by end users from a menu at the end of a data field. Lists of values (LOVs) ensure consistent data entries in the rich client. |
| Naming Rule (TcNamingRule) | Defines the naming conventions for the string property value in different type objects. |

| Data elements that can be updated live (short list) | |
|--|---|
| Display name (Internal name) | Description |
| Naming Rule Attachments (TcNamingRuleAttach) | Attaches the naming rule or revision rule to a property on a business object so the rule can take effect. |
| Note Type (TcNoteType) | Defines a note object associated with a product structure occurrence in a Structure Manager bill of materials (BOM). |
| Property Constant (TcPropertyConstant) | Provides default values to business object properties. Because these constants are attached to properties, they are inherited and can be overridden in the hierarchy. |
| Revision Naming Rule (TcRevNamingRule) | Defines the naming convention and sequence for a revision property. |
| Status (TcStatus) | Applies status to an object after it goes through a workflow. Typical statuses are Pending and Approved . |
| Tool (TcTool) | Represents a software application, such as Microsoft Word or Adobe Acrobat. Associate a tool with a type of dataset so you can launch the dataset file from Teamcenter. |
| Unit of Measure (TcUnitOfMeasure) | Defines a measurement category (for example, inches, millimeters, and so on). Create a unit of measure (UOM) when you need a new measurement for users. |

10.2

Data elements that can be updated live

This table lists elements that can be used in live updates. To select the elements to allow for live updates, use the **Live Update** preference.

| Data elements that can be updated live (others list) | |
|---|---|
| Display name (Internal name) | Description |
| Alias ID Rule (TcAliasIdRule) | Stores part numbers and other attribute information for similar parts. |
| Alternate ID Rule (TcAlternateIdRule) | Stores information about part numbers and attributes of the same part from different perspectives. They allow different user communities to identify and display an item or item revision according to their own rules rather than by the rules of the user who created the object. |
| Application Extension Point (TcAppExtensionPoint) | Allows for the configuration of applications using a decision table. This extension point defines the table, inputs, and outputs that customers can configure against it. |
| Application Extension Rule (TcAppExtensionRulev) | Determines when an application extension point is used, and defines inputs and outputs. |
| Business Context (TcBusinessContext) | Defines the user groups for whom a rule applies. |
| Business Object Constant (TcTypeConstant) | Provides default values to business objects. Because these constants are attached to business objects, they are inherited and can be overridden in the hierarchy. |
| Business Object Constant Value Override (TcTypeConstantAttach) | Provides default values to business objects. Because these constants are attached to business objects, they are inherited and can be overridden in the hierarchy. An override changes the value that you set in your template. |
| Change (TcChange) | Represents an alteration to requirements. |
| Dispatcher Service Config (DispatcherServiceConfig) | Defines the visualization file format that a dataset file is translated into. |

| Data elements that can be updated live (others list) | |
|--|---|
| Display name (Internal name) | Description |
| EDA Derived Data (EDADerivedDataConfig) | Configures information that is derived from an ECAD design such as derived items and datasets. For example, a schematic drawing can be automatically generated from the schematic design and saved along with the schematic item. |
| Global Constant (TcGlobalConstant) | Provides consistent definitions that can be used throughout the system. These constants have only one value, either the default value or the value you set. |
| Global Constant Value Override (TcGlobalConstantAttach) | Provides consistent definitions that can be used throughout the system. These constants have only one value, either the default value or the value you set. An override changes the value that you set in your template. |
| ID Context (TcIdContext) | Defines when you use unique item IDs. ID contexts are used when you create alias IDs or alternate IDs. |
| IRDC (IRDC) | Defines how item revisions are handled. IRDCs standardize item revision behavior at specific times in the life cycle, such as during item creation, checkin, checkout, save as, and revise. |
| Occurrence Type (TcPSOccurrenceType) | Determines how items occur in a product structure. An occurrence consists of one component in an assembly including its relative position with respect to its parent assembly. Occurrence types are representations of the PSOccurrence business object. |
| Print Configuration (PrintConfiguration) | Defines batch print settings. When you batch print an object such as an item revision, all the documents associated with that object are printed. |

| Data elements that can be updated live (others list) | |
|--|--|
| Display name (Internal name) | Description |
| Property Constant Value Override (TcPropertyConstantAttach) | Provides default values to business object properties. Because these constants are attached to properties, they are inherited and can be overridden in the hierarchy. An override changes the value that you set in your template. |
| Storage Media (TcStorageMedia) | Defines a storage device category such as a hard disk or optical device. It is used by third-party content-storage systems. |
| System Stamp Configuration (SystemStampConfiguration) | Defines the system stamp (such as date or watermarks) on documents in batch printing. |
| Teamcenter Component (Teamcenter Component) | Specifies objects that can be used to create categories of model data. For example, Teamcenter Component objects are used to create categories for verification rules. |
| Verification Rule (VerificationRule) | Configures conditions to be used with specific applications. |
| View Type (TcViewType) | Controls the name of a product structure view object. A view type is a BOM view revision (BVR) category. The product structure view types work with the item and item revision business objects to maintain product structure information. |

10.3 Working with live updates

Use one of the following processes to update live data:

- Single administrator

Use this process if you are a single administrator who makes live updates and distributes them to servers. In this process, make your updates on a preproduction server before sending the updates to production servers. This process is recommended in most situations.

- Multiple administrators

Use this process if you have multiple administrators who make live updates and distribute them to servers. In this process, there is no preproduction server.

Caution

Siemens PLM Software recommends using a single administrator if possible to better control the live updates distributed to servers.

10.4 Live updates: single versus multiple administrators

Typically, Business Modeler IDE developers create a template that contains the data model extensions. The server administrator then uses Teamcenter Environment Manager (TEM) to update the production database with the custom template.

Previously, if further updates needed to be made to the custom template in the database (for example, new LOV values need to be added):

1. The developer updated the custom template in the Business Modeler IDE with the new LOV values.
2. The server administrator shut down the production server and used TEM to install the template to update the production database.

However, the recommendation to shut down the production server each time prior to updating the custom template in the database was too restrictive for administrators who wanted to update live data (for example, LOV values) on a regular basis. Therefore, the live updates functionality was devised to allow administrators to update nonschema data on running production servers.

Now:

- A Business Modeler IDE *developer* works on a template project, tests the changes on the test site, and during the next system downtime, packages the changes and sends them to the server administrator.
- In addition to the developer, there is a *live data Business Modeler IDE administrator*. This administrator works on the live update project. During system downtime, this administrator collects the changes (packages) from the developers and performs a TEM deployment of these changes to the preproduction and production servers. When the server is running, this administrator works on the live update project to add any necessary live updates.

10.4.1 Single Business Modeler IDE administrator of live updates

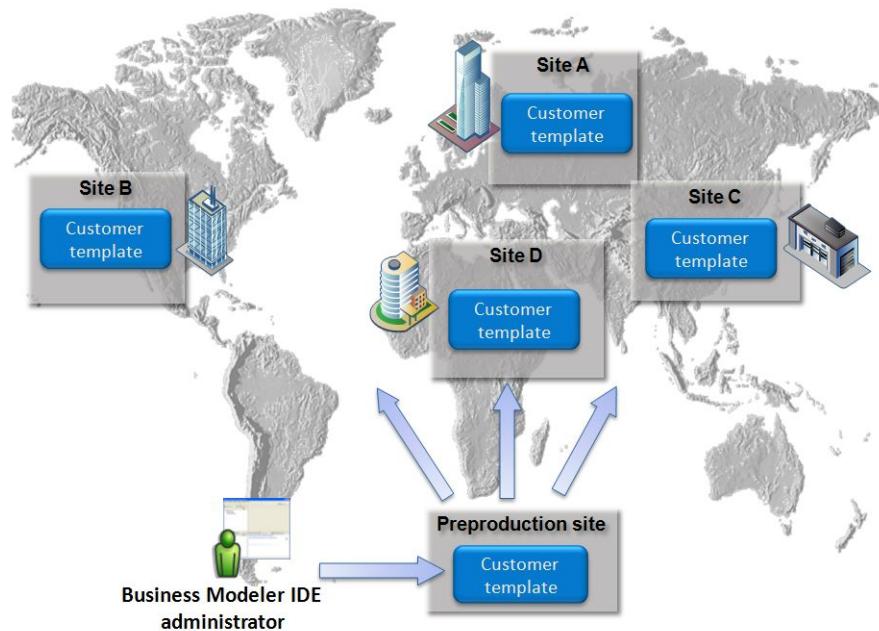
In the single administrator environment, there is only one live updates administrator who is responsible for distributing updates to all sites, both preproduction and production.

In the multiple administrator environment, there may be more than one live updates administrator per site, and they share the responsibility of keeping their respective sites updated.

Use this process if you are a single administrator who makes live updates and distributes them to servers. In this process, make your updates on a preproduction server before sending the updates to production servers. This process is recommended in most situations.

- Control all updates with one administrator.
- Perform live updates in one environment.
- Prevalidate updates.
- Package the template and deploy to each site through Teamcenter Environment Manager (TEM).

Single administrator environment



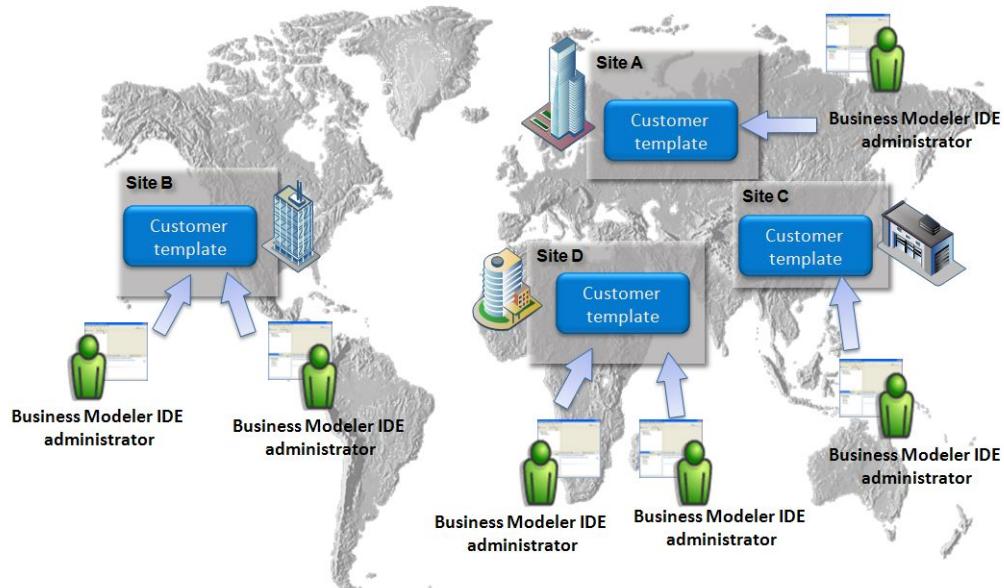
10.4.2 Multiple Business Modeler IDE administrators of live updates

In the multiple administrator environment, there may be more than one live updates administrator per site, and they share the responsibility of keeping their respective sites updated.

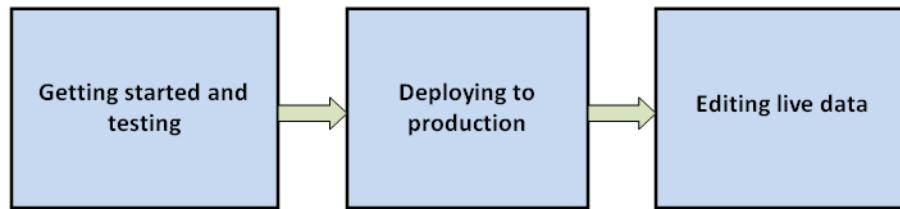
Use this process if you have multiple administrators who make live updates and distribute them to servers. In this process, there is no preproduction server.

- Perform live updates with multiple administrators.
- Clients at each site update live data.

Multiple administrators environment



10.5 Business Modeler IDE process review



Business Modeler IDE process flow

- Developing extensions and testing

First, create a new template project so that you can store your extensions in it for easy deployment to database sites. Then, perform a live deploy to push your template to a test Teamcenter server where you can validate it in a safe environment.

- Deploying a template to a production site

After you validate your template in the test environment, you can update your production environment.

- Editing extensions in a live production site

After you deploy your extensions to a production server, you may want to create and perform a *live update* because you are updating the data on a live system.

- Edit live data

Once the production site is running, you can edit the live data at any time.

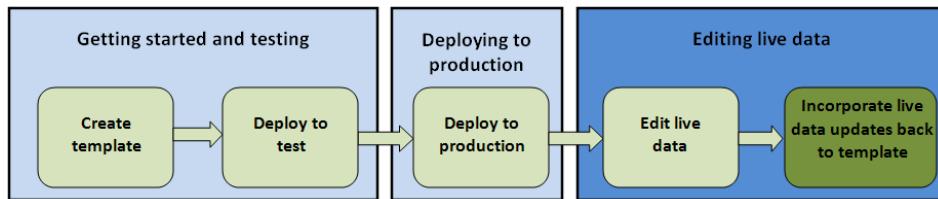
- Incorporate live data updates from the production site

Incorporate all the live data updates performed at the production site back into the development environment.

Note

More details of this step are in the next topic.

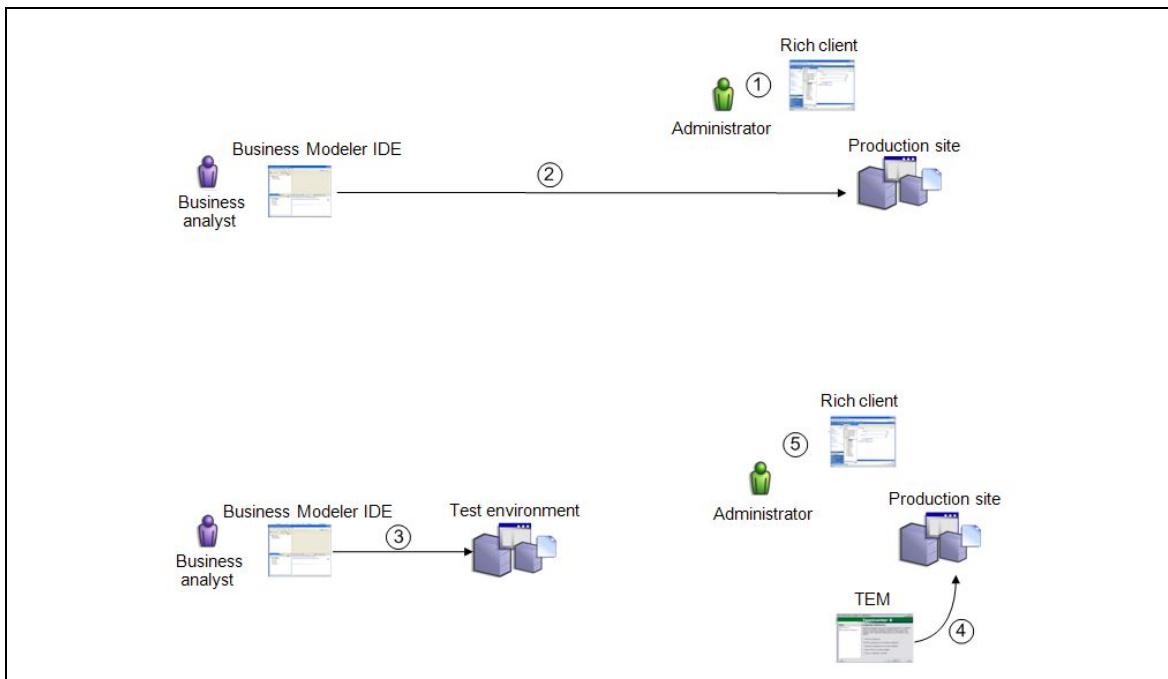
10.5.1 Incorporate live data updates from the production site



Eventually, you must incorporate all the live data updates performed at the production site back into the development environment. This event occurs as you prepare for your next system downtime.

If you do not perform this step, and you deploy your development environment template (which does not include the latest live data updates) to the production site, the live data updates already present in the production site are removed. Because this is undesirable, the system blocks the deployment. To ensure that the deployment is not blocked, you must perform this step.

To incorporate the latest live data, you can run a wizard from the development environment Business Modeler IDE to log on to the production Teamcenter server and retrieve the latest extensions. Or, if your production site is offline from your Business Modeler IDE, you can ask an administrator to run a command line to extract the latest updates from the site and send them to you for incorporation. In either case, once the extensions are incorporated, you can continue with testing, packaging, and deploying to your production site.



Incorporating live data updates from the production site

1. Before you extract the latest live updates, it is a good practice to first block anyone from making any more live updates to ensure you get the latest. You can do this by clearing the **Allow Live Updates** check box in the **Live Update** preference dialog box in the rich client at the production site.
2. From your development environment Business Modeler IDE, launch the Incorporate Latest Live Update Changes wizard. Log on to the production site to pull the latest updates to your Business Modeler IDE. The Business Modeler IDE presents the changes to you in a graphical merge tool. Use this tool to determine which elements you want to keep or discard. Click **Finish** to save the changes to your local Business Modeler IDE template.
3. At this point, you have integrated the latest live updates with your ongoing development work. Deploy this template to your test environment and validate that it works as expected.
4. To update your production site, deploy the template to a production site.
5. Before allowing users to edit live data again, reset the **Allow Live Updates** check box on the **Live Update** preference.

10.5.2 Live update process: single administrator

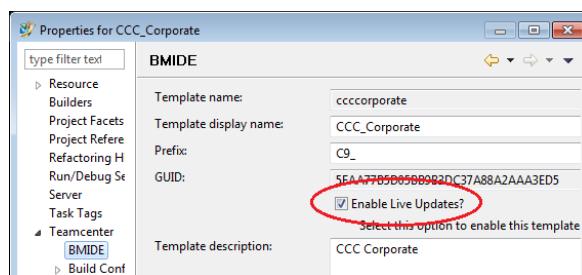
Follow this process when you are a single administrator who performs live updates to a production server.

1. In the developer environment, ensure that a template is enabled for live updates and deploy it to the server. A custom template is enabled for live updates when the **Enable Live Updates?** check box is selected on the template project.
2. In Teamcenter, configure the **Live Update** preference to select the data model elements to update on the server.
3. Create a live update project by choosing **File→New→Project→Business Modeler IDE→New Live Update Project**.
4. Make changes to data in the template and perform live updates on the preproduction and production servers by clicking **BMIDE** on the menu bar and choosing **Deploy Template** or **Deployment Page**.
5. If other sites also require the data updates you have created, such as vendors or partners, you can package the template and send it to each site so the updates can be installed using Teamcenter Environment Manager (TEM).
6. After live updates have been installed on a production server, make live updates visible to end users.
7. In the developer environment, incorporate the latest live updates from the production site by running the Incorporate Latest Live Update Changes wizard.

10.5.3 Enable a template for live updates and deploy it

You must create a Business Modeler IDE template project and install it to the server before you create a live updates project. The Business Modeler IDE template project holds the schema data such as business objects and classes that can only be installed to a production server in a template using Teamcenter Environment Manager (TEM); the live updates project contains the nonschema elements such as LOVs and rules that can be installed to a production server using live update.

1. Create the Business Modeler IDE template project.
2. Enable the Business Modeler IDE template project for live updates with the **Enable Live Updates?** check box.



Note

This check box is selected by default when you create a new template project.

3. Install the Business Modeler IDE template project to the server.

- Test server

If you are installing to a test server only, choose **BMIDE→Deploy Template** on the menu bar.

- Production server

Package the template and install it using Teamcenter Environment Manager (TEM).

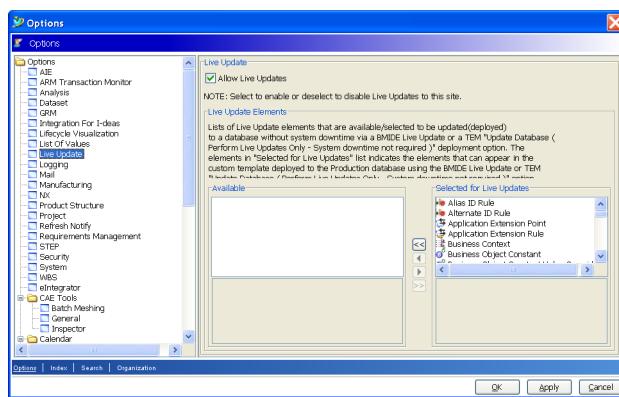
10.5.4 Configure the Live Update preference

Use the **Live Update** preference to select the data model elements to enable for live update.

- To select the data elements to allow for live update use the My Teamcenter application in the rich client **Edit→Options** and select **Live Update** in the left pane. Move the data elements from the **Available** list on the left to the **Selected for Live Updates** list on the right and click **Apply**.
- If you have a four-tier environment, recycle servers in all server manager pools to ensure each warm server receives the latest preference settings. For example, in the .NET server manager administrative interface, click the **Restart Warm Servers** button.

Live update preference

When you perform a live update, only the selected data model elements are updated on the server.



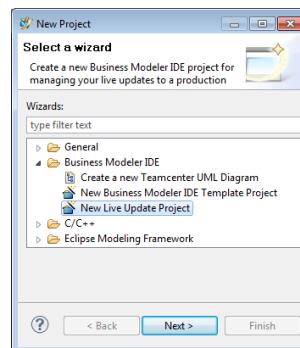
Note

When you select the **Allow Live Updates?** check box, it sets the **BMIDE_ALLOW_LIVE_UPDATES** preference to **true**.

10.5.5 Create a live update project

A *live update project* is a project that manages live data in an already-installed custom template on the server. A live update data project holds only data to be deployed to a running production server. Create one live update project for each custom template enabled to receive live updates on the server.

1. Ensure that a template is installed on the server that is enabled to receive live data updates.
2. Choose **File→New→Project**, and in the **New Project** dialog box, choose **Business Modeler IDE→New Live Update Project**.



3. Fill in the appropriate **Teamcenter Login** information and connect to the server and complete the live update project creation process.
 - a. If there is only one template on the server that is enabled for live updates, the **Next** button is unavailable. Click **Finish**.
 - b. If there are multiple templates on the server enabled for live updates, click **Next**.

Click the arrow in the **Template Name** box to select the template to use for live updates.

Click **Finish**.

The project is created and named *template-name_live_update*.

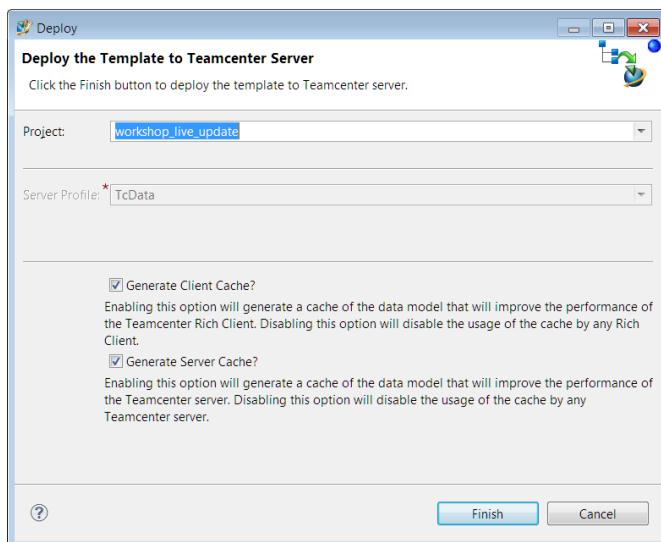
10.5.6 Perform live updates

After you create a live update project, you can revise data, such as lists of values (LOVs), and use the **Deploy Template** feature to distribute the updates to a preproduction server. A *preproduction* server is a Teamcenter server that has an exact copy of the templates that the production sites have. Use this server to test a live update before deploying the live update to the production servers.

After testing, you can use the **Deployment Page** to deploy live updates to production servers. Only use this if you are working in a single administrator environment.

Live updates steps

1. Create data in the live update project.
2. Revise data, such as lists of values (LOVs).
3. Deploy to a preproduction server for testing.
 - Right-click the live update project in the **Business Objects** view and choose **Deploy Template**, or choose **BMIDE→Deploy Template** on the menu bar.
 - Type the password, click the **Connect** button, and when a connection is established, click **Finish**.

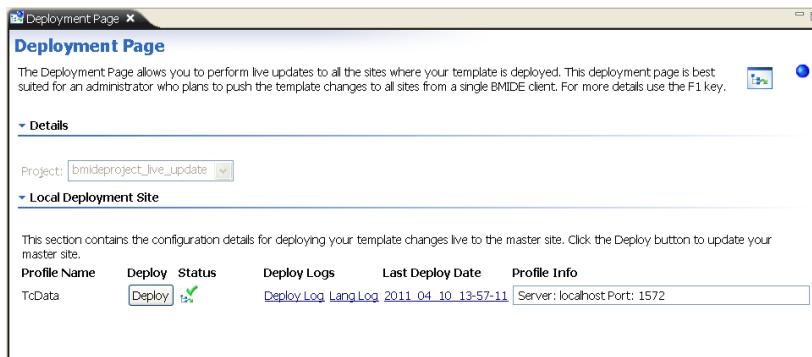


Note

The system checks if the live update project is synchronized with the server. If there is data on the server that is not in the live update project, the Synchronize wizard runs, allowing you to resolve the conflicts.

4. Deploy to production servers.

- After you finish testing and are ready to deploy to the production servers, choose **BMIDE→Deployment Page** or right-click the live update project in the **Business Objects** view or the **Extensions** view and choose **Open Deployment Page**.



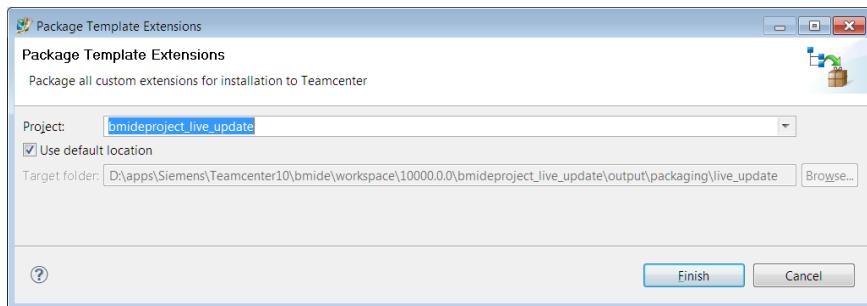
- Click the **Deploy** button to deploy the template to production servers.

10.5.7 Package a live update project for installation to other sites

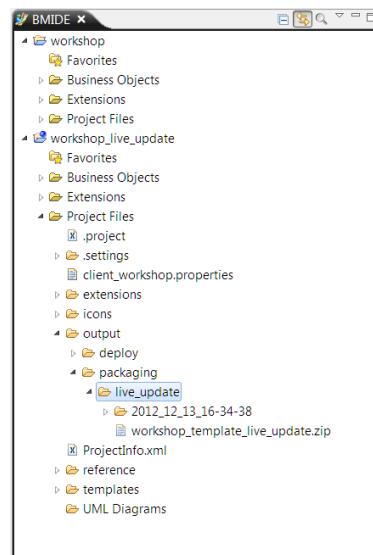
If other sites also require the live updates you have created, such as vendors or partners, you can package the template and send it to the administrators of each site. Administrators install the template to their sites using the **Update the Database (Perform Live Updates – System downtime not required)** option in Teamcenter Environment Manager (TEM).

1. Package the template.

- Select the live update project in the **Business objects** view or the **Extensions** view and choose **BMIDE→Package Template Extensions**.
- Click the arrow in the **Project** box to select the live update project to package and click **Finish**.



The packaged files are saved in the **packaging/live_update** directory under the project. Packaged files are placed in timestamped folders to keep track of packages. The most recent package is in the top folder.



2. Copy the **template-name_template_live_update.zip** file from the **packaging** directory on your Business Modeler IDE client to a directory that is accessible by the administrators at other sites.

3. Administrators at other sites perform the live update by installing the live updates using TEM. In this scenario, TEM has already been used to install the template that the live updates are intended for.
- In the **Feature Maintenance** panel, under the **Teamcenter Foundation** section, select **Update the Database (Perform Live Updates Only – System downtime not required)**. This updates the database with live updates that contain only nonschema data such as LOVs and rules.



Note

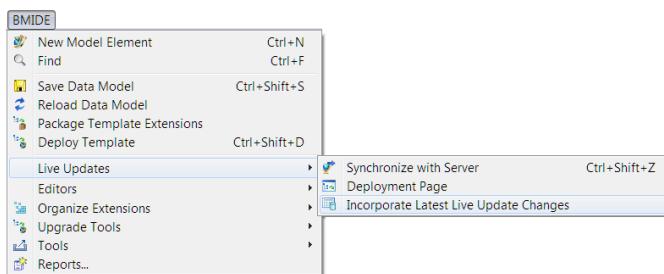
The system checks if the live update project is synchronized with the server. If there is data on the server that is not in the live update project, the update fails. You must synchronize the data model from the Business Modeler IDE, repackage, and attempt the update once more.

If installation of the live updates fails, check the message in the TEM panel. Installation may have failed because the server you are attempting to install to does not have the **Live Update** preference set to accept live changes. In this case, you must ask the administrator of that production server to change the preference to accept the live updates.

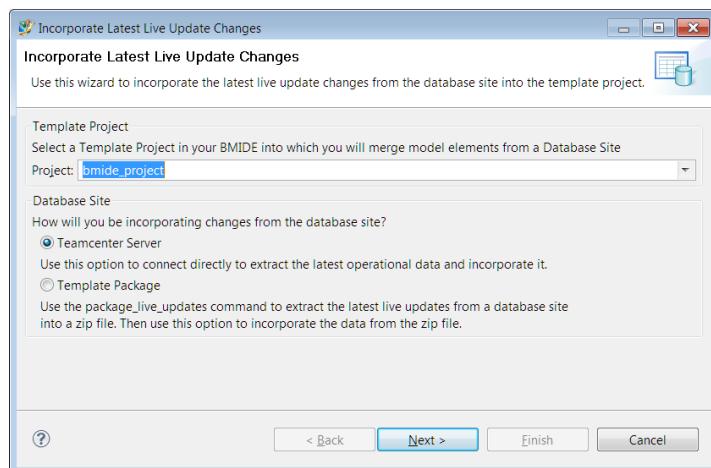
10.5.8 Incorporate latest live updates

At the next system downtime, you must incorporate all of the latest updates from the production environment into your standard template project. Run the Incorporate Latest Live Update Changes wizard to either update directly from the production server or update from a template file obtained from the production server using the **package_live_updates** utility.

1. Choose **BMIDE→Incorporate Latest Live Update Changes**. Click **Next**.



2. Using the **Incorporate Latest Live Update Changes** dialog box:



- In the **Project** box, select the standard project into which you want to incorporate the live updates.
- Under **Database Site**, select the mode to provide the data model:
 - o **Teamcenter Server** – Select if you want to incorporate all the custom live updates data model on a server into your project. This option accesses the server directly to obtain the templates from the server.
 - o **Live Update Zip** – Select if you want to incorporate all the custom live updates data model from a template file into your project. Use this option if you cannot access the server directly.

- Depending on your selection, perform one of the following in the **Source Model** dialog box:
 - o If you selected **Teamcenter Server**, in the dialog box, type your user name and password and click **Connect** to log on to the server.
 - o If you selected **Live Update Zip**, click the **Browse** button to the right of the **Template Zip** box to locate the packaged template's ZIP file.

The system checks if the live update project is synchronized with the server. If there is data on the server that is not in the live update project, the Merge Data Model wizard runs, allowing you to resolve the conflicts.

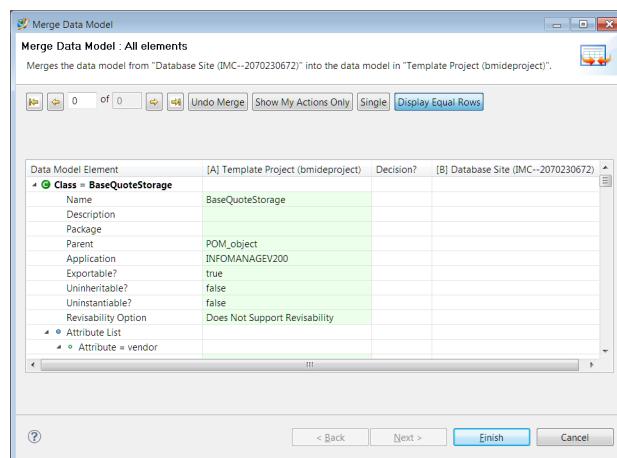
Caution

Be patient. The incorporation may take some time.

3. In the **Merge Data Model** dialog box, resolve any conflicts that the merge compare identified between the data model in your target project and in the source.

Note

Because there are many different scenarios you may encounter when performing a merge, samples are provided in the Merge Samples wizard. For tutorials, choose **BMIDE→Merge Samples**.

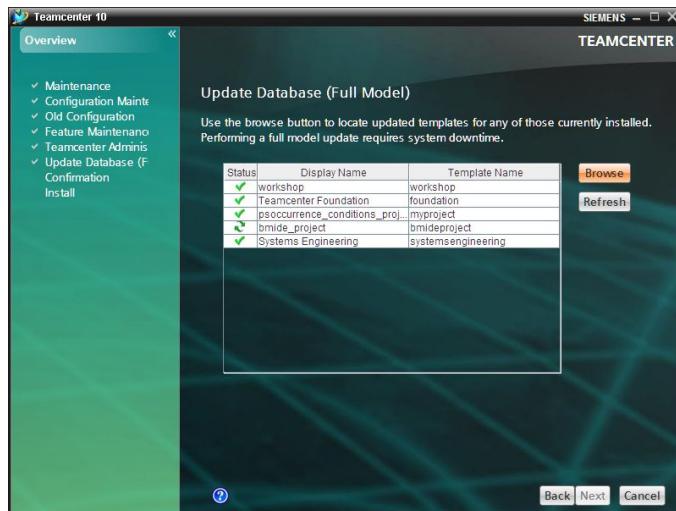


4. Once you incorporate live update changes, you perform a full model update with Teamcenter Environment Manager (TEM).
 - Package your template by choosing **BMIDE→Package Template Extensions**.

- In the **Feature Maintenance** panel, under the **Teamcenter Foundation** section, select **Update the Database (Full Model – System downtime required)**. This updates the database with the full data model in the standard (non live update) template.



- Select your template in the table and click **Next**.



10.6 Live updates reference tasks

Reference tasks about live updates.

- Synchronize data model

The system checks if the live update project is synchronized with the server. You must synchronize the data model with the server from the Business Modeler IDE.

- Data model compare preferences

Set your preferences for data model synchronize and merge between these choices:

- Automatic merge
- Always show Merge Tool during synchronization
- Show Merge Tool only for conflicts that need my resolution

- Make live updates visible to end users

- Package live updates in the database

Run the **package_live_updates** utility to package data model.

- Study the merge samples

Browse through the list of available samples to learn how to use the merge tool. Note that these are only samples. Clicking the **Finish** button has no effect on your data model. To access merge samples, choose **BMIDE→Merge Samples**.

10.6.1 Synchronize data model

Synchronize the data model to ensure your live update project and dependent templates have the most up-to-date data model.

- Select the project to be synchronized and choose **BMIDE**→**Synchronize with Server**, or click the **Synchronize** button  on the toolbar.

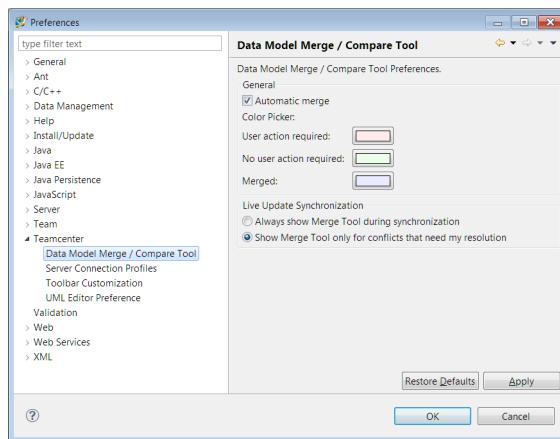
The **Teamcenter Login** dialog box is displayed.

- In the **Project** box, select the live update project you want to synchronize.
- If there are changes to the custom template, the Merge wizard is shown based on the live update synchronization preference settings to help resolve merge conflicts:
 - If the preference is set to **Always show Merge Tool during synchronization** (the default), the Merge wizard is shown to help reconcile the conflicts.
 - If the preference is set to **Show Merge Tool only for conflicts that need my resolution** and there are no change conflicts, all elements of the data model are automerged, and the Merge wizard is not shown. If there are change conflicts, all other elements except for the change conflict elements are automerged, and the Merge wizard is shown.

10.6.2 Data model compare preferences

Perform the following steps to set preferences for the Incorporate Latest Live Update Changes wizard:

1. Choose **Window→Preferences**.
2. In the **Preferences** dialog box, choose **Teamcenter→Data Model Merge / Compare Tool**.



- **Automatic Merge** – used to control the automatic merging of the data model when the Incorporate Latest Live Update Changes wizard is launched. If the check box is selected, all differences except for the change conflict differences are automatically merged when the wizard runs. If the check box is cleared, no merges are done when the wizard is launched.
- **Live Update Synchronization**
 - **Always show Merge Tool during synchronization** – displays the Merge wizard during synchronization even if the tool can automatically reconcile all data model differences without user intervention. This option allows you to inspect the incoming changes made by other users to the live update data in the database. In this mode, you must merge each element. This is the default selection.
 - **Show Merge Tool only for conflicts that need my resolution** – automatically reconciles all data model differences without user intervention. If any change conflict differences are found, the Incorporate Latest Live Update Changes wizard is launched so you can address the conflicts.

10.6.3 Make live updates visible to end users

After live updates have been installed on a production server, the updates are visible to end users who log on after the update. However, users logged on when the update is made must log off and log on again to ensure that the servers cycle through the changes.

When each user logs on, a server is started and assigned to the client. During startup, the types, properties, and LOVs in the database are built into a cache for performance reasons. Each server has its own cache. When a user updates an LOV to the database, each client still sees the original LOVs and rules because server caches are not updated. Administrators ask users to log off and log on again to see the changes so that a new cache is built.

In four-tier environments, administrators can use the server manager to restart servers at off-peak hours to ensure all live update changes are reflected in servers. On J2EE consoles, a system administrator can facilitate warm server refresh. (This is not available on .NET.)

10.6.4 Package live updates in the database

Run the **package_live_updates** utility to package data model for a template that resides on the server. The template package file that is created includes all the live updates in the template. This is similar to packaging a template within the Business Modeler IDE, but this utility is run on the Teamcenter server. This utility can only be run on a template that is enabled for live updates deployment.

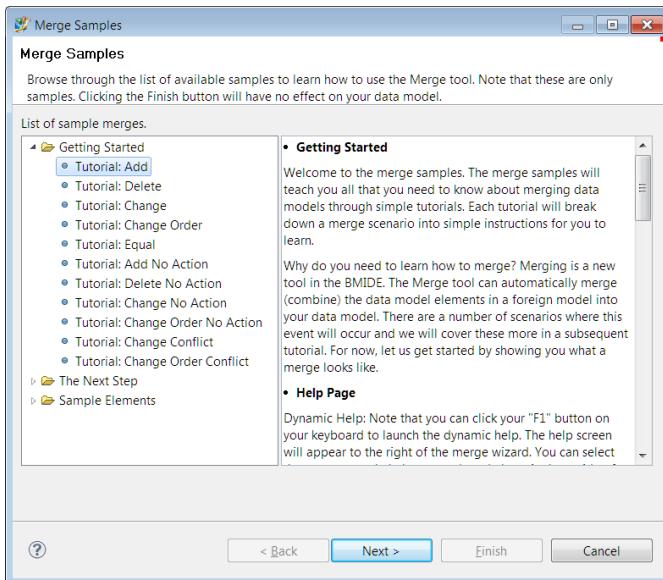
For example:

```
package_live_updates -u=user-id -p=password -g=group  
-template=template-name -dir=directory
```

After running this utility, you can merge the live updates that the template contains into a template on your system by running the merge wizard.

10.6.5 Study the merge samples

To become familiar with merge scenarios, study the merge samples. Choose **BMIDE→Merge Samples**.



Merge tool features

Key features of the merge tool are:

- Stepping – click the forward and backward arrows to step through each change.
- Filtering – To see only the rows where there are differences, click the **Display Equal Rows** button so that it is unselected. This is a toggle button. When it is unselected, only rows with different values are shown. When it is selected, all rows are shown, including all those with equal values.
- Automerger – click the **Auto Merge** button to let the tool merge all changes for you. Merged (changed) elements display a light-blue background for easy identification.

Before automerge

Merge Data Model : All elements
Merges the data model from "Database" into the data model in "Your BMIDE".

| Data Model Element | [A] Your BMIDE | [B] Database |
|--------------------|----------------|--------------------|
| LOV = Cities | | Add B to A |
| Name | | Cities |
| Description | | Description |
| Package | | ListOfValuesString |
| Type | | Exhaustive |
| Usage | | |
| Lower Range | | |
| Upper Range | | |
| Based On LOV | | |
| Direct Dependency | | true |
| Values List | | |
| LOV Value | | |
| Value | | Dallas |
| Description | | |
| Package | | |
| Condition | | isTrue |
| COTS Deleted? | | false |
| LOV Value | | |
| Value | | Detroit |
| Description | | |
| Package | | |
| Condition | | isTrue |
| COTS Deleted? | | false |
| LOV Value | | |
| Value | | Seattle |
| Description | | |
| Package | | |
| Condition | | |
| COTS Deleted? | | |

Buttons at the bottom: ? < Back Next > Finish Cancel

After automerge

Merge Data Model : All elements
Merges the data model from "Database" into the data model in "Your BMIDE".

| Data Model Element | [A] Your BMIDE | [B] Database |
|--------------------|--------------------|--------------------|
| LOV = Cities | | Reset |
| Name | Cities | Cities |
| Description | Description | Description |
| Package | | ListOfValuesString |
| Type | ListOfValuesString | Exhaustive |
| Usage | Exhaustive | |
| Lower Range | | |
| Upper Range | | |
| Based On LOV | | |
| Direct Dependency | true | true |
| Values List | | |
| LOV Value | | |
| Value | Dallas | Dallas |
| Description | | |
| Package | | |
| Condition | isTrue | isTrue |
| COTS Deleted? | false | false |
| LOV Value | | |
| Value | Detroit | Detroit |
| Description | | |
| Package | | |
| Condition | isTrue | isTrue |
| COTS Deleted? | false | false |
| LOV Value | | |
| Value | Seattle | Seattle |
| Description | | |
| Package | | |
| Condition | | |
| COTS Deleted? | | |

Buttons at the bottom: ? < Back Next > Finish Cancel

10.7 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

10.8 Summary

The following topics were taught in this lesson:

- Understand the difference between single administrator and multiple administrators live update process.
- Enable a template for live updates.
- Create a live update project.

Review Business Modeler IDE

| Lesson | Topics |
|---|--|
| <i>BMIDE fundamentals</i> | <ul style="list-style-type: none"> Describe a BMIDE project and process. Create a project in the BMIDE. Set up and test a connection to Teamcenter. |
| <i>BMIDE process and data model</i> | <ul style="list-style-type: none"> Perform the BMIDE extension process. Create an item business object. Perform a BMIDE deployment. |
| <i>Item business object configuration</i> | <ul style="list-style-type: none"> Create item business objects. Configure a variety of item properties. Add an icon to a custom business object. |
| <i>Form business object configuration</i> | <ul style="list-style-type: none"> Create form business objects. Configure a variety of form properties. |
| <i>LOV (list of value) extensions</i> | <ul style="list-style-type: none"> Create a LOV and attach the LOV to a property. Create a cascading and dynamic LOV. |
| <i>Relation business object configuration</i> | <ul style="list-style-type: none"> Create a relation business object. Configure relation properties. |
| <i>Dataset business object configuration</i> | <ul style="list-style-type: none"> Create a dataset business object. Create a tool extension. |
| <i>Option extensions and BMIDE reports</i> | <ul style="list-style-type: none"> Create the BMIDE options: note type, status type, units of measure. Compare data model from two different sources with the BMIDE reports. |
| <i>Rule extensions</i> | <ul style="list-style-type: none"> Create naming rule and revision naming rule. Create a display rule. Add deep copy rules on item revision. Create conditions to use with naming rules. |
| <i>Data model live updates</i> | <ul style="list-style-type: none"> Understand the live update process. Create a live update project. Deploy a live update project. |

Lesson

11 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.

Help topics

Additional information for this lesson can be found in:

- *Organization Guide*
- *Security Administration Guide*
- *Utilities Reference*

11.1 Introduction to Organization

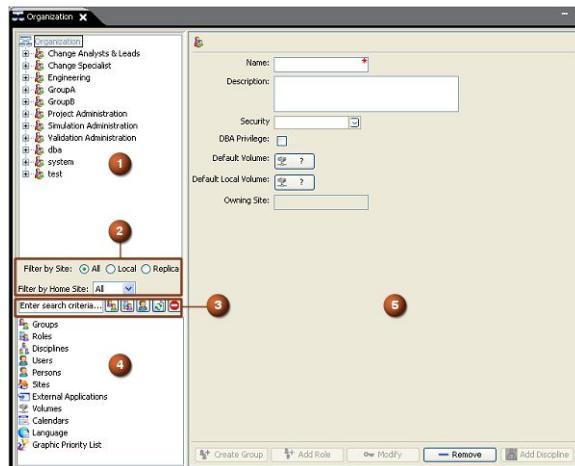
The Organization application enables you to create and maintain your company's organization within Teamcenter by organizing user accounts and their respective permissions and user groups. User accounts help you:

- 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 external applications
- Miscellaneous tasks:
 - Defining languages
 - Defining graphic priority lists

11.1.1 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 **Filter by Home Site** box Use the **Filter by Home Site** box to filter objects (group, role, or user) by owing location.
- 3 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.
- 4 **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, and calendars.
- 5 Definition pane Displays the properties for the selected Organization object.

11.1.2 Organization hierarchy

The Organization application supports two ways to create and manage your company's organization within Teamcenter:

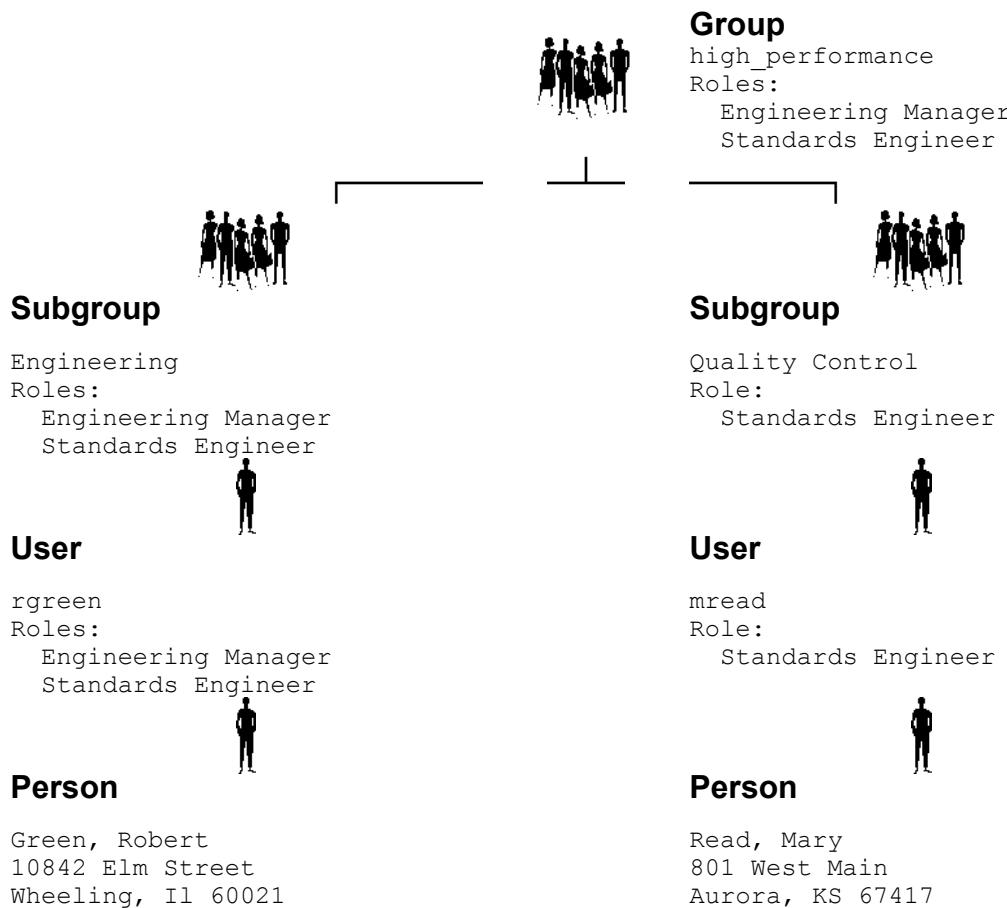
- From the bottom up using the **Organization List** tree.

You also use the **Organization List** tree to manage sites, volumes, and calendars.

- From the top down using the **Organization** tree.

11.1.3 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:

- o Using the **Security** setting, which allows or restricts access to data.
- o 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.

11.2 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.

11.2.1 What is a person?

Persons are individuals who work at your site. A person has properties such as **Name**, **Address**, and **Employee Number**.

- Consider creating all persons at your site using the following naming convention:

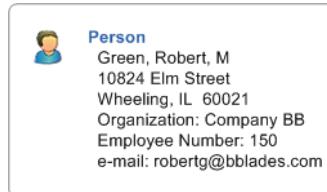
last-name, first-name, middle-initial

- 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



11.2.2 What is a user?

A *user* is a person with an account known to the Teamcenter system. One person can have several user accounts in Teamcenter. The Teamcenter implementation of user is completely separate from any 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, along with defining multiple citizenships.
- Assign a license bundle to a user.

Example



11.2.3 Specifying password restrictions

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 (**PASSWORD_minimum_characters**)
- Mixed case required (**PASSWORD_mixed_case_required**)
- Minimum number of alpha or numeric characters required (**PASSWORD_minimum_alpha** and **PASSWORD_minimum_digits**)
- 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
```

Note

If a user attempts to log on to Teamcenter without entering a password, a logon failure occurs.

11.2.4 Teamcenter Security Services

In addition to application authentication within Teamcenter, Security Services allows you to move from one Teamcenter product solution, such as Teamcenter Enterprise, 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 **Idapsync** 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 10.1.

11.3 What is a role?

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.

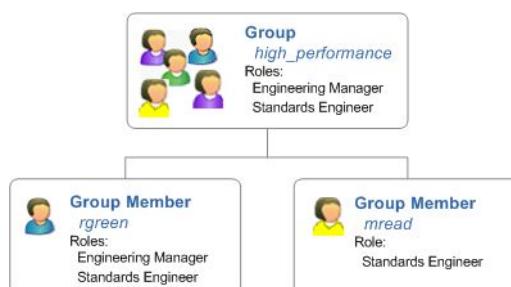
Tip

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.
- Assign a default rule within a group.

Example



Robert Green is an Engineering Manager. In addition to his responsibilities as Engineering Manager, Robert must also perform standards work. Therefore, user **rgreen** has been assigned two roles in the **high_performance** group: **Engineering Manager** and **Standards Engineer**.

11.4 What is a group?

Groups contain members (users) who take on a role or multiple roles in the group. Groups represent data ownership and therefore control data access. Two groups are provided with Teamcenter: **dba** and **system**.

Warning

Do not delete the **system** group provided with Teamcenter. It is required for the product to function properly.

- 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

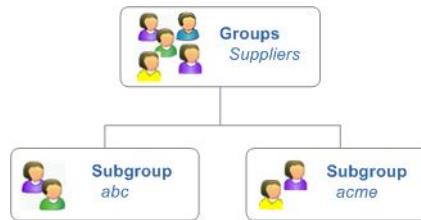


11.4.1 What is a subgroup?

A *subgroup* is a group with another group designated as its parent. A subgroup can also be designated as a parent group itself. The position of subgroups within the Organization hierarchy can be managed by parenting and reparenting groups.

- Subgroups are an excellent way to organize your users.
- Subgroups inherit access permissions, volumes, and preferences from their parent.

Example



11.4.2 Group hierarchies

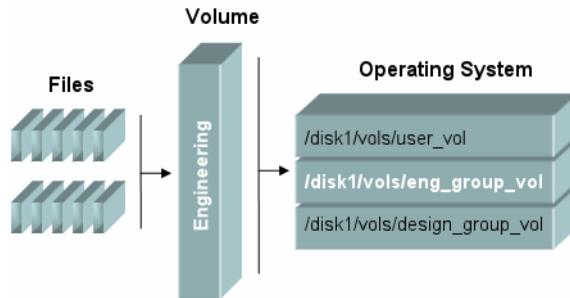
Groups are organized into one or more trees or hierarchies. Each group has exactly one parent group (unless it is at the top or root of the hierarchy, when it has no parent) and can have one or more subgroups.

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. |

11.5 What is a volume?

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



- 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.

11.6

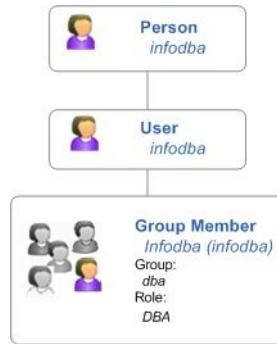
Considerations for managing administration accounts

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.

11.6.1 infodba account



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.

11.6.2 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 → Logging tab). |
| 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. |

11.7 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. Do not delete the roles provided with Teamcenter. They are required for the product to function properly.
Roles, volumes, and persons are independent definitions. Any of these can be created without consideration of the other definitions.
 - a. Create roles.
 - b. Create volumes.
 - c. Enable File Management System (FMS) control of new volumes.
 - d. Create groups and subgroups. Do not delete the **system** group provided with Teamcenter. It is required for the product to function properly.

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.

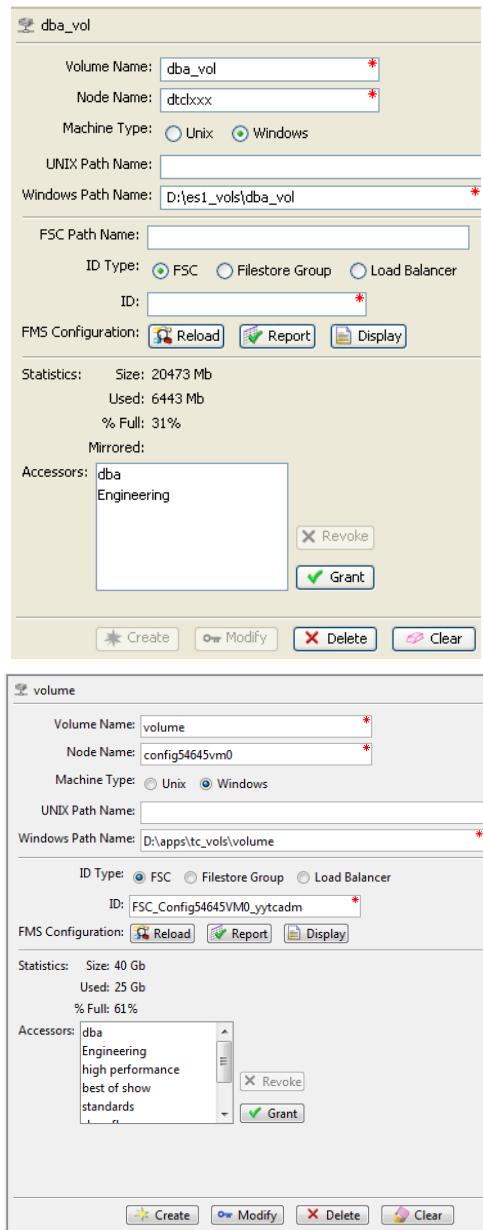
The **make_user** utility allows you to create your organization from a command line or script.

11.7.1 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**.

11.7.2 Creating a volume



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. 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.

7. Grant users or groups access to the volume.
8. Click **Create**.

11.7.3 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**.

The system saves the changes to the volume definition.

11.7.4 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.

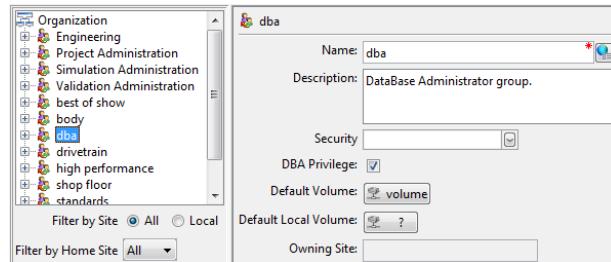
When you grant access to a group, the system generates a subdirectory identified by the group name. The system does not generate subdirectories for subgroups, regardless of whether access inheritance is enabled. Subgroups share the directory of the original group.

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 nonzero number to allow inherited access.

Note

Inherited access applies to all volumes including default local volumes, also known as *store and forward volumes*.

11.7.5 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.

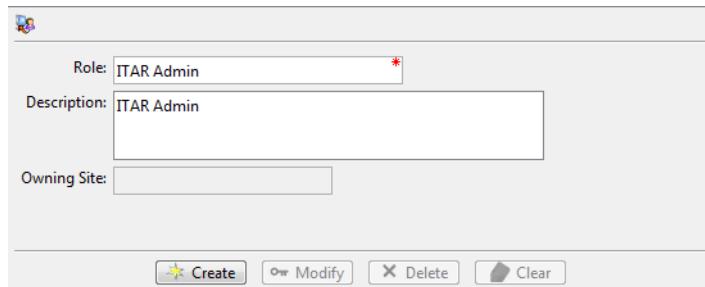
- Using the **Organization** tree

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.

11.7.6 Creating a role



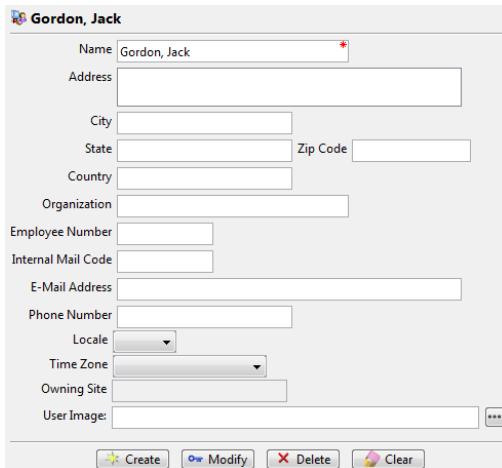
You create roles to reflect the skills and responsibilities of the users in your organization. Roles can be created using the **Organization List** method described or you can create and add a role to the **Organization** tree using the Organization Role wizard.

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.

11.7.7 Creating a person

Person definitions contain real-world information about individual Teamcenter users.



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.

11.7.8 Creating a user

You create user accounts to identify each individual who interacts with Teamcenter.

The screenshot shows the 'User Creation' dialog for a user named 'Gordon, Jack (jgordon)'. The main fields are filled with the values: Person Name: 'Gordon, Jack', User ID: 'jgordon', OS Name: 'jgordon', and Password: (empty). The 'Default Group' is set to 'dba'. Below these are sections for 'ADA/ITAR Attributes' which include dropdown menus for IP Clearance, Gov't Clearance, and TTC Date (set to 'No date set'). There are also fields for Geography, Nationality, and Citizenship, each with a small icon and a question mark help button.

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.

Note

- If a user is created without specifying a password, Teamcenter assigns the user ID as the password.
 - If a user attempts to log into Teamcenter without entering a password, a login failure occurs.
 - Last System Access 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.
- Select the top-level **Users** node from the **Organization List** tree.

The **Users** pane appears.

2. Complete the following information:

Note

- Do not use the delimiters defined in the **TC_user_delimiters** preference when entering user information in either the **Person Name** box or the **User ID** box. (If the **TC_user_delimiters** preference is not set, parentheses () are the default delimiters.) Otherwise, the user name and user ID display incorrectly in the **Organization List** tree.
- If you inadvertently use the characters set in the **TC_user_delimiters** preference when you create a user or a person, use the **make_user** utility to correct the user ID and person name delimiters.

- 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 the user's OS name in the **OS Name** box.

Note

Teamcenter uses the user's OS name as a backup e-mail address if no e-mail address is set in the **Person** object. This allows the user to receive notifications and subscriptions.

- 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.

- **Citizenships**

Specifies the citizenships of the user. The user can have multiple citizenships. This box is optional.

To add a citizenship to the **Citizenships** list, enter a two-letter country code in the text box and click the plus button . To add additional citizenships, click the **Edit** button  to set this box in edit mode, enter the two-letter country code in the text box, and click the plus button .

To sort the list of multiple citizenships alphabetically, use the **Sort** button .

To remove citizenship entries from the **Citizenship** list, select a citizenship in the **Citizenship** list and click the minus button .

Note

- o Citizenship is a two-letter country code from ISO 3166, for example, **US** (United States) and **GB** (Great Britain).
- o If a country code LOV is attached to the **fnd()citizenships** property of the **User** business object, a combination box is displayed to allow the selection of a citizenship from the country code list.
 - i. Set the licensing level to the appropriate level for the tasks the user performs.

License levels are used to enable usage by time or by features. For descriptions of the available license levels, see your license agreement documentation.

For information about administering Teamcenter licenses, see the *System Administration Guide*.
 - j. Select the appropriate license bundle from the **License Bundle** list. As an administrator, you can assign a license bundle to a user.

If the selected license bundle has a base license level, the **License Level** box is updated with this value and is made noneditable.

- k. Select the home site from the **Home Site** list. This is the site from which the user is physically present and works.
- l. Select which remote sites, if any, the user cannot interactively log into by clicking the **Select Sites** button to display the **Site Selection** dialog box.

Note

Users are always able to log on to their home site.

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.

11.7.9 Group member settings



| 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> |

11.7.10 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.

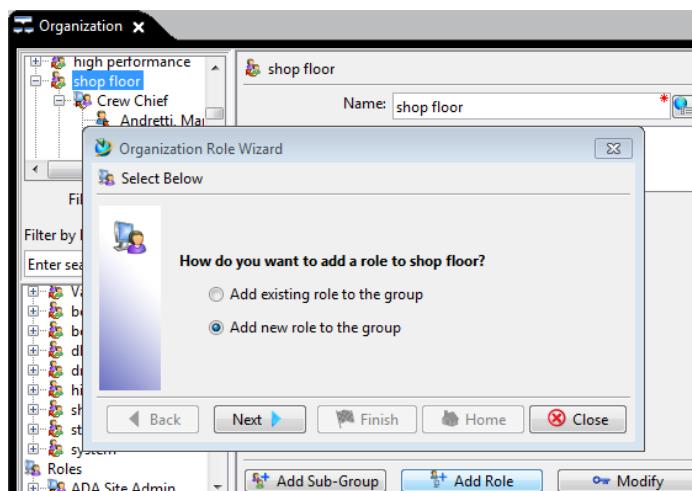
1. Select the group node in the **Organization** tree to which you want to add the role.

The **Groups** pane appears.

2. Click **Add Role**.

The Organization Role wizard appears.

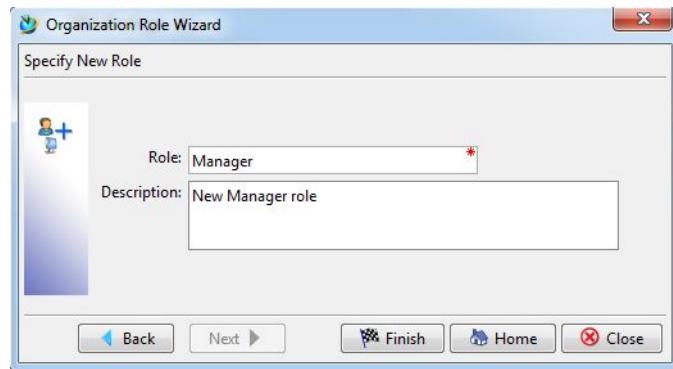
3. Select **Add new role to the group** and click **Next**.



Note

The first role added to a group becomes the default role for that group.

4. Type the following information and click **Close** to create the new role or click **Finish** to perform another action:
 - A new role in the **Role** box.
 - Optionally, a descriptive character string in the **Description** box.



5. If you clicked **Finish**, 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.

11.7.11 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.

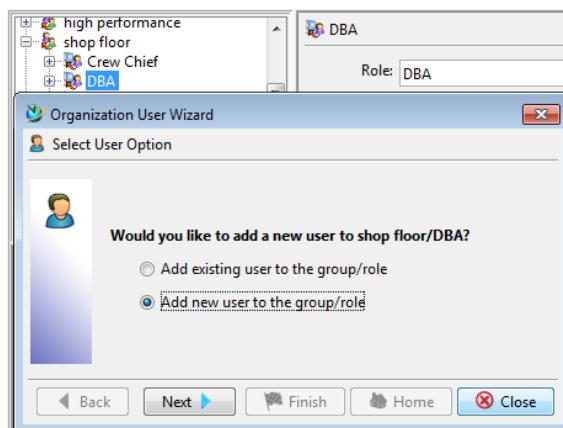
1. Select a role node from the **Organization** tree.

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 the user's 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.



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. Set the licensing level to the appropriate level for the tasks the user performs.

License levels are used to enable usage by time or by features.

For descriptions of the available license levels, see your license agreement documentation.

For information about administering Teamcenter licenses, see the *System Administration Guide*.

- h. Select the appropriate license bundle from the **License Bundle** list. As an administrator, you can assign a license bundle to a user.

If the selected license bundle has a base license level, the **License Level** box is updated with this value and is made non-editable.

- i. When all required input is complete, click **Next** or **Finish**.
- j. 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 appears. 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.

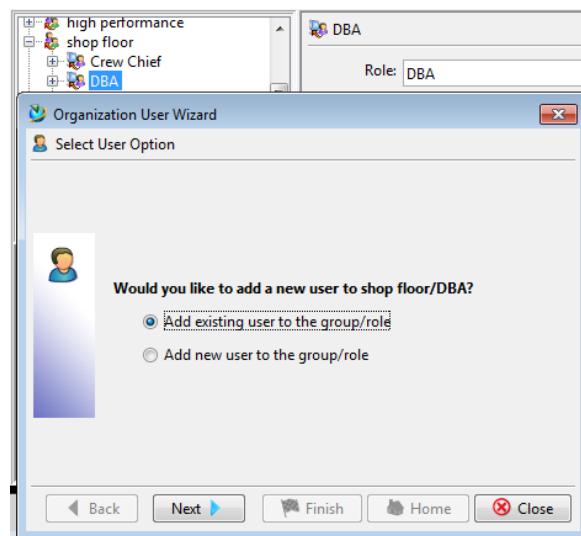
7. Click **Close** to dismiss the Organization User wizard or repeat steps 3 through 6 of this section to add additional users to the same role/group combination.

11.7.12 Add an existing user to a role/group using the Organization User wizard

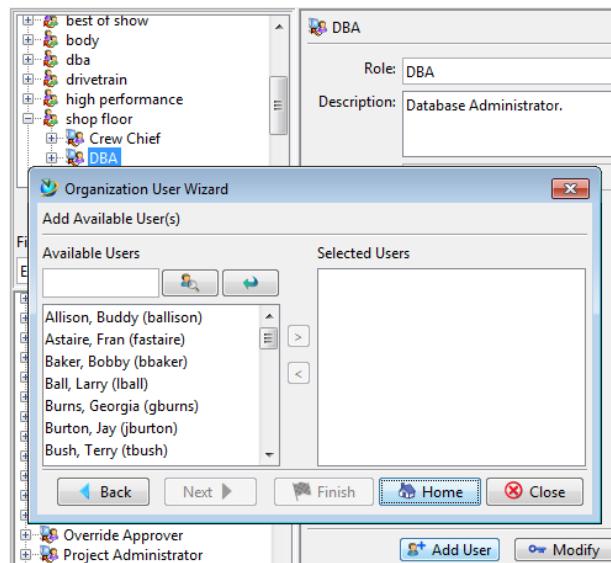
The Organization User wizard can be used to add an existing user to a group during the process of creating a group/role combination in the **Organization** tree.

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.
2. Click **Add User**.
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 right arrow (>) or left arrow (<) buttons. After you select all the users to be added, click **Finish** to continue or **Close** to dismiss the wizard.

5. If you clicked **Finish**, 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.

11.7.13 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.

Note

A replicated (remote) user is designated by having two green dots beside the user symbol to designate it as a remote object . As an administrator, you cannot modify the associated data. However, you can remove remote users from your local organization structure.

11.7.14 Inactivate a user account

1. Perform the following:
 - Select a user definition from the **Organization** tree.

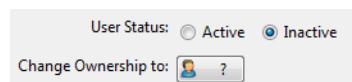
The **Users** pane displays the properties of the user definition.

Note

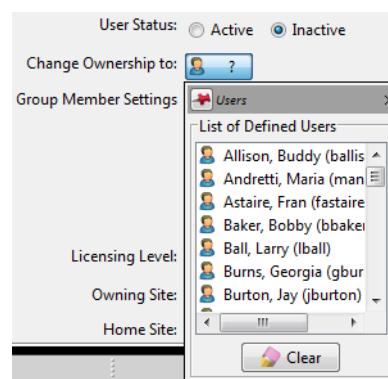
Because database objects are owned by users, you must decide what to do with any objects owned by the user being deactivated. You can either change ownership of these objects to another user or allow ownership to be retained by the inactive user.

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 displays on the **Change Ownership** button.

6. Click **Modify**.

7. If the **Inactivate All Members** dialog box appears 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.

11.7.15 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.

11.8 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 (group, role, and user). To use Organization find, type the text in 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 informs you of this.

You can also filter objects by owning location (remote or local).

Note

Replicated (remote) objects (user, group, role, and person) have two green dots beside them to designate them as remote objects .

After performing a search, you can reload the **Organization** tree by clicking the **Reload** button . This refreshes the organization in the current session; changes made in different sessions cannot be guaranteed to be updated.

Note

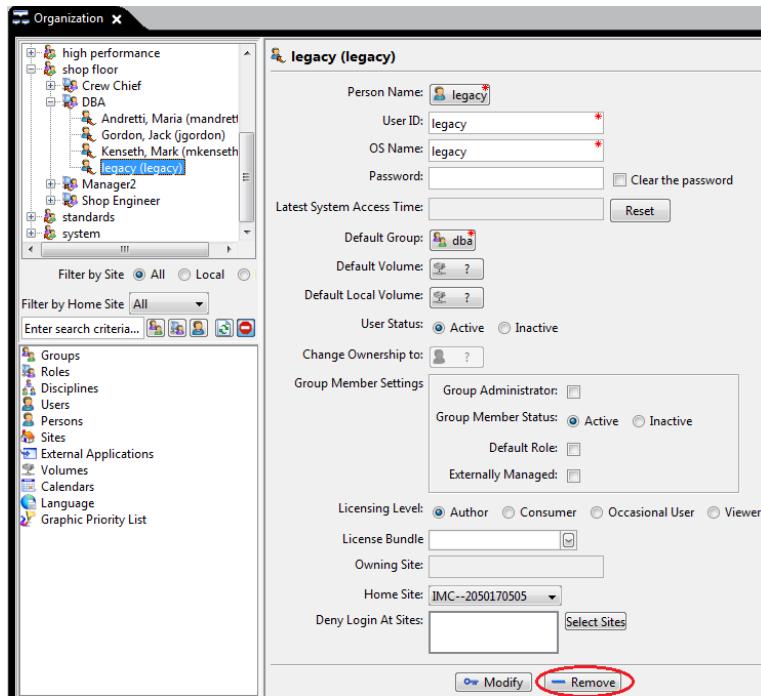
To get new organization objects (groups, subgroups, roles, users, and disciplines) and modified objects in another session to display in the **Organization** tree, select **View→Refresh Window** in the **Organization** perspective. This action refreshes components loaded in the current session so that changes made in another session are available in the current session. Because this action usually takes a long time, use this when you want your current session to be in sync with the latest change in the database by another session.

To suppress the display of inactive members in a group in the **Organization** tree, use the **Suppress inactive group members** button in the **Organization** pane.

11.9 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.

11.9.1 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.

11.9.2 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 on 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.

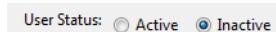
- You cannot deactivate a group member if they have any pending Workflow tasks. You must first delegate these tasks to another group member and then deactivate the user. You can use the **global_transfer** utility to transfer one user's tasks to another user.

You can also reassign the user's tasks using the My Teamcenter inbox feature.

- Select the user (group member) you want to deactivate from a group or subgroup in the **Organization** tree.

Teamcenter displays the user's information.

- In the **Group Member Settings** section, select the **Inactive** option.



- Click **Modify**.

11.9.3 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, select the **Active** option.

User Status: Active Inactive

3. Click **Modify**.

11.9.4 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.

11.10 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.

11.10.1 make_user utility examples

- **Create a person and user and add him 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
```

An *option_name* is any command line argument.

An *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
```

11.11 Set your own user variables for the classroom activities

Teamcenter Application and Data Model Administration

The activities were set with your own user information on the title page of the electronic activities. Your user variables will appear in the activities.

If changes are needed, set them as supplied by your instructor.

Teamcenter Application and Data Model Administration

Student Activities
August 2013
MT25460 - Teamcenter 10.1

| | |
|---------------------------|---|
| User ID: | <input type="text" value="jgordon"/> |
| Password: | <input type="text" value="jgordon"/> |
| User Name: | <input type="text" value="Gordon, Jack"/> |
| Student Files location: | <input type="text" value="D:\users\student\student_files"/> |
| TC_ROOT location: | <input type="text" value="D:\tc_root"/> |
| TC_DATA location: | <input type="text" value="D:\tc_data"/> |
| Installed Node ID: | <input type="text" value="tc_TC101svr"/> |
| Connection Profile: | <input type="text" value="Teamcenter Training"/> |
| Teamcenter Help location: | <input type="text" value="https://lmdcontent.industrysoftware.automation.s"/> |

Make sure to click **Save** to save your user variables.

11.12 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

11.13 Summary

The following topics were taught in this lesson:

- Define the organization hierarchy.
- Define administrative privileges.
- Create an account.
- Manage the organization hierarchy.
- Search the organization.
- Set up accounts manually.

Lesson

12 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.
- Control access to working and in-process data.
- Configure group security to control access for specific groups of users.
- Configure project-level security to control access to data in specific projects.

Help topics

Additional information for this lesson can be found in:

- *Access Manager Guide*
- *Security Administration Guide*

12.1 Getting started with Access Manager

Access Manager (AM) controls user access to data objects stored in Teamcenter by using the AM rule tree applicable at your site. Teamcenter is installed with the AM default rule tree. Using your site requirements, you configure the AM rule tree consisting of:

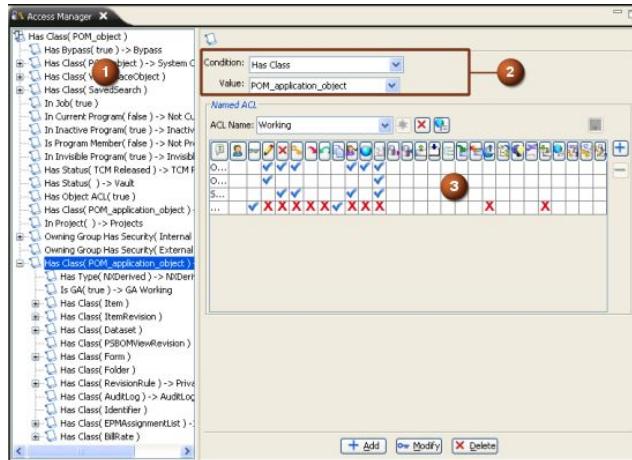
- Rules (that are defined as branches) based on your data objects.
- Access control lists (that end the branches) to grant or deny access.

Rules and access control lists (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.

12.1.1 Access Manager interface



- 1 Rule tree pane
- 2 Rule properties
- 3 Named ACL table

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.

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.

Displays the ACL name and accessor entries for the rule selected in the rule tree. You can create, modify, and delete named ACLs.

12.1.2 About 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.

Note

Rules and ACLs do not control the creation of objects. They only determine what operations can be performed on existing objects.

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.

12.1.3 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.

12.1.4 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.

12.1.5 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).

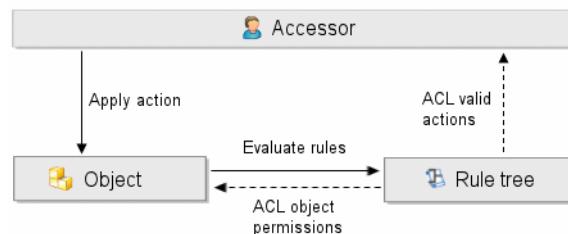
Note

Rules do not control the creation of objects. They only determine what operations can be performed on existing objects.

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).



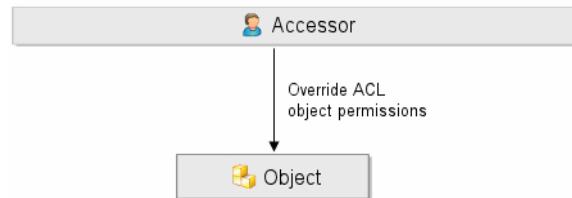
User actions against objects cause the rule tree to be evaluated to dynamically build an access control list for the object. The ACL controls permissions for the object and determines who (accessors) can do what (actions) to the object.

12.1.6 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.

Note

ACLs do not control the creation of objects. They only determine what operations can be performed on existing objects.

- Each ACL contains a list of accessors and the privileges granted, denied, or not set for each accessor.
- Each individual pairing of an accessor with their privileges is considered a single access control entry (ACE).

12.1.7 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.

12.1.8 Lifecycle 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. |

12.2 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
```

12.2.1 How rules are defined

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 | | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

12.2.2 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.

12.2.3 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.

12.2.4 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 |

12.2.5 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
 - ☐ Condition {Value}→Named ACL
 - ☐ Condition {Value}→Named ACL

12.2.6 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 |
| 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.

| T 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.

12.2.7 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 as a new object. |
| Note | | |
| | | It still allows copy and paste of the object as a reference, with no new object created. |
|  | 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. |

| Symbol | Privilege | Description |
|--------|---------------------------------|---|
| | Transfer in | Controls the privilege to assign ownership of objects when they are imported into the database. |
| | 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. |
| | | 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. |
| | | 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. |
| | IP Admin | Enables users to add users to manage IP licenses. |
| | ITAR Admin | Enables users to add infodba users to manage ITAR licenses. |
| | CICO | Controls administrative override privileges for the checkin, checkout, transfer checkout, and cancel checkout features. |
| | Translation | Controls the privilege to add translated text using the Localization button. |
| | View/Markup | Controls the privilege to view and create markups. |
| | Batch Print | Controls the privilege to print multiple objects. |

| Symbol | Privilege | Description |
|---|--------------------------------|--|
|  | Administer ADA Licenses | Controls the privilege to create, modify, or delete ADA licenses in using the ADA License application. |

12.2.8 Categories of accessors

| Accessor | Description |
|-----------------------------|---|
| Owning User | Users who initially created an object. Ownership can be transferred and additional privileges (for example, delete) are usually granted to an object's owner that are not granted to other users. |
| 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. |
| Group | Project-oriented cluster of users. This allows all users in a group to access a common pool of project data regardless of the actual work each user performs. |
| Groups with Security | Users who have the given security value, either Internal or External . |
| Role | Function-oriented cluster of users. |
| Role in Group | Users who have a specific role in a specific group. Use this for granting privileges to all users performing the same skills and/or responsibilities on the same project. |
| Role in Owning Group | Users with a specific role in the object's owning group. This is useful for granting privileges to an inner-circle of users with the same skills and/or responsibilities on the same project. For example, all designers in the owning group are usually granted write privilege on their development data. |
| System Administrator | Users who are members of the system administration group. |
| Group Administrator | User who has special maintenance privileges for the group. |
| Site | A specific site. |
| Remote Site | Any site that is not local. |
| World | Any user, regardless of group or role. |
| User | A specific user. |

| Accessor | Description |
|--------------------------|--|
| Approver (RIG) | <p>Users who are members of a signoff team in a workflow process with a specific role in a specific group (RIG).</p> |
| | <p>Note</p> <p>This accessor is only used in workflow ACL and matches the signoff RIG requirements for the release level associated with the workflow ACL.</p> |
| Approver (Role) | <p>Users who are members of a signoff team in a workflow process in a specific role.</p> |
| | <p>Note</p> <p>This accessor is only used in workflow ACL.</p> |
| Approver (Group) | <p>Users who are members of a signoff team in a workflow process in a specific group.</p> |
| | <p>Note</p> <p>This accessor is only used in workflow ACL.</p> |
| Approver | <p>Users who are members of a signoff team in a workflow process regardless of their role and group.</p> |
| | <p>Note</p> <p>This accessor is only used in workflow ACL.</p> |
| Task Owner | <p>Task owner is given privileges for the task's target data.</p> |
| Task Owning Group | <p>The owning group are given privileges for the task's target data.</p> |
| Responsible Party | <p>Users responsible for performing a particular task. This ensures that only the user assigned as responsible party is given privileges to the task's target data.</p> |
| Project Team | <p>Team members in a particular project.</p> |
| Project Teams | <p>Team members in any active project for the object.</p> |

| Accessor | Description |
|--------------------------------------|---|
| Current Project Team | Users who are members of a particular current project team. Applicable only when the project is set as the current project of the team members and if the current project is active. |
| Current Project Teams | Users who are members of current project teams. Applicable only when the object is in the current project of the team members, and the current project is active. |
| Role in Projects of Object | Users who have a specific role in one of the projects of the object. This accessor is affected by the values set in the AM_PROJECT_MODE preference. It is effective only when the user is logged in with the specified role in the current project, and the current project is one of the projects assigned to the defined object. |
| Role in Project | Project members with a specific role in a specific project. This is affected by the values set in the AM_PROJECT_MODE preference. |
| Public Schedule | Access to all users for schedules which are templates or made public. This accessor applies to the Schedule Manager application. |
| RoleInSchedule | Membership privileges of the logged in user within a particular schedule. Member privileges (accessor IDs) can be COORDINATOR , PARTICIPANT , or OBSERVER . This accessor applies to the Schedule Manager application. |
| RoleInAnySchedule | Membership privileges of the logged in user across all schedules in the system. Member privileges (accessor IDs) can be COORDINATOR , PARTICIPANT , or OBSERVER . This accessor applies to the Schedule Manager application. |
| User Excluded | The user or group is cited in a valid exclusion license attached to the object. |
| User Has Government Clearance | Compares the user's clearance with the object classification and tests whether the user has clearance above, below, or equal to that required to access the object. |

| Accessor | Description |
|--|--|
| User ITAR Licensed | User is cited in a current license associated with the selected object. |
| User ITAR Unlicensed | User is not cited in a current license associated with the selected object. |
| User Under Government Clearance | The user's clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the government clearance on the user and the government classification on the object come from a common multi-level scheme defined by the ITAR_level_list_ordering preference. |
| User Over Government Clearance | The user's clearance is over the level required by the object. This accessor is typically used to grant access and is only applicable when the government clearance on the user and the government classification on the object come from a common multi-level scheme defined by the ITAR_level_list_ordering preference. |
| User IP Licensed | User is cited in a current license associated with the selected object either directly or by membership in a cited organization (group). |
| User IP Unlicensed | User is not cited in a current license associated with the selected object. |
| User Has IP Clearance | Compares the user's clearance with the object classification and tests whether the user has clearance above, below, or equal to that required to access the object. |
| User Over IP Clearance | The user's clearance is over the level required by the object. This accessor is typically used to grant access and is only applicable when the IP clearance on the user and the IP classification on the object come from a common multi-level scheme defined by the IP_level_list_ordering preference. |
| User Under IP Clearance | The user's clearance is below the level required by the object. This accessor is typically used to revoke access and is only applicable when the IP clearance on the user and the IP classification on the object come from a common multi-level scheme defined by the IP_level_list_ordering preference. |

12.2.9 Rule tree conditions

| Condition | |
|----------------------|---|
| Has Bypass | <p>Value true or false</p> <p>Description Specifies whether the user has bypass privileges set. Bypass privilege supersedes other privileges.</p> |
| Has Class | <p>Value <i>class-name</i></p> <p>Description Specifies an object class. The object is evaluated to determine if it is of the specified class.</p> |
| Has Attribute | <p>Value <i>class:attribute=value</i></p> <p><i>class</i> is the class of the object for which you set the rule. <i>attribute</i> is an attribute of the class. Supported attribute types are string, integer, double, logical, and reference. <i>value</i> is the value for which the attribute is evaluated.</p> <p>Blank spaces are not allowed in the rule syntax. Logical values must be either 0 (false) or 1 (true). References can only be checked for a null_tag (0) or non-null (non-zero) value.</p> <p>Description Specifies an attribute and value associated with a particular class.</p> |
| Has Type | <p>Value <i>type-name</i></p> <p>Description Specifies the object type against which the object is evaluated.</p> |

| | |
|------------------------|--|
| In Job | <p>Value true or false</p> <p>Description Specifies whether the target object is in a workflow job (process). This condition does not expect an ACL attached to a rule. It is a placeholder that indicates the point at which workflow ACLs are applied in the rule tree hierarchy.</p> |
| Has Object ACL | <p>Value true or false</p> <p>Description Specifies that an ACL is associated with an object. This condition does not expect an ACL attached to a rule. It is a placeholder that indicates the point at which process ACLs and object ACLs are applied in the rule tree hierarchy.</p> |
| Has Status | <p>Value <i>status-name</i></p> <p>Accepts null entry null=all.</p> <p>Description Specifies the status type against which the object is evaluated.</p> |
| Has Description | <p>Value <i>text-string</i></p> <p>The description value can contain wildcard characters.</p> <p>Description Specifies a description for the object. The object is evaluated to determine whether the description matches this value.</p> |
| Has Name | <p>Value <i>text-string</i></p> <p>The name value can contain wildcard characters.</p> <p>Description Specifies a name against which the object is evaluated.</p> |

| | |
|---------------------------|--|
| Has Form Attribute | <p>Value</p> <p><i>form-storage-class:attribute=value</i></p> <p><i>form-storage-class</i> is the storage class for the form type on which you set the rule. <i>attribute</i> is an attribute of the form. Supported attribute types are POM_string, POM_int, and POM_double. <i>value</i> is the value for which the attribute is evaluated.</p> <p>Blank spaces are not allowed in the rule syntax.</p> <p>Description</p> <p>Enables access control of items and item revisions by setting conditions on attributes of the Masterform class. This rule can be applied to the ItemRevisionMaster form to control access to the item.</p> <p>This rule can also be used to control write access to the properties of items and item revisions, which in turn determine who can add or remove datasets associated with the item or item revision through a Specification relation.</p> <p>This rule cannot be used to control access to the datasets, and it cannot be applied to user-defined forms. It should be added below the Working→Item Revision/Item Rule rule in the rule tree.</p> |
| Has Item ID | <p>Value</p> <p><i>item-id</i></p> <p>The item ID value can contain wildcard characters.</p> <p>Description</p> <p>Specifies an item ID against which the item is evaluated.</p> |
| Is Archived | <p>Value</p> <p>true or false</p> <p>Description</p> <p>Specifies an item ID against which the item is evaluated.</p> |
| Is Local | <p>Value</p> <p>true or false</p> <p>Description</p> <p>Specifies whether the object's residence in the local database is evaluated. This condition is used when Multi-Site Collaboration is implemented.</p> |

| | |
|--------------------------|---|
| Inactive Sequence | <p>Value true or false</p> <p>Description Used in conjunction with the Inactive Sequence Objects ACL. This condition specifies that previous sequences are historical and cannot be worked on independently. The latest sequence is always the working sequence for the revision.</p> |
| Owning User | <p>Value <i>user-ID</i></p> <p>Description Evaluates whether the object is owned by the specified user.</p> |
| Owning Group | <p>Value <i>group-name</i></p> <p>Description Evaluates whether the object is owned by the group under which the user is logged on to Teamcenter. Wildcard characters can be used with the Owning Group condition to allow you to define rules applying to a group and all its subgroups. For example, assume that the Design group has two subgroups: Analysis.Design and Development.Design. By defining a value for the Owning Group condition using a wildcard, you can define a general rule to control access to all data owned by the Design group and its subgroups, for example:</p> <ul style="list-style-type: none"> □ Owning Group (*Design) -> design_group_acl |
| Owning Site | <p>Value <i>site-name</i></p> <p>Description Evaluates whether the object is owned by the specified site. This condition is used when Multi-Site Collaboration is implemented.</p> |

| | |
|----------------------------------|---|
| Owning Group Has Security | <p>Value internal or external</p> <p>Description</p> <p>Evaluates whether the owning group of the object has a security string. This condition is true only if the security value of the owning group is equal to the value of this condition.</p> |
| Is SA | <p>Value true or false</p> <p>Description</p> <p>Specifies whether the user's system administration group membership is evaluated.</p> |
| Is GA | <p>Value true or false</p> <p>Description</p> <p>Specifies whether the user's status as a group administrator in the current group is evaluated.</p> |
| In IC Context | <p>Value true or false</p> <p>Always use the true value for this condition. The false value applies the rule to all objects, regardless of whether structure edits are being made.</p> <p>Description</p> <p>Enables structure edits (occurrence edits, occurrence notes, transform edits, and attachment edits) to be controlled by the Structure Manager, Manufacturing Process Planner, Multi-Structure Manager, or Part Planner application. The rule does not depend on the properties of the object.</p> <p>When there is an active incremental change in the structure editor, the IC Context (true) condition is satisfied and its associated ACL is applied.</p> |

| | |
|---------------------------|--|
| In Project | <p>Value <i>project-ID</i></p> <p>The syntax for this rule is:</p> <pre>In Project (project-ID)-project_acl</pre> <p>Description</p> <p>Specifies a project to which the object must be assigned. The condition is evaluated as being true when the active project to which the object is assigned matches the project specified for this rule condition. If you use an empty string as the value for this condition, the condition is deemed true if the object is assigned to any active project.</p> |
| In Current Project | <p>Value <i>project-ID</i></p> <p>Description</p> <p>Specifies the project ID against which the object is evaluated. The condition is evaluated as being true when the object is in the current active project of the logged-on user, and the project ID of the current project matches the value for this condition.</p> <p>This rule is not delivered with the default installation of Teamcenter. It must be added manually.</p> |
| Is Project Member | <p>Value true or false</p> <p>Description</p> <p>Specifies whether the user's membership in the project is evaluated. This condition is only true when the user is a current member of the project.</p> |
| Is Program Member | <p>Value true or false</p> <p>Description</p> <p>Specifies whether the user's membership in the program is evaluated. This condition is only true when the user is a member of the owning program or a shared program.</p> |

| | |
|-----------------------------|--|
| In Current Program | Value true or false Description Specifies access based on whether the program to which the data is assigned is the current program under which the user is logged on to Teamcenter. |
| In Inactive Program | Value true or false Description Controls access to data based on whether the status of the owning program is inactive . |
| In Invisible Program | Value true or false Description Controls access to data based on whether the status of the owning program is invisible . |
| Is Owned By Program | Value true or false Description Controls access to data based on whether the status of the owning program is invisible . |
| | Value true or false Description Controls access to data based on whether the status of the owning program is invisible . |
| User Nationality | Value Two-character ISO 3166 codes. This condition accepts negation using a minus (-) prefix. For example, -us indicates any user not from the U.S. Description Specifies the nationality of a user. |

| | |
|--------------------------------------|--|
| Group Nationality | <p>Value Two-character ISO 3166 codes.</p> <p>This condition accepts negation using a minus (-) prefix. For example, -us indicates any user belonging to a group not from the U.S.</p> <p>Description Specifies the nationality of a group or organization.</p> |
| User Location | <p>Value Two-character ISO 3166 codes.</p> <p>This condition accepts negation using a minus (-) prefix. For example, -us indicates any user located outside the U.S.</p> <p>Description Specifies the location of the user.</p> |
| Site Location | <p>Value Two-character ISO 3166 codes.</p> <p>This condition accepts negation using a minus (-) prefix. For example, -us indicates any user at a site outside the U.S.</p> <p>Description Specifies the location of the site.</p> |
| User Is ITAR Licensed | <p>Value true or false</p> <p>Description Verifies the existence of a valid ITAR license that names the current user as a licensee.</p> |
| Has Government Classification | <p>Value Specific government classification attribute values that can be prefixed by the following operators:</p> <p>>, >=, <, <=, =</p> <p>Description Validates the government classification attribute value of the object against the value specified for the condition.</p> <p>The operators can be used without a clearance value in which case the government classification attribute</p> |

| | |
|---|--|
| | <p>of the object is compared to the user's clearance level based on the specified operator.</p> <p>If the object has no government classification attribute value, this rule does not apply.</p> |
| Has No Government Classification | <p>Value</p> <p>Description</p> <p>Matches if the object has a null value for the government classification attribute.</p> |
| User Has Government Clearance | <p>Value</p> <p>Specific government clearance attribute values that can be prefixed by the following operators:</p> <p>>, >=, <, <=, =</p> <p>Description</p> <p>Validates the user's clearance level against the value specified for the condition.</p> <p>The operators can be used without a clearance value in which case the user's clearance is compared to the government classification attribute of the object based on the specified operator.</p> <p>If no value is supplied, the user must have a clearance value set.</p> |
| User Is Excluded | <p>Value</p> <p>true or false</p> <p>Description</p> <p>Specifies whether the user or group is cited by a valid exclusion license.</p> |
| User Has IP Clearance | <p>Value</p> <p>Specific clearance values that can be prefixed by the following operators:</p> <p>>, >=, <, <=, =</p> <p>Description</p> <p>Validates the user's clearance level against the value specified for the condition.</p> <p>The operators can be used without a clearance value in which case the user's clearance is compared to the IP classification attribute of the object based on the specified operator.</p> |

| | |
|-------------------------------------|--|
| | If the data is not IP classified, the User Has IP Clearance condition is evaluated as being true regardless of whether or not the user is assigned a clearance level. |
| Object Has IP Classification | <p>Value</p> <p>Specific IP classification attribute values that can be prefixed by the following operators:</p> <p>>, >=, <, <=, =</p> <p>Description</p> <p>Validates the IP classification attribute value of the object against the value specified for the condition.</p> <p>The operators can be used without a clearance value; the IP classification attribute of the object is compared to the user's clearance level based on the specified operator.</p> <p>If the object has no IP classification attribute value, this rule does not apply.</p> |
| Has No IP Classified | <p>Value</p> <p>Description</p> <p>Matches if the object has a null value for the IP classification attribute.</p> |
| User Is IP Licensed | <p>Value</p> <p>true or false</p> <p>Description</p> <p>If set to true verifies the existence of a valid (not expired) IP license that names the current user or their group as a licensee.</p> |

12.2.10 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.

| AccessorType | 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 |
|----------------------|------|------|-------|--------|--------|---------|--------|------|
| Owning User | | | ✓ | ✓ | ✓ | | | |
| Group Administrator | | | | ✓ | ✓ | | | |
| Owning Group | | | ✓ | | | | | |
| System Administrator | | | | ✓ | ✓ | | | |
| World | | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ |

Result: After all the rules are evaluated, the following is the result. Note that the **Working** ACL grants the owning group write permission, but the **UGMASTER** ACL already removed that privilege. The figure also shows the applied named ACL.

| Accessor | User | Read | Write | Delete | Change | Promote | Demote | Copy | Named ACL |
|----------------------|-------------------|------|-------|--------|--------|---------|--------|------|----------------|
| World | | | | | | | | | Import /Export |
| Remote Site | | | | | | | | | Import/ Export |
| Role in Owning Group | Designer | | ✓ | | | | | | UGMASTER |
| Owning Group | | | ✗ | | | | | | UGMASTER |
| Owning User | | ✓ | ✓ | ✓ | | | | | Working |
| User | tsproxy (tsproxy) | | ✓ | ✓ | | | | | Working |
| Group Administrator | | | | ✓ | ✓ | | | | Working |
| Owning Group | | | ✓ | | | | | | Working |
| System Administrator | | | | ✓ | ✓ | | | | Working |
| World | | ✓ | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | Working |

12.3 Understanding the rule creation process

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.

12.3.1 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.

12.3.2 Create an access control list (ACL)

1. In the **Named ACL** section of the Access Manager, enter the ACL name in the **ACL Name** box.



2. Click **Create** to the right of the **ACL Name** box.
3. Click **Save** 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. (Optional) Click **Localization** to display the **Language Translations** dialog box and set localized values for the ACL.
9. Click **Save** .

12.3.3 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.

12.3.4 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.

12.3.5 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.

12.3.6 Good rule 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.

- **Use the Has Property condition to create custom rules based on the value of compound properties.**

For example:

```
Item:my_custom_prop=my_custom_prop_value
```

In this example, **Item** is the type name and **my_custom_prop** is the compound property.

- **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.

- **Do not create GRM relations**

Do not create Generic Relationship Management (GRM) relationships between Teamcenter business objects, such as BOM View, and Access Manager objects, such as AM Tree, Named ACL, and AM_ACE. Creating such relationships can result in unpredictable behavior with Access Manager during run time.

12.3.7 Cautions for using rule trees

- **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. Modifying the **Systems Objects** branch of the rule tree is not supported unless specifically advised to do so by Siemens PLM Software.

- **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.

12.4 About verifying the effect of access rules

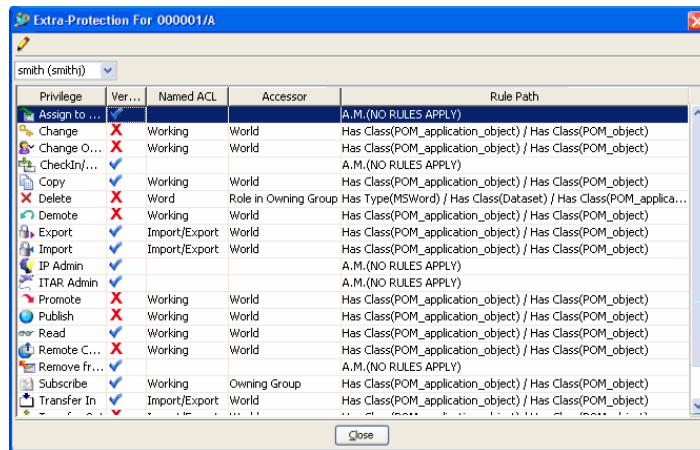
After you implement access rules, verify that the rules produce the desired privileges for different types of accessors. You can do this by viewing the access privileges in My Teamcenter. You can also determine which rules result in a privilege being granted or denied by viewing the verdicts in the **Extra Protection** dialog box.

In addition, you can view performance statistics.

12.5 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.



12.6 About 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 only be modified by one user at any given time. This behavior is ensured by implicit and explicit checkout.

12.6.1 About configuring access to working data

Access to working data is configured by adding rules to the **Has Class(POM_application_object) -> Working** branch of the Access Manager rule tree. The **Working** ACL, which is delivered as part of your Teamcenter installation, configures access to working data of all object types, as shown in the following table.

The **Working** ACL specifies the privileges for the following accessors:

- World

Grants all users (**World**) read and copy privileges to all data and denies write, delete, and other privileges to all other users. This is very important, because it prevents those privileges from being granted by rules elsewhere in the rule tree.

- **Owning User**

Grants write and delete privileges to all data that the user creates. In addition, the ACL grants change privilege, which allows the owning user to create object ACLs for data they own. This is a significant privilege and is often granted only to managers rather than to general users or groups of users.

- **Owning Group**

Grants write privileges to any user who is in the group that owns the data.

- **System Administrator**

Grants delete and change privileges to administrators.

12.6.2 Guidelines for applying the delete and change privileges

The following guidelines should be considered when applying the delete and change privileges:

- **Delete**

Delete privilege is generally limited to the owning user. However, you can also grant delete privileges to group administrators and system administrators for the purpose of maintaining the database.

Tip

You can establish a folder in which users can place data that they want deleted, and an administrator can be assigned responsibility for deleting the data.

- **Change**

Caution must be used when granting change privileges. Change privilege allows accessors to define object ACLs that take precedence over rules in the Access Manager rule tree. Therefore, you can subvert the access rules by creating an object ACL granting write privileges to the **World** accessor. For this reason, the **Has Object ACL** condition is placed lower in the rule tree than the **In Job** and **Has Status** conditions.

If change privileges are granted to the owning user, restrictions on change privileges can be applied for specific data types lower in the tree.

12.6.3 Controlling access to revision rules

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

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

Access Manager rule:

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

Private revision rule ACL:

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

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

Public revision rule ACL:

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

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

12.7

About 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 access control list (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

If you associate ACLs directly with the **In Job** condition, they 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 named ACL to use while the associated task handler is current or started. For example, if this handler is placed on the **Start** action of a **Review** task, when the task starts, the named ACL specified in the handler's argument is the ACL used by Access Manager to determine access rights for the target objects of the workflow process. The ACL is applied to the task and all subsequent tasks in the workflow process unless it is changed by another instance of the **EPM-set-rule-based-protection** handler or the process completes.

If the **EPM-set-rule-based-protection** handler is not defined, the system uses the workflow ACL. The current workflow ACL stays in effect until the same workflow sets another ACL later in the process or the process completes.

Workflow ACLs are created in the Workflow Designer application within the context of a specific task and are considered an attribute of the task.

12.7.1 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).

Note

This accessor is used only 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.

Note

This accessor is used only in a workflow ACL.

- **Approver (Group)**

Users in a specific group who are members of a sign-off team in a workflow process.

Note

This accessor is used only in a workflow ACL.

- **Approver**

Users who are members of a sign-off team in a workflow process regardless of their role and group.

Note

This accessor is used only in a workflow ACL.

- **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.

12.7.2 Workflow ACL example

The **ApprovalACL** ACL grants privileges to sign-off team members using the **World** accessor (all users) and the **Approver(RIG)** accessor.

| Approver(RIG) | Engineering Manager in high_performance | | | | | | | | | | | |
|---------------|---|--|--|--|--|--|--|--|--|--|--|--|
| World | | | | | | | | | | | | |

The **World** accessor is explicitly granted read and copy privileges and is explicitly denied write, delete, change, promote, demote, and change ownership privileges.

The **Approver(RIG)** (Engineering Manager in the **high_performance** group) is explicitly granted promote privileges.

In addition, the privileges set for the **World** accessor (with the exception of the promote privilege) are implicitly inherited by the **Approver(RIG)** accessor.

12.7.3 Parallel task and parallel process ACL conflict resolution

When multiple workflows set named ACLs concurrently, the logical **OR** applies to the competing workflow ACLs. To determine privileges allowed to a user of an object in process, the system uses a simplified processing scheme.

All ACLs associated with the object in the workflow process are taken into account.

- When one ACL grants a privilege, access is granted.
- When no ACL grants a privilege, but one or more ACLs denies it, access is denied.
- When no ACL grants or denies the privilege, access is neither granted nor denied.

12.7.4 Workflow access examples

The following examples illustrate possible scenarios in which two tasks compete to apply privileges to the same target object.

Scenario 1

The user is a member of **Group B**.

The **Task 1** named ACL grants read privileges to **Group A**.

The **Task 2** named ACL grants read and write privileges to **Group B**.

Result: The user is granted read and write privileges.

Scenario 2

The user is a member of **Group B**.

The **Task 1** named ACL grants read privileges to **Group B**.

The **Task 2** named ACL grants read and write privileges to **Group B**.

Result: The user is granted read and write privileges.

Scenario 3

The user is an approver on **Task 2**.

The **Task 1** named ACL grants read privileges to the **Approver** accessor.

The **Task 2** named ACL grants read and write privileges to the **Approver** accessor.

Result: The user is granted read and write privileges.

Scenario 4

The user is an approver on **Task 1** and **Task 2**.

The **Task 1** named ACL grants read privileges to the **Approver** accessor.

The **Task 2** named ACL grants read and write privileges to the **Approver** accessor.

Result: The user is granted read and write privileges.

Scenario 5

The user is the responsible party on **Task 2**.

| | | |
|---|--|---|
| The Task 1 named ACL grants read privileges to the Approver accessor. | The Task 2 named ACL grants read and write privileges to the Responsible Party accessor. | Result: The user is granted read and write privileges. |
|---|--|---|

Scenario 6

The user is the responsible party on **Task 1** and an approver on **Task 2**.

| | | |
|--|---|---|
| The Task 1 named ACL grants read privileges to the Responsible Party accessor. | The Task 2 named ACL grants read and write privileges to the Approver accessor. | Result: The user is granted read and write privileges. |
|--|---|---|

Scenario 7

The user is the responsible party on both tasks.

| | | |
|--|--|---|
| The Task 1 named ACL grants read privileges to the Responsible Party accessor. | The Task 2 named ACL grants read and write privileges to the Responsible Party accessor. | Result: The user is granted read and write privileges. |
|--|--|---|

Scenario 8

The user is a member of the **task_owner_group** group on **Task 1** and is a member of the **approver_group** group on **Task 2**.

| | | |
|--|--|---|
| The Task 1 named ACL grants read privileges to the task_owner_group group. | The Task 2 named ACL grants read and write privileges to the approver_group group. | Result: The user is granted read and write privileges. |
|--|--|---|

12.8

About configuring group-level security

Groups represent collective bodies of users (group members) who share data. Group members are assigned functional roles within a group. Users can be assigned multiple roles within a group and they can also be members of multiple groups. Groups and users roles within groups can be used as the basis for granting access to data in Teamcenter.

Groups can be arranged in a hierarchy, which provides a powerful way of defining access rules for high-level groups that are then implicitly inherited by groups that are lower in the hierarchy. However, if access is explicitly defined for a lower level group that is also subject to implicitly inherited access rules, the explicitly defined access rules override the implicitly inherited rules for that group.

Note

Group inheritance does not apply to the **Owning Group** accessor type.

To better understand the examples, you should first understand the rule conditions, access control lists (ACLs), and accessors that comprise the Access Manager rules, as well as the groups and their specified security levels that are used in the examples.

| Rule condition | Description |
|----------------------------------|---|
| Owning Group Has Security | Evaluates whether the owning group of the object has a security string. This condition is true only if the security value of the owning group is equal to the value of this condition. |
| Owning Group | Evaluates whether the object is owned by the group specified in the group-name argument of the condition. |
| | Wildcard characters can be used with the Owning Group condition to allow you to define rules applying to a group and all its subgroups. For example, assume that the Design group has two subgroups: Analysis.Design and Development.Design . By defining a value for the Owning Group condition using a wildcard, you can define a general rule to control privileges to all data owned by the Design group and its subgroups. |

Note

Subgroup names, when displayed out of the context of the tree structure,

| Rule condition | Description |
|----------------|--|
| | are formatted with the lowest group in the hierarchy listed first and the highest group listed last. For example, the subgroup name Analysis.Design indicates that the Design group is a parent of the Analysis subgroup. |

For example:

- ❑ Owning Group("Design") → design_group_acl

| ACL | Description |
|---------------------------|---|
| internal_group_acl | Grants read and modify privileges to the group that owns the data, grants read privileges to other internal groups, and denies privileges to external groups. |

| 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). |

| Groups | Security level |
|----------------------|----------------|
| Design | Internal |
| Development | Internal |
| Manufacturing | Internal |
| Supplier 1 | External |
| Supplier 2 | External |

12.8.1 Example of 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 access is 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 rule-based security).

| Owning Group | | | |
|----------------------|----------|--|--|
| Groups with Security | Internal | | |
| Groups with Security | External | | |

12.8.2 Example of 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, and access is 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:

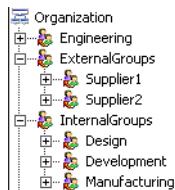
↳ **Owning Group Has Security (“External”) -> external_group_acl**

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 | | |

12.8.3 Example of configuring supplier security using hierarchical groups

The security implemented in [Example of configuring security to prevent suppliers from viewing internal data](#) and [Example of configuring security for data owned by a supplier \(external data\)](#) can also be implemented by using the **Owning Group** condition to define a security rule granting all internal users read privileges to data owned by internal groups. To do this, you must first create a hierarchical group structure (using the Organization application) with one root group containing all internal groups and another root group containing all external groups.



Based on this hierarchical group structure, you can grant all internal groups read privileges to company-owned data and deny supplier groups read privileges to company-owned data by writing the following rule:

↳ Owning Group("*InternalGroups") → group_read_acl

| Owning Group | | | | | | | | | | | | |
|--------------|----------------|--|--|--|--|--|--|--|--|--|--|--|
| Group | InternalGroups | | | | | | | | | | | |
| Group | ExternalGroups | | | | | | | | | | | |

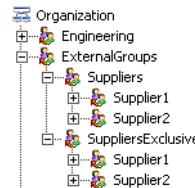
You can set the **Owning Group** condition to define a security rule that grants all external users read privileges to data owned by external groups. Again, this method requires the appropriate hierarchical group structure.

↳ Owning Group("*ExternalGroups") → suppliers_acl

| Owning Group | | | | | | | | | | | | |
|--------------|----------------|--|--|--|--|--|--|--|--|--|--|--|
| Group | InternalGroups | | | | | | | | | | | |
| Group | ExternalGroups | | | | | | | | | | | |

12.8.4 Example of configuring security for special project data using hierarchical groups (fully restrictive external group security)

In this example, ABC Part Company wants to implement a strict security rule to protect some of the data owned by one of their suppliers. To do this, they extended the hierarchical group structure.



Note

Subgroup names, when displayed out of the context of the tree structure, are formatted with the lowest group in the hierarchy listed first and the highest group listed last. For example, the subgroup name **Analysis.Design** indicates that the **Design** group is a parent of the **Analysis** subgroup.

Based on this hierarchical group structure, the following rule can be defined to prevent access to supplier data by users who do not belong to the specific supplier group:

❑ **Owning Group("*.SuppliersExclusive.*") -> suppliers_exclusive_acl**

Note

To make the restricted security of this rule work effectively, the supplier group must only transfer existing data and create new data when they are logged on as a member of one of the subgroups of the **SuppliersExclusive** group.

The **suppliers_exclusive_acl** ACL has the following definition.

| Owning Group | | | | | | | | | | | |
|--------------|----------------|--|--|--|--|--|--|--|--|--|--|
| Group | InternalGroups | | | | | | | | | | |
| Group | ExternalGroups | | | | | | | | | | |

Tip

The security implemented in this example can also be achieved by using the **In Project** condition and adding the data to be protected to a special project.

12.9

About configuring security for project and program data

Project-level security refers to a security scheme based on a combination of projects and access rules in Teamcenter. Projects are entities that correlate groups of users, potentially at different physical sites, with the data associated with a given project or subset of a project.

Program-level security refers to a security scheme based on a combination of projects that are configured to use program security and the corresponding program access rules in Teamcenter.

12.9.1 Basic concepts about projects

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 or regular team members who are explicitly granted **ASSIGN_TO_PROJECT** or **REMOVE_FROM_PROJECT** privileges can assign data to and remove data from projects.
- Project and program names must be unique within your site. Projects and programs cannot have the same name as any group at the site.
- Data can be assigned to or removed from projects manually or automatically 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.

12.9.2 What are groups?

Groups organize users and play an important role in access control for projects and programs. The following concepts apply to groups:

- Access to data is generally determined based on the owning group.
- Groups can be categorized as being **Internal** or **External**. This allows you to differentiate between internal users and suppliers when granting data access. It also allows you to protect supplier data from being accessed by unauthorized internal users or by other suppliers.
- Groups can be structured in a hierarchy that allows access controls to be inherited by subgroups, because members of subgroups are implicitly members of the parent group on which the access controls are implemented.

Note

Groups are defined and maintained using the Organization application.

12.9.3 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 groups of users based on the group's categorization as internal (OEM) or external (supplier) by applying the **Owning Group Has Security** condition.
- Grant access to data assigned to projects by applying the **In Project** condition.

Note

This rule is applied by default to any object assigned to an active project.

- Grant or deny access to users based on their membership in a project by applying the **Is Project Member** condition.

12.9.3.1 Preferences related to project and program security

The preferences in the following table affect the way that security rules are evaluated for data in projects and programs.

| Preference | Description |
|---------------------------------------|--|
| AM_PROJECT_MODE | Determines whether the system evaluates roles in all active projects or whether only the role in the user's current project is evaluated. |
| TC_project_validate_conditions | Determines how the Assign to project and Remove from project access privileges are validated in conjunction with privileged membership validation. |

12.9.4 Access rules for projects and programs

The Access Manager rule tree delivered as part of the standard Teamcenter installation includes the following rules related to programs and projects:

- **In Current Program(false) -> Not Current Program**
- **In Inactive Program(true) -> Inactive Program**
- **Is Program Member(true) -> Not Program Member**
- **In Invisible Program(true) -> Invisible Program**
- **In Project() -> Projects**
- **Is Project Member(true) -> Project Objects**
- **Is Owned By Program() -> Projects**

12.9.5 Project-level security based on groups

Project-level security is more flexible than group-level security, because data can be added to and removed from projects without requiring the data properties to be changed or the ownership of the data to be transferred from one group to another. However, projects can be used in conjunction with groups when developing a security solution. Access controls based on projects and groups can be applied in a Multi-Site Collaboration environment.

Typically, three Access Manager rule conditions are applied to grouped data when configuring project-level security:

- **Owning Group(*group-name*)**

Defines security on data based on group ownership and hierarchical group behavior, which dictates that subgroups inherit the access controls defined for parent groups.

- **Owning Group Has Security(*group-name*)**

Defines security on data based on ownership by groups with a specified security categorization, either **Internal** or **External**.

- **In Project(*project-id*)**

Defines security on the data owned by multiple groups with different security categories.

12.9.6 Placement of rules in the Access Manager rule tree

The **In Project()** access rule is typically placed above the **Owning Group()** rule or **Owning Group Has Security()** rule in the Access Manager rule tree. This places a higher level of precedence on access based on the project than on access based data ownership.

12.9.7 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.

12.9.8 Implementation considerations for program-level security

You must consider the following points when implementing program-level security:

- Auto assignment of objects to projects.
- Program context.
- Placement of rules in the Access Manager tree.
- Interaction of program and project rules.
- Importing and exporting project data in a Multi-Site Collaboration environment.

12.10 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

12.11 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.
- Control access to working and in-process data.
- Configure project-level security to control access to data in specific projects.

Lesson

13 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.
- Use Teamcenter utilities for projects.

Help topics

Additional information for this lesson can be found in:

- *Project Guide*

13.1 Introduction to Project

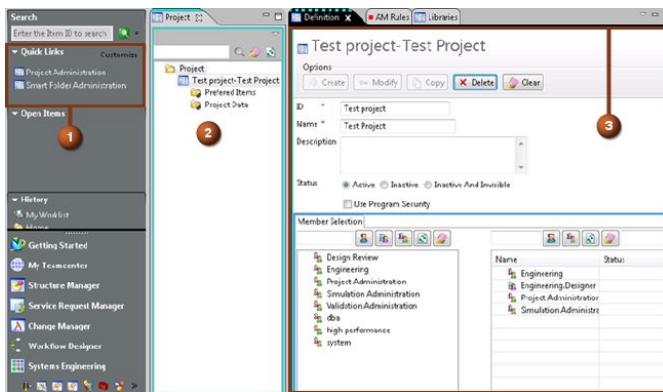
The Project application provides a mechanism for organizing data and implementing access control based on project membership.

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.
- Associate data dictionaries (libraries) with projects.

Project works with Access Manager to control access to data by project members.

13.1.1 Project Administration window



1 Quick Links

Displays links to navigate between the **Project Administration** and **Smart Folder Administration** windows.

2 Project tree

Displays the list of projects in Teamcenter. New projects and programs appear in this list.

3 Project definition and rules

Displays information about the selected project.

If the **Definition** tab is selected, the properties of the project are displayed. If the **AM Rules** tab is selected, the security rules for the project are displayed. If the **Libraries** tab is selected, you can view which data dictionaries are assigned to the project or assign new dictionaries to the project.

13.1.2 Project administration buttons

| Button | Description |
|--|---|
|  Groups | Filters the display of the organization tree or project member tree by groups. You can search for a specific group in either tree by entering the group name in the text box and clicking the  Groups button. |
|  Roles | Filters the display of the organization tree or project member tree by roles. You can search for a specific role in either tree by entering the role name in the text box and clicking the  Roles button. |
|  Users | Filters the display of the organization tree or project member tree by users. You can search for a specific user in either tree by entering the user name in the text box and clicking the  Users button. |
|  Clear | Clears the search criteria and restores the organization tree or project member tree to its previous state. |
|  Refresh tree | Refreshes the organization tree or project member tree and restores it to its original state. |

13.1.3 Project administration tabs

| Tab | Description |
|-------------------|---|
| Definition | Displays the 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. |
| Libraries | <p>Note</p> <p>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.</p> <p>Displays data dictionaries (libraries) that have been defined in the Classification Admin application, and provides tools for associating these libraries with specific projects or programs.</p> |

13.1.4 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 Administration** window.

13.1.5 Basic concepts about projects

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 or regular team members who are explicitly granted **ASSIGN_TO_PROJECT** or **REMOVE_FROM_PROJECT** privileges can assign data to and remove data from projects.
- Project and program names must be unique within your site. Projects and programs cannot have the same name as any group at the site.
- Data can be assigned to or removed from projects manually or automatically 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.

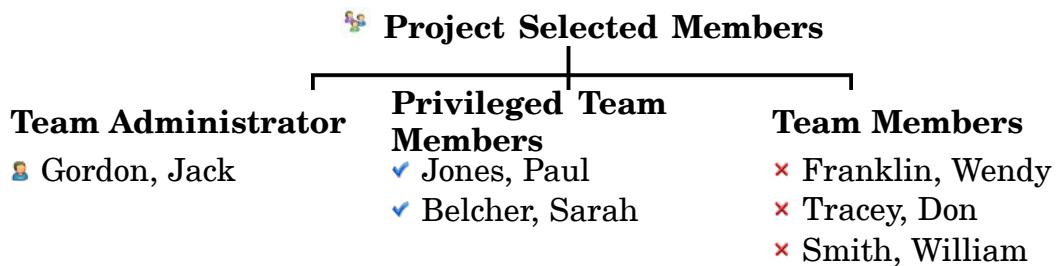
13.1.6 Project administrators and team members

The following table describes the project administrators and team members who can create, manage, and use projects.

| Team role | Definition |
|--|--|
| Project Administrator role in Project Administration group | <p>Teamcenter user with privileges to create and administer projects.</p> <p>Users in the Project Administrator role 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. |
| Team Administrator role in Project Administration group | <p>Project team members with privileges to modify project information. These privileges apply to the project metadata, not to the data assigned to projects.</p> <p>Users in the Team Administrator role 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. |
| Privileged team members | Project team members with privileges to assign or remove objects from their projects. |

| Team role | Definition |
|--------------|---|
| Team members | Team members (users) who do not have privileges to assign objects to or remove objects from their projects. |

The following diagram illustrates a typical project hierarchy.



13.2 Using projects

Project creation and administration tasks are performed in the following order:

1. A project administrator is added to the **Project Administrator** role in 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 team administrators.
3. The user's default project is set.

Note

To assign a default project, you must select the **Use Program Security** option when you define the project or program.

4. As database objects are created, they are assigned to the project automatically or manually by privileged team members.

13.2.1 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 groups of users based on the group's categorization as internal (OEM) or external (supplier) by applying the **Owning Group Has Security** condition.
- Grant access to data assigned to projects by applying the **In Project** condition.

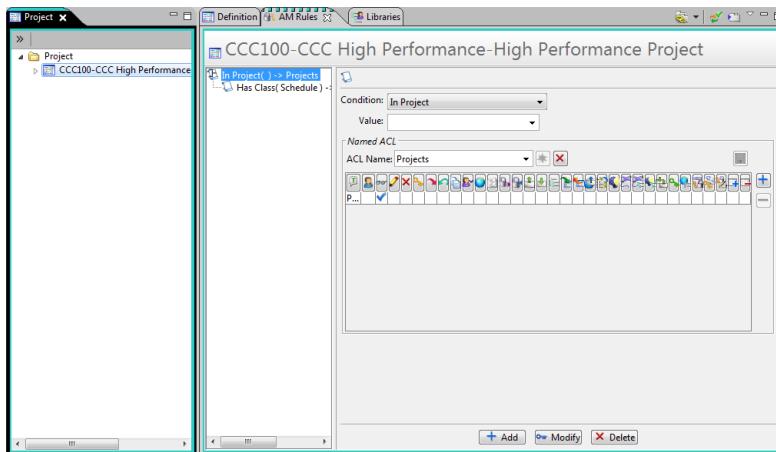
Note

This rule is applied by default to any object assigned to an active project.

- Grant or deny access to users based on their membership in a project by applying the **Is Project Member** condition.

13.2.2 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.

13.2.3 Activating and deactivating projects

Projects have one of the following statuses that control access to the data assigned to the projects:

Active

Project data is visible and can be modified by project team members.

Inactive

Project data is visible to project team members, but the data cannot be modified.

Inactive and Invisible

Project data is invisible and cannot be modified.

Activate or deactivate project

1. Expand the **Project** tree and select the project to be deactivated.
2. Click the **Definition** tab.

Tip

If the **Definition** tab is not displayed, click the **Project Administration** quick link.

- To activate, select **Active** in the **Status** section.
 - To deactivate, select either **Inactive** or **Inactive and Invisible**.
3. Click **Modify**.

13.2.4 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 team administrator.
 - Assigning or removing privileged team member status.
 - Changing access rules.
3. Click **Modify**.

Modifications to the project information are saved to the database.

Note

Siemens PLM Software does not recommend changing the project name or ID after data has been assigned to an active project.

13.3 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 the following ways:

- Manually assigned to projects by users who are designated as privileged team members. Privileged team members include:
 - 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** preference.

- Automatically assigned to projects when the object is created. When a new item revision is created, it is automatically assigned to the project in 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.
- Using the **update_project_bom** utility to assign or remove objects in a product or manufacturing structure to or from a project.

13.4 Teamcenter utilities for projects

This topic provides an overview of Teamcenter utilities for projects. You can use the following utilities to create projects and update project data.

- **`create_project`**
- **`update_project_data`**
- **`update_project_bom`**

create_project

DESCRIPTION

Creates projects in the database based on command line input or input from a text file.

SYNTAX

```
create_project -u=user-id -p=password -g=group
{-id=project-id -name=project-name
[-desc=project-description -status=A | I
-teams=group1~role1~user1~group2~role2~user2...
[-privileged=group1~role1~user1~group2~role2~user2... ]
| [-teamadmin=group~role~group]}
{-input=full-path-to-input-file [-delimiter=delimiter-character]} -h
```

ARGUMENTS**-u -p -g**

Specifies the administrative user, password, and group.

-id

Specifies the ID of the project.

-name

Specifies the name of the project.

-desc

Specifies the project description.

-status

Specifies the status; either active (**A**) or inactive (**I**).

-teams

Specifies group members to be on the project team. This argument accepts valid user, role, and group names. Use the tilde character (~) as a delimiter when specifying group, role, and user, as follows:

```
-teams=group1~role1~user1~group2~role2~user2...
```

In addition, you can specify all members in a group as follows:

```
-teams=group1~*~*
```

-privileged

Defines privileged group members using the following format:

```
-privileged=group1~role1~user1...
```

-teamadmin

Defines the team administrator using the following format:

```
-teamadmin=group~role~user
```

If not specified, the default team administrator is the logged-on user who is running the utility.

-input

Specifies the path to a text file containing multiple entries of project id, project name, project description, teams, and privileged members. Use this option to create multiple projects.

The syntax of the input file is as follows:

-delimiter

Specifies the delimiting character used in the input file to parse id, name, description, status, and teams.

EXAMPLES

- To create a project with ID **123456**, named **ABC Car 123 Model**, description **A high end version of ABC Car**, with an active status assigned to **Car 1** and **Car 2** groups, enter the following command on a single line:

```
create_project -u=user-id -p=password -g=group-name -id=123456  
-name="ABC Car 123 Model" -desc="A high end version of ABC  
Car" -status=A -teams="Car 1"~Designer~Smith
```

- To create projects from an input file, enter the following command on a single line:

```
create_project  
-u=user-id -p=password -g=group-name -input=/tmp/project_input_file.txt
```

- The syntax of the input file is as follows:

```
id|name|desc|A or I|group1~role1~user1~group2~role2~user2...|  
group1~role1~user1  
id|name|desc|A or I\group1~role1~user1~group2~role2~user2...|  
group1~role1~user1
```

update_project_data

DESCRIPTION

Updates project data in the Teamcenter database.

Note

The **-f=update** option initiates the update of all project-related data in a Teamcenter database. This process can take a long time depending on the number of objects assigned to projects. When the project ID (**-pid**) is specified for one or more projects, the action applies only the given projects. When updating specific projects, other project-related data may also need to be updated by running the utility again.

SYNTAX

```
update_project_data [-u=user-id {-p=password |
-pf=password-file} -g=group ]
[ -f=function ] [ -force ] [ -t=relation-type-name1[,relation-type-name2...]
] [ -pid=project-ID1[, project-ID2...] ] [-h]
```

ARGUMENTS

-u -p -g

Specifies the administrative user, password, and group.

-f=

Specifies the function performed by the utility. Must be one of the following options:

update

Updates project-related data and is generally used after site propagation rules are modified. This function can also be used to cleanse project-related data that may have been corrupted by system crashes.

This is the default function for this utility.

list

Lists relation types used for site propagation rules. It also lists the IDs, project administrator, and status of projects defined in the Teamcenter database.

add

Adds relation types of the site propagation rules for project assignment propagation and updates project data to reflect the change in the rules. The relation types are given as a comma-separated string.

remove

Removes relation types of the site propagation rules for project assignment propagation and updates all project data to reflect the change in the site

propagation rules. The relation types are given as a comma-separated string.

delete

Deletes one or more projects identified by the **-pid** option.

bomviewon

Enables BOM view propagation. This allows BOM views and BOM view revisions of an item to be added to the project automatically when the item is added to a project.

bomviewoff

Disables BOM view propagation. By disabling BOM view propagation, BOM views and BOM view revisions of an item are not included in the project by default when the item is added to the project.

-h

Displays help for this utility.

EXAMPLES

The following examples illustrate the use of the **update** function:

- Enter the following command to unconditionally update the specified project in the Teamcenter database using the current site propagation rules:
`-f=update -pid=project-id`

- Enter the following command to unconditionally update the specified list of projects using the current site propagation rules:
`-f=update -pid=project-id1[,project-id2...]`

The project IDs are given as a comma-separated string. For example, **-pid="Proj4000,Proj5000"** specifies that the action is performed on two projects: **Proj4000** and **Proj5000**.

- Enter the following command to update all projects in the Teamcenter database using the current site propagation rules:
`-f=update`

This command is normally used to update all project data after site propagation rules have been modified. The update algorithm updates project data for objects with a last project assignment date prior to the last site propagation rule modification date.

- Enter the following command to unconditionally update all projects in the Teamcenter database using the current site propagation rules:
`-f=update -force`

Note

If the database contains a large number of projects, processing time could be considerable.

The following example illustrates the use of the **list** function:

- Enter the following command to list the project administrator, project status, and relation types used for site propagation rules:

```
-f=list
```

Note

This function is not used with other parameters.

The following examples illustrate the use of the **add** function:

- Enter the following command to append relation types to the site propagation rules and update all project data in the Teamcenter database:

```
-f=add -t=relation-type-name1[,relation-type-name2...]
```

- Enter the following command to append relation types to the site propagation rules and update the project data related to the projects specified by the **-pid** option:

```
-f=add -t=relation-type-name1[,relation-type-name2...]  
-pid=project-id1[,project-id2...]
```

This command is used under special circumstances when a user wants to update specific projects prior to updating the entire database. The database must be updated after the specific projects have been updated to ensure completeness of the project data. To update the remaining project data, run the utility using the **-update** option.

The following examples illustrate the use of the **remove** function:

- Enter the following command to remove relation types from the site propagation rules and update all project data in the database:

```
-f=remove -t=relation-type-name1[,relation-type-name2...]
```

- Enter the following command to remove relation types from the site propagation rules for specific projects:

```
-f=remove -t=relation-type-name1[,relation-type-name2...]  
-pid=project-id1[,project-id2...]
```

This command is used under special circumstances when a user wants to update specific projects prior to updating the entire database. The database must be updated after the specific projects have been updated to ensure completeness of the project data. To update the remaining project data, run the utility using the **-update** option.

The following example illustrates the use of the **delete** function:

- Enter the following command to delete specific projects and remove all project data associated with those projects:

```
-f=delete -pid=project-id1[,project-id2...]
```

Note

Processing time can be considerable depending on the number of objects in each project.

update_project_bom

DESCRIPTION

Allows you to update all items in a BOM structure within specified projects.

SYNTAX

```
update_project_bom -u=user-id -p=password -g=group
[-f={add | remove}] [-type={item | rev}] [-item=item-id | -key=key-id]
[-rev_id=revision-id]
] [-rev_rule=revision-rule] [-unit_no=unit-number] [-date=date]
[-end_item=end-item-id] [-var_rule=variant-rule] [-depth=depth-of-bom]
[-level={ 1 | 2 } ] [-projects=project-lists] [-h]
```

ARGUMENTS

-u -p -g

Specifies the administrative user, password, and group.

-f

Specifies one of the following types of operation for this utility:

Note

If this argument is omitted, the default action is to add items to projects.

add

Adds item objects to projects.

remove

Removes item objects from projects.

-type

Specifies one of the following types to be updated for each BOM line:

item

Updates each child item for the projects.

rev

Updates only child item revision objects to the projects.

Note

If this argument is omitted, the default type is **item**.

-item

Specifies the ID of the root item of the BOM to update.

-key

Specifies the key of the root item of the BOM to update.

-rev_id

Specifies the ID of the root item revision. If omitted, the default is the latest revision.

-rev_rule

Specifies the configuration rule to be applied to the item revision. If omitted, the default is the **Latest Working** revision rule.

-unit_no

Specifies the unit number associated with the revision rule.

-date

Specifies the effectivity date associated with the revision rule. The date should be specified in the following format:

yyyy MM dd hh mm ss

-end_item

Specifies the ID of the end item associated with the revision rule.

-var_rule

Specifies the variant rule to be applied to the BOM structure. If omitted, no variant rule is applied.

-depth

Specifies to what depth the BOM is traversed. If omitted, the entire BOM structure is traversed.

-level

Indicates the level of propagation. Specify either **1** or **2**.

-projects

Lists the projects to which the BOM structure will be added or from which the BOM structure will be removed.

When the **-add** option is specified, all items in the BOM structure are added to these projects.

When the **-remove** option is specified, all items currently belonging to the projects are removed from the projects. You can specify more than one project. If there is more than one project in the list, each project name is separated by a comma (,).

-h

Displays help for this utility.

EXAMPLES

- To display usage help for this utility, enter the following command on a single line:

```
update_project_bom -h
```

- To traverse a BOM structure with the top-level item **ABC001**, item revision **001**, and revision rule **Latest Working**, and to find all child items and add these items to the following three projects: **CusProj1**, **CusProj2**, and **CusProj3**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=add -item=ABC001 -rev_id=001 -rev_rule="Latest Working"
-projects=CusProj1,CusProj2,CusProj3
```

- To traverse a BOM structure with the top-level item **ABC001**, item revision **001**, and revision rule **Latest Working**, and to find all the child revision items and add only these revision items to projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=add -type=rev -item=ABC001 -rev_id=001 -rev_rule="Latest Working"
-projects=CusProj1,CusProj2
```

- To traverse the BOM structure starting at the top-level item **ABC001** with item revision **001** by applying the revision rule **Latest Released** and variant rule **AlphaRelease**, and find all the child revision items and add only these revision items to the projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=add -type=rev -item=ABC001 -rev_id=001 -rev_rule="Latest Released"
-var_rule="AlphaRelease" -projects=CusProj1,CusProj2
```

- To traverse the BOM structure of the top-level item **ABC002**, item revision **001**, and revision rule **Latest Working** and find all the child items and remove these items from projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=remove -item=ABC002 -rev_id=001 -rev_rule="Latest Working"
-projects=CusProj1,CusProj2
```

- To traverse the BOM structure of the top-level item **ABC002**, item revision **001**, and revision rule **Latest working** and find all the child revision items and remove only these revision items from projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=remove -type=rev -item=ABC002 -rev_id=001
-rev_rule="Latest Working" -projects=CusProj1,CusProj2
```

- To traverse the BOM structure in the top three levels with the top-level item **ABC002**, item revision **001**, and revision rule **Latest working**, and find all the child revision items in the top three levels and remove only these revision items from the projects **CusProj1** and **CusProj2**, enter the following command on a single line:

```
update_project_bom -u=user -p=password -g=dba
-f=remove -type=item -item=ABC002 -rev_id=001
```

```
-rev_rule="Latest Working" -projects=CusProj1,CusProj2 -depth=3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: locate all the child items in the BOM based on item revision **001**, revision rule **Latest working**, and include these items in the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: no level 2 propagation.

```
update_project_bom -u=user -p=password -g=dba  
-item=ABC001 -rev_id=001 -rev_rule="Latest Working"  
-level=1 -projects=CusProj1,CusProj2,CusProj3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: locate all the child items in the BOM based on item revision **001**, revision rule **Latest working**, and include these items in the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: collect all dataset type objects attached to the BOM line during the BOM traversal. Recursively find all objects that relate to the dataset type objects through the relation specified in the **TC_project_bom_2nd_level_propagate** preference and propagate all of these level 2 objects in the **CusProj1**, **CusProj2**, and **CusProj3** projects.

```
update_project_bom -u=user -p=password -g=dba  
-item=ABC001 -rev_id=001 -rev_rule="Latest Working" -level=2  
-projects=CusProj1,CusProj2,CusProj3
```

- To traverse the BOM structure with the top-level item **ABC001** and perform:
 - Level 1 propagation: traverse the BOM based on item revision **001**, revision rule **Latest working**, and variant rule **AlphaRelease**, find all the child revision items and include only these revision items in the **CusProj1**, **CusProj2**, and **CusProj3** projects.
 - Level 2 propagation: collect all dataset type objects attached to the revision items in the BOM structure during the level 1 propagation. Recursively locate all objects that relate to the dataset type objects through the relation specified in the **TC_project_bom_2nd_level_propagate** preference and propagate all of these level 2 dependencies in the **CusProj1**, **CusProj2**, and **CusProj3** projects.

```
update_project_bom -u=user -p=password -g=dba  
-item=ABC001 -rev_id=001 -rev_rule="Latest Working"
```

```
-var_rule="AlphaRelease" -level=2  
-projects=CusProj1,CusProj2,CusProj3
```

13.5 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

13.6 Summary

The following topics were taught in this lesson:

- Define projects.
- Create projects.
- Assign objects to projects.
- Use Teamcenter utilities for projects.

Lesson

14 Managing preferences

Purpose

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

Objectives

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

- Create and modify preferences using the **Options** dialog box.
- Create and modify preferences using XML preference files.
- Be familiar with key preferences.
- Report preference changes.
- Import and export preferences.

Help topics

Additional information for this lesson can be found in:

- *Getting Started with Teamcenter*
- *Preferences and Environment Variables Reference*
- *Utilities Reference*

14.1 Managing preferences

Preferences are settings stored in the Teamcenter database that provide a mechanism to control and define Teamcenter behavior by letting you modify the interface, set default behavior, and specify default values. Some preferences are read at startup or when you log on to Teamcenter, and others are read during Teamcenter application usage.

Preferences let you configure many aspects of a session of application behavior, such as how assemblies are revised, whether extension rules are bypassed for specified operations, and which Teamcenter objects are displayed in integrations. Each application may have associated preferences.

- The *definition* of a preference includes all relevant information fields: **Name**, **Protection Scope**, **Category**, **Environment**, **Type**, **Multiple** or **Single** (for values), and **Description**.
- An *instance* of a preference can override the value for a given preference at a specific location. In addition to the information in the preference definition, a preference instance includes **Location** and **Value** or **Values**.

You can manage preferences and set values for preference instances either through a client interface or by importing manually edited preference XML files.

When working with preferences, you consider the protection scope for the definition of each preference and the location where each preference instance applies.

- The *location* specifies the level at which the preference instance value is active. The location for a preference instance can be **Site**, **Group**, **Role**, **User**, or **Environment Variable**. (A location of **None** indicates that the preference is defined, but has no instances.)

Note

A preference location of **Environment Variable** indicates that **Environment Variable** is enabled in the preference definition and that the system has a value set for that environment variable. Preference values set this way are displayed in the preference instance to let you know this value is set by the environment variable.

- The *protection scope* determines who can create a preference instance or change the value of a preference instance. For example, only an administrator can create an instance of a preference defined with **Protection Scope** set to **Site**. Protection scope is a property of the preference definition.

Based on the protection scope, administrators and users can create or modify instances. An instance specifies the protection scope and value of a preference at a specific location. The protection scope also determines precedence behavior when there are multiple instances of the same preference.

For information about working with preference settings through options in the thin client, see the *Thin Client Interface Guide*.

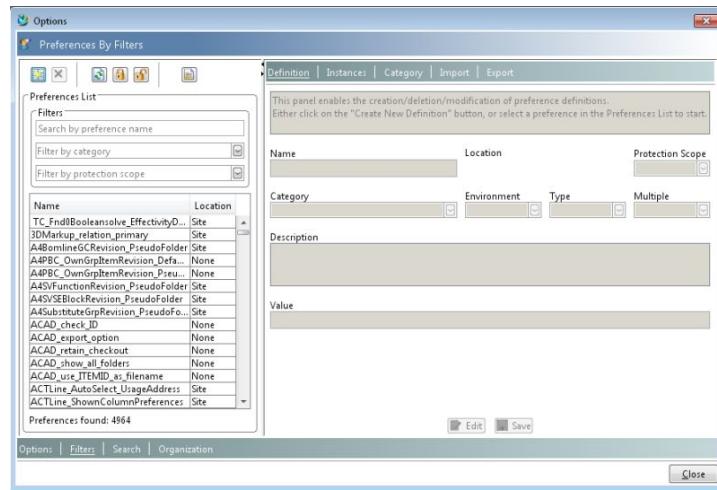
For more information about working with preferences and using options in the rich client, see the *Rich Client Interface Guide*.

To centralize data model-dependent preferences in the Business Modeler IDE, some preferences, such as *ItemRevision-business-object_Maturity_Level* preferences, are converted to Business Modeler IDE template data model objects, such as business object constants. You can migrate such preferences using the **Preferences Migration Wizard**.

For information about migrating preferences to data model objects, see the *Business Modeler IDE Guide*.

14.1.1 Preference definition elements

Administrators can create preference definitions in the Teamcenter rich client **Options** dialog box.



A preference definition consists of the following elements:

Information box The text box on top on the right side provides information such as recommended actions and reasons why some actions are unavailable.

Name

The preference name must be unique and can contain uppercase and lowercase letters, numbers, periods, and the underscore character.

Location

Specifies the level at which the preference instance value is active. The location for a preference value can be **Site**, **Group**, **Role**, **User**, or **None**.

Note

When **Location** is **None**, the preference has a definition but does not have any instances.

Protection Scope

Specifies who can create or edit a preference instance value. Preference protection scope can be **System**, **User**, **Role**, **Group**, or **Site**.

Note

The protection scope of a **System** preference cannot be changed, and the value specified for a **Site** or **System** protection scope preference can only be changed by Teamcenter administrators.

The protection scope for **User**, **Role**, **Group**, or **Site** hierarchical preferences can only be modified by a system administrator.

| | |
|--------------------|--|
| Category | <p>Indicates the functional area in which the preference is used.</p> <p>Preferences are categorized according to the technical area of Teamcenter they affect. For example, all dataset preferences are assigned to the Application Encapsulation category and all Multi-Site Collaboration preferences are assigned to the Multisite category. As a category may contain a large number of preferences, it may contain subcategories to allow easier location and identification of a specific preference. For example, the Multisite category may contain subcategories called Multisite.General, Multisite.ODS, Multisite.IDSM, and Multisite.RemoteImport. A system administrator can create new categories or subcategories that are appropriate for your business practices and assign preferences to those categories and subcategories. Category is a property on the preference definition and any changes apply to all existing and new instances. Searches for preferences can be limited to certain categories.</p> |
| Environment | <p>Specifies whether a preference name can be used (Enabled) as an environment variable to set the preference value or cannot be used (Disabled) as an environment variable.</p> <p>You can declare the preference name as an environment variable with an appropriate value on the Teamcenter server host if the preference definition contains the Enabled value.</p> <p>In such cases, the environment variable is considered an instance of a preference. When the preference definition specifies Environment as Enabled, a preference search first checks for an environment variable with the same name as the preference and, if defined, uses that environment variable value. Otherwise, the normal search pattern for precedence is followed.</p> <p>For information about preference precedence, see System and hierarchical preferences.</p> |

| | |
|--------------------|--|
| Type | Allows you to define one of five types of preferences: <ul style="list-style-type: none">• String Valid values are any valid string of characters.• Logical Valid value is one of any logical pair (on or off, true or false, 0 or 1).• Integer Valid values are any positive integer.• Double Valid values are floating point (real) numbers (-12.34, 99.05).• Date Valid values are calendar date and time entries in <i>DD-MM-YYYY HH-MM-SS</i> format, for example, 23-Fe-2008 16-32-45. |
| Note | To ensure correct display of date format in the interface, Siemens PLM Software recommends users set values from the Options dialog box, rather than through the XML file. |
| Multiple | Specifies whether a preference is defined to accept multiple values (Multiple) or a single value (Single). |
| Description | Provides information about the functionality affected by the preference. |
| Value | Lists the value or values associated with the preference. Note Do not enter localized business object names in preferences. For example, when entering a value in a preference, you must enter the business object name as it appears in the database, not the localized name of the business object. For information about working with business objects, see the <i>Business Modeler IDE Guide</i> . |

For information about defining preferences and creating instances in the rich client, see the *Rich Client Interface Guide*.

14.1.2 System and hierarchical preferences

There are two types of preferences:

- *System* preferences have a protection scope of **System**, and apply to the entire Teamcenter site. These preferences provide settings that apply to the entire Teamcenter deployment. Several system preferences ship with Teamcenter.
- *Hierarchical* preferences have a protection scope of **User**, **Role**, **Group**, or **Site**. The system uses the hierarchy, beginning with **User**, then checking **Role**, **Group**, and **Site**, to retrieve a preference value.

The system uses the hierarchy to search for preference values, starting at the level defined by the protection scope and going up the different levels in the order: from current user, to current role, then current group, and then site.

For example, the system looks for the preference value of a role protected preference in the following manner:

- If a value exists for the current role, that value is used.
- If no value exists for the current role, but a value exists for the current group, that value is used.
- If no value exists for the current role or group, but a value exists for the site, that value is used.
- If no value exists for the current role, group, or site, no value is returned.

Preference values can be defined at different levels depending on the defined protection scope:

- Teamcenter administrators can create instances for preferences that have a protection scope of **User**, **Group**, **Role**, or **Site**.
- Teamcenter group administrator users can create instances for preferences that have a protection scope of **Group**, **Role**, or **User**.
- Teamcenter nonadministrator users can only create instances for preferences that have a protection scope of **User**.

14.1.3 Managing protection scope

All preferences have a protection scope.

Hierarchical preferences that are available immediately in a new installation each have a default protection scope of **Site**, **Group**, **Role**, or **User**. Only system administrators can change the protection scope for hierarchical preferences.

Administrators can manage protection scope for preferences in two ways:

- In the rich client **Options** dialog box

This method is best for modifying small numbers of preferences individually.

- Using the **preferences_manager** utility

This method is best for updating the protection scope, values, and behavior of multiple preferences.

For more information about using the **preferences_manager** utility, see the *Utilities Reference*.

For information about working with preference settings through options in the thin client, see the *Thin Client Interface Guide*.

For more information about working with preferences and using options in the rich client, see the *Rich Client Interface Guide*.

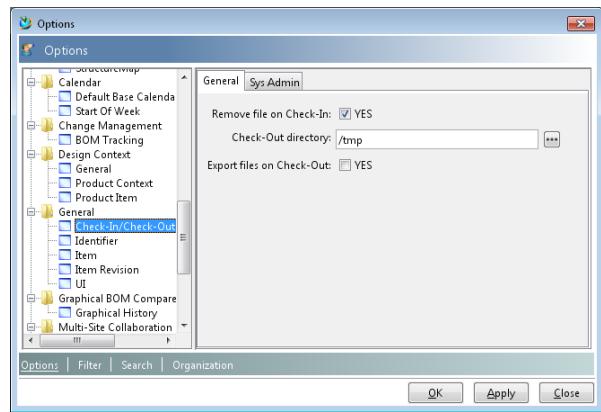
Note

Some preferences are automatically created when certain functionality is added.

14.2 Using the rich client Options dialog box

In the rich client, use the **Edit→Options** menu command to display the **Options** dialog box. The banner at the bottom of this dialog box contains links (**Options**, **Filter**, **Search**, and **Organization**) to different panes: **Options**, **Preferences By Filter**, **Preferences By Search**, and **Preferences By Organization**.

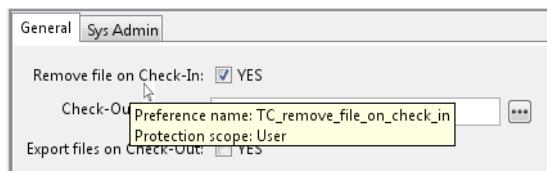
The **Options** pane in the **Options** dialog box lets you interactively select to specify a wide range of behaviors.



- The **Options** pane displays a subset of available preference behaviors organized by functional area on the left side, with descriptive titles and current values on the right side. Some values are further organized by tabs.

Note

To see the name and protection scope of the preference for the behavior, place your mouse cursor over the option. This information is available even when the behavior is not available for editing.



- The **Preferences By Filter**, **Preferences By Search**, and **Preferences By Organization** panes let administrators define preferences, and, depending on protection scope, let administrator and nonadministrator users create instances of preferences or change the values of preference instances to which they have access. Depending on access, these panes display tabs for **Definition**, **Instances**, **Category**, **Import**, and **Export**.

For more information about working with preferences and using options in the rich client, see the *Rich Client Interface Guide*.

14.2.1 Set a preference option

Note

An *option* is an encapsulation of a preference in the **Options** pane in the **Options** dialog box. Place the mouse cursor over an option to display the preference name and its protection scope.

In the **Options** dialog box, in the **Options** pane, fields are enabled or disabled, depending on the protection scope.

You can place your mouse cursor over a value label in the **Options** pane to see a tooltip listing the name and protection scope for the relevant preference. Preference protection scope is also displayed in the **Options** dialog box on the **Preferences By Filters**, **Preferences By Search**, and **Preferences By Organization** panes, when you have selected the **Definition** tab, for administrator users, and the **Instances** tab, for all users.

1. Choose **Edit→Options**.

The **Options** dialog box is displayed with the **Options** pane selected.

2. Click the functional area in the tree that corresponds to the application or system behavior that you want to modify. For example, choose **General→Item** to set item-related options such as whether all revisions are displayed in the interface by default.
3. In the right pane of the dialog box, modify the preference settings.
4. Click **Apply** or **OK**.

14.2.2 Create a new preference definition

Note

Only Teamcenter administrators can create new preference definitions. Mandatory fields are marked with an asterisk (*) and must be populated to enable the **Save** button.

1. Choose **Edit→Options**.
2. In the **Options** dialog box, click the **Filters** or **Search** link.
3. With the **Definition** tab selected, click **Create a new preference definition** .

Teamcenter displays the preference creation boxes.

4. Type a name for the preference in the **Name** box.
The preference name is case sensitive and can contain spaces and special characters.
5. Select one of the following **Protection Scope** options:

User
Role
Group
Site
System

For more information about protection scope, see the *Preferences and Environment Variables Reference*.

6. Select the category for this preference in the **Category** box.
7. Choose whether this preference can be set by using the preference name as an environment variable in the **Environment** box.
8. Select the value type for this preference, either **String**, **Integer**, **Double**, **Logical**, or **Date**.
9. Select one of the **Multiple** options. **Single** limits the values to a single entry. **Multiple** allows multiple values to be entered for the preference.
10. Type a description of the preference in the **Description** box.
Descriptions allow you to communicate valid values, suggested usage, and preference behavior.
11. For a single value, type a value in the **Value** box.

For multiple values:

- a. Type a value in the box beneath the **Values** list.
 - b. Click the **Add** button .
- Teamcenter adds the value to the list.
- c. Repeat until all values are added.

12. Click **Save**.

Teamcenter adds the preference to the database and applies it to the current session.

14.2.3 Create a preference instance for a group

Note

Only Teamcenter administrators and group administrators can create instances or modify values of preferences with **Group** and **Role** protection scope.

1. Choose **Edit→Options**.
2. In the **Options** dialog box, click the **Organization** link to display the **Preferences By Organization** pane.
3. In the **Preferences List** box, select the preference for which you want to create a **Group** instance.

The details of the selected preference are displayed in the **Instances** tab.

4. With the preference selected, use the **Organization** tree to locate and select the group to which to apply the preference instance.

The information box indicates whether you can create an instance of the selected preference for the selected group.

5. Click **Create a new preference instance** .

The **Value** box becomes editable.

6. For a single value, type a value in the **Value** box.

For multiple values:

- a. Type a value in the box beneath the **Values** list.

- b. Click the **Add** button . Teamcenter adds the value to the list.

- c. Repeat until all values are added.

7. Click **Save**.

Teamcenter adds the preference to the database and applies it to the current session.

14.2.4 Create a preference instance for a nonadministrator user

Note

Nonadministrator users can create and modify instances of preferences with **User** protection scope.

1. Choose **Edit→Options**.
2. In the **Options** dialog box, click the **Filters** or **Search** link, and then click the **Instances** tab.
3. In the **Preferences List** box, select the preference for which you want to create a **User** instance.
 - The details of the selected preference are displayed in the **Instances** tab, and the information box indicates whether you can create an instance of the selected preference.
 - When there are several values, such as values at the site level, and for your group, and for your role, select the level that you want to create the new instance.

The value or values of the new instance are populated from the selected preference instance. That is particularly useful for multivalued preferences, when the preference contains several values and the intent at the user level is to simply add one value to the list.

Note

The selected preference may be in the **Site** location, but it must have **Protection Scope** set to **User**.

4. Click **Create a new preference instance** .
- The **Value** box becomes editable.
5. For a single value, type a value in the **Value** box.
- For multiple values:
- a. Type a value in the box beneath the **Values** list.
 - b. Click the **Add** button . Teamcenter adds the value to the list.
 - c. Repeat until all values are added.
6. Click **Save**.
- Teamcenter adds the preference to the database and applies it to the current session.

14.3 Creating and editing preferences from preference XML files

Administrators can manually edit preference XML files to create or modify a preference, and then use the **preferences_manager** utility to load the XML files, or use the rich client to import the files.

The **preferences_manager** supports batch operations and effective bulk processing.

14.3.1 Edit a preference XML file

1. Make a copy of the preference XML file, for example, the **tc_preferences.xml** file.
2. Apply your changes using an XML editor.
3. Import the changes using the **preferences_manager** utility or the rich client **Options** dialog box.

For more information about the **preferences_manager** utility, see the *Utilities Reference*.

For information about using the **Options** dialog box to import preferences into the database, see the *Rich Client Interface Guide*.

To view the changes, use the rich client **Edit→Options** menu command to display the **Options** dialog box, and then locate the new preferences in the **Preferences By Filter**, **Preferences By Search**, or **Preferences By Organization** pane.

preferences_manager

DESCRIPTION

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**, **rpfles**, and **gpfiles** preference files to the database.

The **preferences_manager** utility is location in *TC_ROOT/bin* directory.

EXAMPLES

- Displays the help information on the console.

```
preferences_manager -h
```

- Displays the options that can be used with **mode=generatexml** option.

```
preferences_manager -mode=generatexml -h
```

- Generates the site preferences XML file from the legacy site preference file.

```
preferences_manager -u=user-id -p=password -g=dba  
-mode=generatexml -context=Teamcenter  
-file=legacy-preference-file  
-out_file=C:\temp\site_pref.xml
```

- 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.

```
preferences_manager -u=infodba -p=password -g=dba  
-mode=import -scope=SITE  
-file=c:\temp\site_pref.xml -action=SKIP
```

- 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.

```
preferences_manager -u=infodba -p=password -g=dba  
-mode=import -scope=SITE  
-file=c:\temp\site_pref.xml -action=MERGE
```

- Imports a preference and override the values in the database with the values specified for the preference on the command line.

```
preferences_manager -u=infodba -p=password -g=dba  
-mode=import -scope=SITE  
-file=c:\temp\site_pref.xml -action=override
```

- Exports all group preferences in the database for the user's group when the utility is run for that specific group.

```
preferences_manager -u=user-id -p=password  
-g=group-name -mode=export -scope=GROUP  
-out_file=c:\temp\group_pref.xml
```

- Exports all user preferences in the database for a user when the utility is run as that specific user.

```
preferences_manager -u=user-id -p=password  
-g=group-name -mode=export -scope=USER  
-out_file=c:\temp\user_pref.xml
```

14.3.2 Specify dual OS values for a single preference

If you import preferences using the **preferences_manager** utility, you can define both UNIX and Windows values for the same preference. To do so, you must set **array** to false, directing the system to return a single value for the preference query, even when multiple values are defined.

Note

If you define both UNIX and Windows values for a preference, the value returned is based on the server operating system.

1. Create an XML file containing the relevant preference definitions.
2. Set **array** to **false**.
3. Define **platform** values for both UNIX and Windows. For example:

```
<preference name="TC_audit_log_dir" type="String" array="false" disabled="false">
  <preference_description>Specifies the audit log directory.</preference_description>
  <context name="Teamcenter">
    <value platform="UNX">$TC_LOG/audit</value>
    <value platform="WNT">%TC_LOG%\audit</value>
  </context>
</preference>
```

4. Import the XML file to the database with the **preferences_manager** utility using the **OVERRIDE** option.

For more information about this utility, see the *Utilities Reference*.

14.4 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 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.

14.4.1 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. In the rich client, choose **Edit→Options** to display the **Options** dialog box.
2. Click either **Filters** or **Search**.
3. Click **Report**  in the banner above the **Preferences List** box.
The **Report** dialog box appears.
4. Select the protection 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 modified values. Use the forward and back buttons at the top of the dialog box to view the various preferences.

5. (Optional) Export the report in XML format:
 - a. In the **Report Changes** dialog box, type a path and file name in the **Export To File** box (or browse to an existing file).
 - b. (Optional) Select **Open After Export**.
 - c. Click **Export**.

The preference report is exported to the specified file.

14.5 Import and export preferences

Preferences can be imported with the **Options** dialog box.

- Using the **Import** and **Export** links in the **Options** dialog box.

Use the **Import** link in 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 in 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.

14.5.1 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. Select one of the following options for handling duplicate preferences:

- **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.

14.5.2 Export preferences from the database to an XML file

1. Choose **Edit→Options**.
2. In the **Options** dialog box, click the **Filters**, **Search**, or **Organization** link.
3. Click the **Export** link at the top of the pane.
4. Select the hierarchy level to export in the **From Location** list.
5. (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.
6. Type a path and file name for the export file in the **Export To File** box or click the button to the right of the box and locate a file in your directory structure.
7. (Optional) Select the **Open After Export** check 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.

14.6 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

14.7 Summary

The following topics were taught in this lesson:

- Create and modify preferences using the Options dialog box.
- Create and modify preferences using .XML preference files.
- Report preference changes.
- Import and export preferences.

Lesson

15 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.
- Import and export query definitions.

Help topics

Additional information for this lesson can be found in:

- *Query Builder Guide*

15.1 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.

15.1.1 Basic concepts for using Query Builder

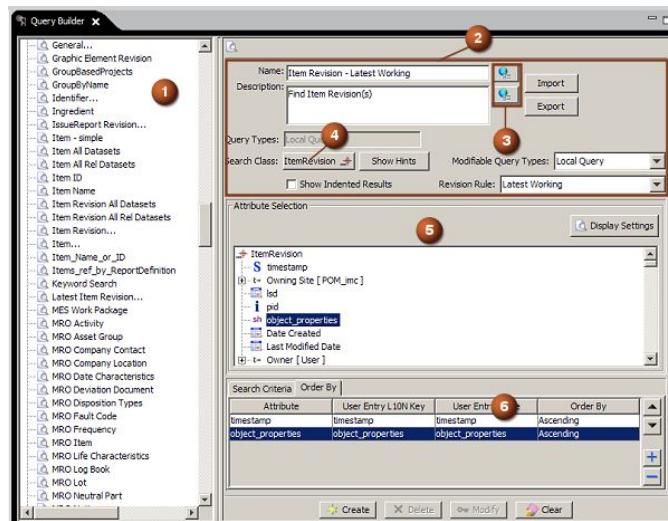
Depending on your needs, you can use an existing query from the saved queries tree. Saved queries are subject to standard object protection and are accessed by users through the search feature in My Teamcenter.

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 based on an existing definition](#)
- 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.

15.1.2 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 **Localization** button Displays the **Language Translations** dialog box that lists existing translation values.
- 4 **Search Class** button Displays the **Class/Attribute Selection** dialog box that lists the query classes for selection.
- 5 **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.
- 6 **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.

15.1.3 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. |

| 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.

Note

Siemens PLM Software recommends the following when using the **Search Criteria** pane:

- For each clause, make certain the **User entry L10N key** value is unique.
- When you use **AND** and **OR** clauses together, the search result displays unexpected results. Therefore, use the **AND** operator (rather than the **OR** operator) and place multiple default values in the **Default value** column separated by a semicolon, as shown in the following example.

| | Attribute | User entry L10N key | User entry name | = | Default value |
|-----|-------------|---------------------|-----------------|---|----------------------|
| | object_name | object_name | Name | = | * |
| AND | object_type | object_type | Type | = | UGMASTER;DirectModel |

The **Order By** tab contains the following search criteria elements.

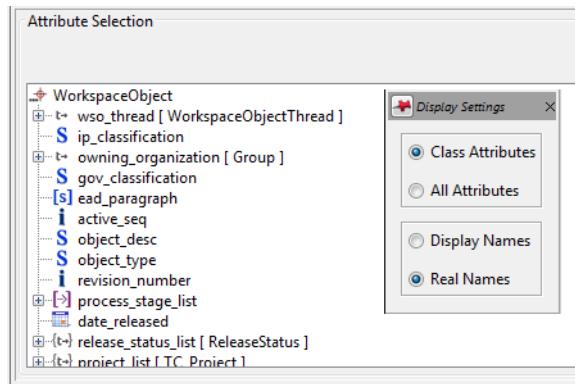
| Element | Description |
|----------------------------|---|
| Attribute | The selected database attribute appears in this box. |
| User Entry L10N Key | Specifies the localization key used to look up user entry names. The localization key-value pairs are defined in the qry_user_entry_names_locale.xml 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. |
| Order By | Specifies the sort order or overrides a default sort order defined for a saved query. Each attribute can be set to arrange in either Ascending or Descending order. |

15.1.4 Properties for workspace objects

To create query definitions, you must understand the attributes defined on each class. Because there are many classes in the database, each with many attributes, you should understand the **WorkspaceObject** class hierarchy first.

The **WorkspaceObject** class defines the *common attributes* of all the workspace objects: folder, form, dataset, item, and item revision.

- The attributes without the plus \oplus symbol can be directly set when searching for objects in the **WorkspaceObject** class.
- The attributes with the plus \oplus symbol refer to another class in the class structure, which may have direct attributes, or may have more attributes with \oplus symbols.



Key points

- Use Query Builder to create customized searches for objects called *query definitions*.
- Query definitions are based on objects attributes.
- The **Name** and **Type** attributes are defined directly to the **WorkspaceObject** class.
- Some other attributes for the **WorkspaceObject** class are inherited from parent classes, but where these attributes are actually defined is not a concern at this time.

15.1.5 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.

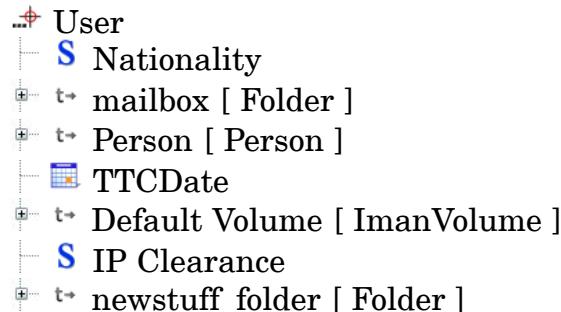
15.1.6 Using class attribute selections

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.

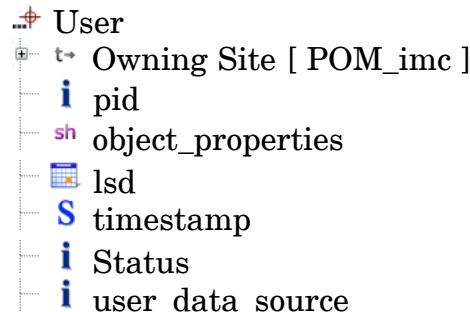
Attribute Selection box



- Parent classes.

When all attributes for the **Search Class** are displayed, inherited attributes are also displayed in the **Attribute Selection** box.

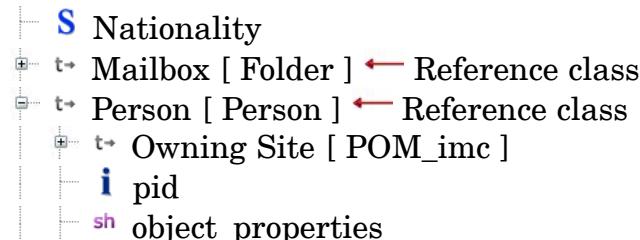
Attribute Selection box



- 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

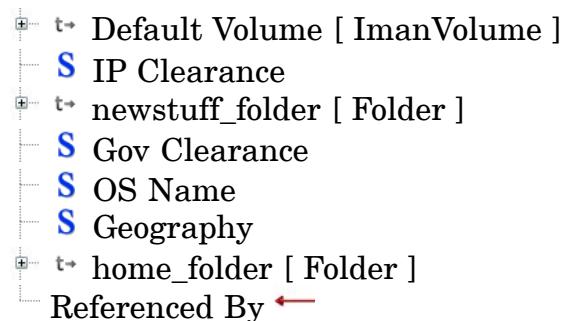




- Referenced by classes.

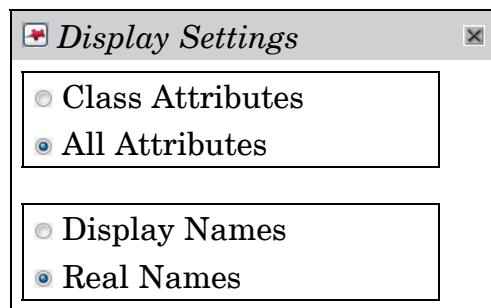
When classes related to the **Search Class** are not displayed, the **Referenced By** attribute is used to locate additional reference classes to display in the **Attribute Selection** box.

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 \oplus can be directly set when searching for objects.
- The attributes with the plus symbol \oplus refer to another class in the class structure, which may have direct attributes, or may have more attributes with plus symbols.

15.2 Creating customized searches using Query Builder

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

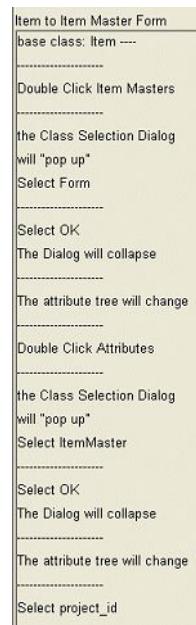
15.2.1 Create a new query based on an existing definition

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.

15.2.2 Create a query using the hints feature

Query hints are provided to assist you in navigating the schema by presenting a relationship to traverse and the steps required to build that relationship into your query definition.



Steps to use the hints

1. Click the **Show Hints** button in the saved query properties pane.
The system displays the hints pane.
2. Click the **Choose Hint** button located at the top of the hints pane.
The system displays the choose hint pane containing the directory of query hints.
3. Expand the **Item Queries** folder, select **Item to Item Master Form**, and click **OK**.
The dialog box closes and the system displays the selected hint in the hints pane. The first entry in the hint is the base class, followed by the traversal steps, ending with the attribute of the item master form class that is added to the query clause.

Note

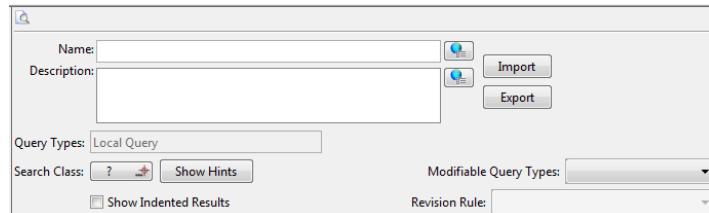
The available hints are dependent on the **Search Class**.

15.2.3 Create a query using the IS_NULL operator

The **IS_NULL** and **IS_NOT_NULL** operators allow you to create queries to find objects with null attribute values, such as items that do not have descriptions, or objects with an attribute containing any value other than null. Clauses that use this operator are treated as having a fixed value; therefore, there is no need to enter a name and default value for the clause.

15.2.4 Create queries

Start by entering the query information at the top of the Query Builder dialog box.

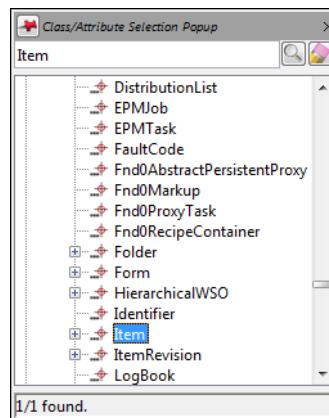


1. If necessary, click the **Clear** button to remove existing information from the Query Builder boxes.
2. In the **Name** box, type a unique name for the query.
3. Optionally, type a description of the query in the **Description** box.
4. Select **Local Query** from the **Modifiable Query Types** list.

15.2.5 Class/Attribute selection

Continue to define the search by selecting the **Search Class** and the **Attribute Selection**.

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.

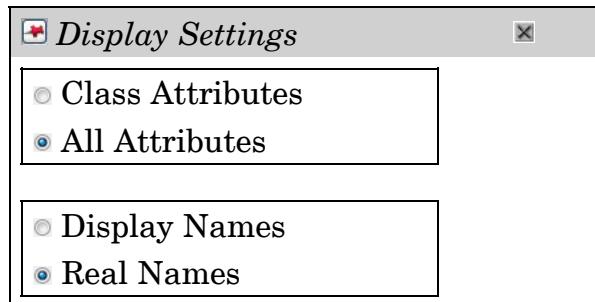


1. Click the **Search Class** button and the **Class/Attribute Selection** dialog box appears.
2. Specify the desired search class by one of the following:
 - By selecting an entry from this dialog box.
 - By typing the class name (or partial name and wildcards) in the box at the top of the dialog box, and click the **Press to add input name** 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.
3. To display the search results in an indented or hierarchical form, select **Show Indented Results**.
4. Double-click to select at least one of the attributes in the **Attribute Selection** tree and the attribute is added to the **Search Criteria** table.
5. Specify the desired search criteria. You must specify the following required search criteria boxes:
 - **Attribute**
 - **User Entry L10N Key**
 - **Logical operators**
6. 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.

15.2.6 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 \oplus 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 \oplus to directly add them to the **Search Criteria** table.

15.2.7 Custom item type query definitions

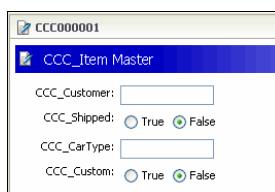
The general purpose of the query engine is to search the database for many kinds of data. It is often necessary to build queries to find items and item revisions based on the item master and item revision master form attributes and list of values.

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 base 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.

15.3

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.

15.3.1 Import query definitions

You can 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.

15.3.2 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.

15.4 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

15.5 Summary

The following topics were taught in this lesson:

- Define query definitions basic concepts.
- Create and modify query definitions.
- Import and export query definitions.

Lesson

16 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*

16.1 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**

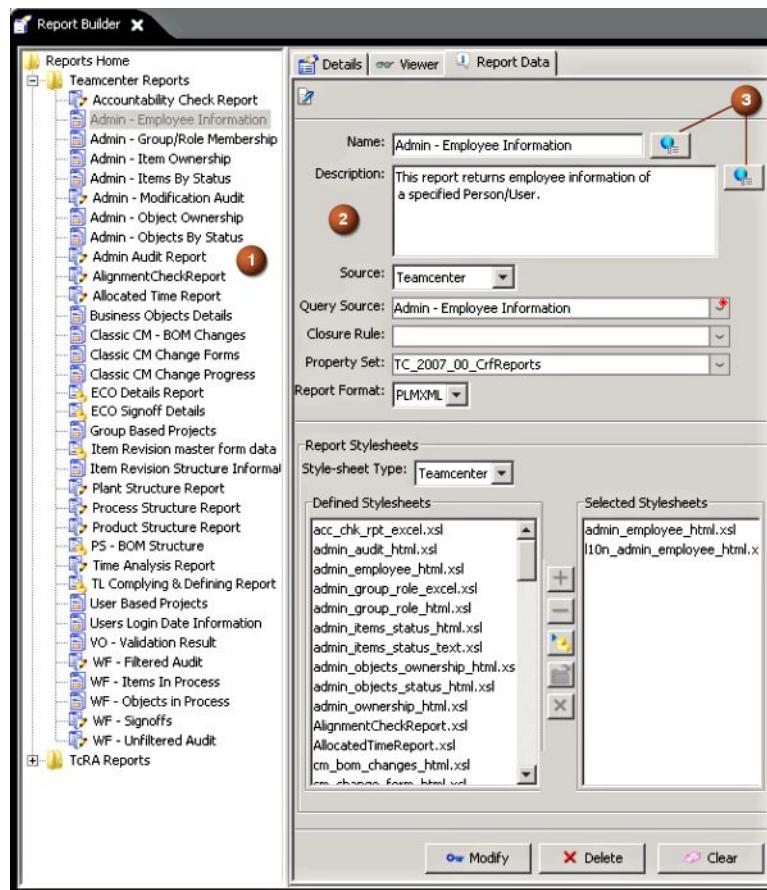
Reports that can be run on a particular item, for example, reports that show the BOM list for an item or the workflow signoff for an item.

- **Custom**

Reports that use custom processing.

Report Builder works with the Query Builder application and the PLM XML/TC XML Export Import Administration application. To assemble reports, you can use saved queries created in Query Builder and property sets created in PLM XML/TC XML Export Import Administration.

16.1.1 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 reporting and analytics.

2 Report Builder tabs

The Report Builder tabs display file details, properties, and configurable data for the selected report.

3 Localization button

Displays the **Language Translations** dialog box that lists existing translation values.

16.1.2 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 stylesheet to format the output.

16.1.3 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/TC 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.

16.1.4 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 stylesheet 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 TC XML 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 stylesheets 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 stylesheets. The sample stylesheets 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.

16.1.5 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-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. |

16.2

Importing and exporting report definitions

The **import_export_reports** utility allows report definitions, their dependent data (for example, saved query definitions and property set definitions), and their associated stylesheets 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]
```

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 stylesheets are written. On import, this is the name of the directory where the report definition and stylesheets are located. |

16.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

16.4 Summary

The following topics were taught in this lesson:

- Different types of report definitions.
- Create summary and item report definitions.
- Import and export report definitions.

Lesson

17 PLM XML import and export

Purpose

The purpose of this lesson is to introduce the PLM XML/TC XML Import Administration application and identify the basic tasks necessary for creating transfer mode objects.

Objectives

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

- Define and create transfer modes.
- Create closure rules and property sets.
- Implement transfer mode best practices.

Help topics

Additional information for this lesson can be found in:

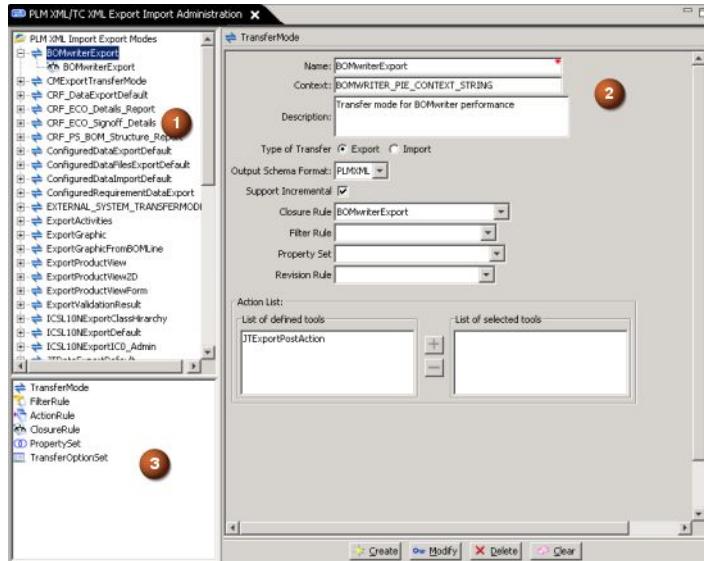
- *PLM XML/TC XML Export Import Administration Guide*

17.1 What is PLM XML/TC XML Export Import Administration?

The PLM XML/TC XML Export Import Administration application is used by Teamcenter administrators who migrate data between Teamcenter and other applications. You use PLM XML/TC XML Export Import Administration to create transfer mode objects that contain rules that configure import or export operations. This simplifies the import and export process for end users, who can select a transfer mode object when importing or exporting data and do not need to know the specific rules the transfer mode contains. You do not actually import or export data using PLM XML/TC XML Export Import Administration; you use other applications, such as My Teamcenter or Structure Manager.

17.1.1 PLM XML/TC XML Export Import Administration interface

The PLM XML/TC XML Export Import Administration interface is the standard rich client interface.



- | | |
|--|--|
| <p>1 Object tree</p> | <p>Lists the transfer modes and associated objects. When selected, the details for the object are displayed in the TransferMode pane.</p> |
| <p>2 TransferMode pane</p> | <p>Displays the details of the selected object. You can create, modify, and delete objects in this pane.</p> |
| <p>3 TransferMode tree</p> | <p>Lists the objects that you can add to a transfer mode.</p> |

17.1.2 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.

17.2 Basic tasks using PLM XML/TC XML Export Import Administration

With PLM XML/TC XML Export Import Administration, you create transfer mode objects that help end users export and import Teamcenter objects and system data. To create a transfer mode object, you may need to:

- [Create or edit a closure rule.](#)
- Create or edit a filter rule.
- Create or edit an action rule.
- Create or edit a property set.

Once you create these items, you can assemble them into a transfer mode object.

You can also create a transfer option set for use with Data Exchange.

17.3 What are transfer modes?

Transfer mode objects are created in PLM XML/TC XML Export Import Administration and are displayed as options when users import or export objects or system data using one of the Teamcenter applications. They allow users to export and import data with little knowledge other than the name of the transfer mode object that they should use; for example, **ToCompanyA**, **FromCompanyB**, or **ToCompanyB**.

Note

When importing data in to Teamcenter, object names and IDs that are too long for the field are truncated.

Transfer mode objects are made up of the following items that configure the import or export operation.

- **Closure rules**

Define the scope of the data translation.

- **Filter rules**

Use conditions to apply a finer level of control over which data gets translated along with the primary objects.

- **Action rules**

Sets of methods that can be called before, during, and after the translation.

- **Property sets**

Provide a mechanism for PLM XML objects to have arbitrary properties in the form of **UserData** elements.

17.3.1 Create closure rules

Closure rules control the scope of the data translation on both input and output. They specify how the data structure is traversed by specifying which relationships are of interest and what should be done when these relationships are encountered.

Closure rules are comprised of five parts:

- Primary object selector
 - Secondary object selector
 - Relation selector
 - Action
 - Optional conditional clause
1. Open PLM XML/TC XML Export Import Administration.
 2. Select **ClosureRule** in the lower left pane of PLM XML/TC XML Export Import Administration.
 3. Type a unique name and description.
 4. Select **Export** or **Import** 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**.

17.3.2 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.

Example

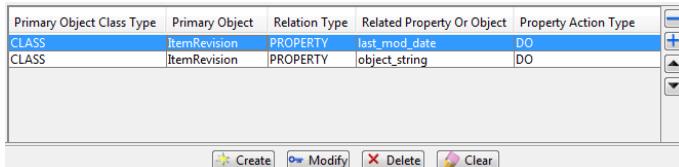
You define a property set using the following syntax:

- CLASS.ItemRevision:PROPERTY.last_mod_date:DO
- CLASS.ItemRevision:PROPERTY.object_string:DO

The data transformed in the PLM XML file looks like this:

```
<ProductRevision id="id12" name="assy" accessRefs="id5" masterRef="#id20" revision="A">
  <UserData id="id15">
    <UserData value="02-Dec-2003 12:37" title="last_mod_date"/>
    <UserData value="000338/A-assy" title="object_string"/>
  </UserData>
</ProductRevision>
```

In the **New Property Set** dialog box, you can define property set clauses without writing code.



- A property set adds the Teamcenter custom data into a PLM XML file that may not exist in the PLM XML schema.
- The property sets controls the **UserData** element.
- The **UserData** element is a container for a list of name-values pairs that allows you to add any attribute or property of a Teamcenter object to the PLM XML operations.

17.3.3 Create 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.

1. Open the PLM XML/TC XML Export Import Administration.
2. Select **TransferMode** in the lower left pane of PLM XML/TC 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 **Export** or **Import** 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**.

Once you create the transfer mode object in PLM XML/TC XML Export Import Administration, you can use the rich client, thin client, or a command line utility to import or export data.

17.4 Transfer mode best practices

Transfer modes focus on specific data traversals. You can modify transfer modes to tune them for better performance.

17.4.1 Transfer mode limitations

Do not use the **JTDataExportDefault** and **JTDataImportDefault** transfer modes to directly export or import JT files with the **plmxml_export** and **plmxml_import** utilities or the **PLM XML** option in the user interface. Instead, use the PLM XML format rather than the JT assembly format. This exports and imports JT files, along with the PLM XML file containing the item information and assembly relationships.

Furthermore, do not use the following transfer modes with the **plmxml_export** and **plmxml_import** utilities:

- **BOMwriterExport**
- **PLMXMLAdminDataExport**
- **TIEImportDefault**
- **TIEPDXExportDefault**
- **TIEUnconfiguredExportDefault**

17.4.2 Use specific type identifiers for better performance

The following example describes a typical optimization of the closure rules in a transfer mode. This example is not complete and does not constitute a specific modification. The following closure rules are used in the **ConfiguredDataExportDefault** transfer mode:

```
TYPE.BOMLine : TYPE.* :PROPERTY.bl_attachments:TravAndProc:  
    SECONDARY.al_source_class==Dataset  
TYPE.BOMLine : TYPE.* :PROPERTY.bl_attachments:TravAndProc:  
    SECONDARY.al_source_class==Form  
TYPE.BOMLine : TYPE.* :PROPERTY.bl_attachments:TravAndProc:  
    SECONDARY.al_source_class==Folder
```

These closure rules:

- Traverse and process any **SECONDARY** type from **BOMLine** to any type by using the **bl_attachments** property.
- Execute the **al_source_class** function on the **SECONDARY** type.
- Support its export if the function returns a dataset, form, or folder.

To optimize the closure rules, change the types to take advantage of specific type identifiers. The engine looks only at these types and not all types.

```
TYPE.BOMLine : TYPE.Dataset:PROPERTY.bl_attachments:TravAndProc  
TYPE.BOMLine : TYPE.Form :PROPERTY.bl_attachments:TravAndProc  
TYPE.BOMLine : TYPE.Folder :PROPERTY.bl_attachments:TravAndProc
```

17.4.3 Unload objects in transfer mode to speed processing

You can use the **PIE_transfer-mode-name_unload_objects** and **PIE_transfer-mode-name_process_in_chunks** user preferences to unload objects you no longer need in memory. The **PIE_transfer-mode-name_unload_objects** preference tells Teamcenter which objects to unload from memory and in what order.

Caution

If you unload objects, such as BOM lines, required by other rich client functions or features, you may cause errors. Also ensure all objects are saved to the database before unloading.

If you set the **PIE_transfer-mode-name_unload_objects** preference, you can also use the **PIE_transfer-mode-name_process_in_chunks** preference to specify the objects that are to be unloaded in chunks or batches. The default is that all objects are unloaded in chunks.

17.4.4 Improve export performance

You can improve the performance of your PLM XML exports by setting the following preference:

- **PLMXML_export_packed_bom_<transfer-mode-name>**

If you set this preference to **TRUE**, it improves the PLM XML export performance for the specified transfer mode. But if you export a packed BOM, you may lose data in the exported XML file. Therefore, do not import an XML file with a packed BOM into Teamcenter. Set this preference to **TRUE** only if you plan to use the XML file in a third-party application.

17.4.5 Maintain COTS scope rules

COTS scope rules (transfer modes, closure rules, filter rules, and so on) are maintained in XML files in the *TC_DATA* directory (for example, **defaultTransfermodes.xml**). The file is imported to Teamcenter using the **tcxml_import** command line utility during the database installation and upgrade. Starting in Teamcenter 9.1, the transfer mode **.xml** files are imported in overwrite mode as part of the upgrade. You do not need to manually import the files. To reload the latest changes from the **.xml** file, you can run the **tcxml_import** utility in overwrite mode to ensure that the updates to existing scope rules are also imported to the database. For example:

```
$ TC_ROOT/bin/tcxml_import  
-u=user -p=password -g=group -file=$TC_DATA/defaultTransfermodes.xml  
-scope_rules -scope_rules_mode=overwrite
```

Siemens PLM Software recommends that you do not modify the COTS scope rules. If necessary, you can create a copy of the COTS rules and modify the copy as needed.

17.5 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

17.6 Summary

The following topics were taught in this lesson:

- Use transfer modes to simplify the import and export process for end users.
- Transfer mode objects are made up of closure rules, filter rules, actions rules, and property sets.
- Modify transfer modes to tune them for better performance.

Lesson

18 Introduction to Workflow Designer

Purpose

The purpose of this lesson is to learn to use Workflow Designer, providing the necessary background skills for workflow process modeling in the next lesson.

Objectives

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

- Describe the workflow process.
- Describe Enterprise Process Modeling (EPM).
- Define a process model.
- Define Workflow templates.
- Create Workflow process templates.

Help topics

Additional information for this lesson can be found in:

- *Workflow Designer Guide*

18.1 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**.

18.1.1 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 using 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. |

18.1.2 Workflow process approaches

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.

18.2

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

18.3 Basic tasks using Workflow Designer

To design and maintain workflow processes in Workflow Designer, you can perform the following actions:

- Create templates.
- View templates.
- Add tasks to templates.
- Link tasks.
- Modify task behavior.
- Import and export workflow templates.

18.4 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 Workflow Designer.

3. Click **Edit Mode**  to edit templates.

To use Workflow Designer in edit mode, you must be a member of the system administration group.

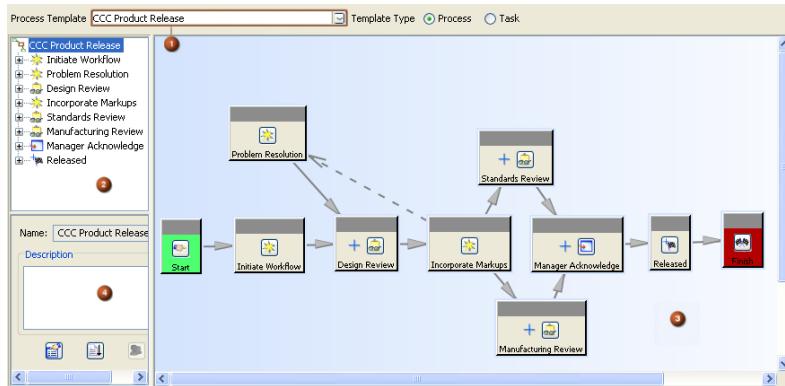
Note

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.

18.5 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 Task hierarchy tree

Displays hierarchical relationship of all tasks in the selected workflow process template or of all subtasks contained within the selected task template. For example, selecting a container task displays all its subtasks. Task order within this tree does not indicate task execution order.

3 Process flow pane

Displays a graphical representation of all tasks in the selected workflow process template or of all subtasks within a selected task template.

4 Template manager pane

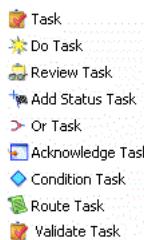
Contains elements related to managing the selected workflow process template or task template. The elements displayed depend on the status and configuration of the selected template.

18.5.1 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. *Process templates* define a blueprint of a process or task to be performed at your site.

Workflow task templates

A *task* is a fundamental building block used to construct a process. Each task defines a set of actions, rules, and resources used to accomplish that task. Example tasks include **Task Do**, **Review**, **Add Status**, and **Route**.



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.

18.5.2 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 **EPMTTask** template.

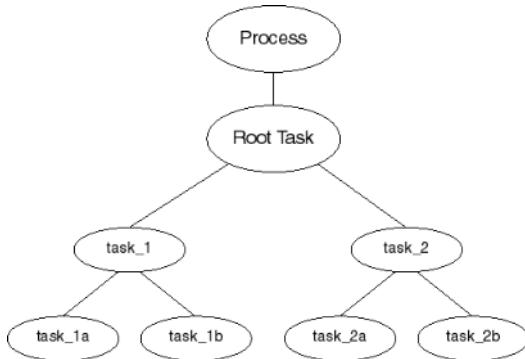
This table lists the task templates available in Workflow Designer.

| Symbol | Task template | Definition |
|--------|-------------------------------------|---|
| | Task | Uses the EPMTTask 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. |
| | Review Task | Uses the select-signoff-team and perform-signoff subtasks, each of which has its own dialog box. |
| | Wait for Undecided Reviewers | 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 EPM-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. |

18.6 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:

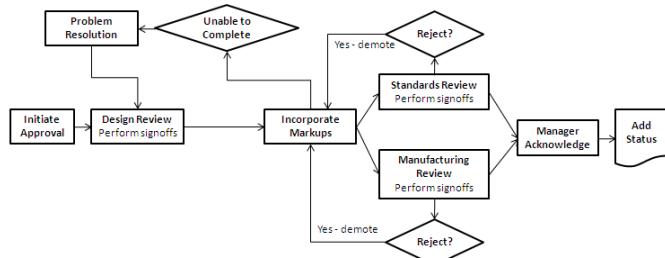
- A specific process by placing workflow and/or change management tasks (**do**, **perform signoff**, **route**, **checklist**, and so on) in the required order of performance.
- Additional process requirements (such as quorums and duration times) in the template, using workflow handlers.

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.

18.7 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.

18.8

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.

Tasks are used to perform actions within a job. When all tasks defined for a job are completed, the job is complete.

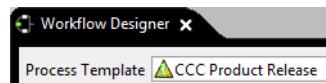
Examples

- 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.

18.8.1 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.

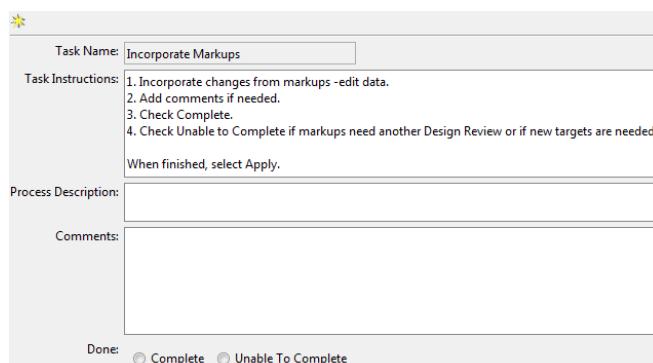
18.8.2 Do task templates

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.



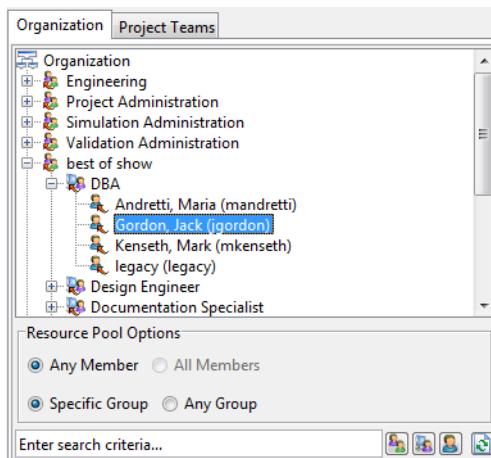
If you require user authentication before a **Do task** can be performed, add the **EPM-require-authentication** handler to the **Perform** action of the task. When you implement user authentication for this task, a password box displays below the **Comments** box. Users must type their user password in this box before they can click **Apply** and complete the task.

18.8.3 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. Signoff team members choose one of these options to perform the signoff.

- **Approve**
- **Reject**
- **No Decision**

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.

| | | | |
|------------------|-------------------------------|--|---|
| Signoffs Quorum: | <input type="radio"/> Numeric | <input checked="" type="radio"/> Percent | <input type="checkbox"/> Wait For Undecided Reviewers |
|------------------|-------------------------------|--|---|

18.8.4 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.

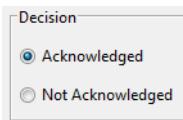
Note

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.

18.8.5 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.

18.8.6 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.

18.9 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

18.10 Summary

The following topics were taught in this lesson:

- Describe the workflow process.
- Describe Enterprise Process Modeling (EPM).
- Define a process model.
- Create multiple task Workflow process templates.

Lesson

19 Building 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:

- Modify task behavior using attributes, handlers, and access control lists.
- Deploy Workflow templates to users and groups.

Help topics

Additional information for this lesson can be found in:

- *Workflow Designer Guide*

19.1 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. Possible task attributes are:

- o Named ACL
- o Template name
- o Signoff quorum
- o Release status
- o 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:

- o Set protections
- o Assign reviewers
- o Add related objects as targets
- o Demote a task
- o Perform a signoff
- o 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.

19.1.1 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 signoff team in a workflow process with a specific role in a specific group (RIG).

Note

This accessor is used only 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 signoff team in a workflow process.

- **Approver (Group)**

Users in a specific group who are members of a signoff team in a workflow process.

- **Approver**

Users who are members of a signoff 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.

19.2 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:

- o Displaying information
- o Retrieving the results of previous tasks (inherit)
- o Notifying users
- o Setting object protections
- o Launching applications

- **Rule handlers**

Use rule handlers to integrate workflow business rules into workflow processes at the task level. Rule handlers attach conditions to an action. Rule handlers confirm that a defined rule has been satisfied. If the rule is met, the handler allows the task to continue. If the rule is not met, the handler prevents the task from continuing.

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.

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 |

Note

- Handler values are case sensitive and must be accurate to the letter.
- If an argument calls for the name of an object, attribute, or property defined in the Business Modeler IDE, it must use the actual name, not its display name.

19.2.1 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.

Action handlers

- **EPM-create-status** and **EPM-set-status**
 - The **EPM-create-status** handler creates a status object and attaches it to the root task.
 - The **EPM-set-status** handler applies the appropriate status objects to the target objects.

Note

You can use the **delete** argument with the **EPM-set-status** handler to remove status objects from targets that were applied in Workflow processes.

- **EPM-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.

- **EPM-require-authentication**

Displays a password box in the perform dialog box or pane of the task within which it has been placed.

- **EPM-demote** and **EPM-demote-on-reject**

The **EPM-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 **EPM-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.

- **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.

Rule handlers

- **EPM-assert-targets-checked-in**

Verifies that all target objects in this process are checked in.

- **EPM-disallow-adding-targets** and **EPM-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 **EPM-disallow-removing-targets** handler prevents targets from being removed from a process after the process is started.

Note

Never add a Change Management handler to a workflow process. CM handlers are designed to be used only in change processes.

19.3 Adding secure tasks

To secure a task, you use the **EPM-require-authentication** action handler. A password box appears and the user must type the logon password to complete the task.

Place the **EPM-require-authentication** handler on the **Perform** action of the following tasks:

- **Do** tasks
- **perform-signoff** tasks
- **Condition** tasks

Note

When working with a **Route** task, place the **EPM-require-authentication** handler on the **Perform** action of the **perform-signoff** subtask of either the **Review** or **Acknowledge** tasks.

19.4 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→Workflow Process**.

Using the template filter

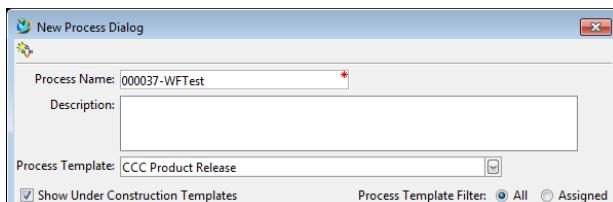
Process templates are assigned by using **Preferences** for the specified group.

In Workflow Designer, choose **Edit→Template Filter** and then select templates for user groups and object types.

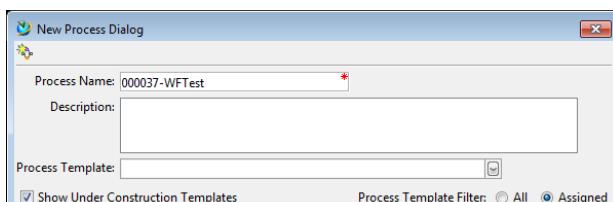
Blocking users from choosing from all process templates

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→Workflow Process**.

- Setting **CR_allow_alternate_procedures** to **any** allows the **All** option in the **Process Template Filter** to be selected.

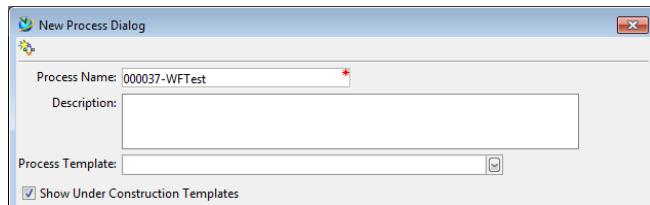


Setting **CR_allow_alternate_procedures** to **Assigned** allows the **Assigned** option in the **Process Template Filter** to be selected.



- 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.



19.5 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

19.6 Summary

The following topics were taught in this lesson:

- Add security to tasks in process templates.
- Assign templates to users and groups.

Lesson

20 Workflow sharing and additional paths

Purpose

The purpose of this lesson is to explain several ways of setting alternate workflow paths in Teamcenter when designing a workflow.

Objectives

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

- Import and export process templates.
- Link tasks in a workflow.
- Define and create a failure path.
- Set **Condition** paths.
- Use a **Validate** task to branch a workflow.

Help topics

Additional information for this lesson can be found in:

- *Workflow Designer Guide*

20.1 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.

20.1.1 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.

20.1.2 Import a process template

1. Choose **Tools**→**Import**.

The system displays the **Import Workflow Templates** dialog box.

2. Type the path to the directory containing the export file in the **Import File** box, or click the **Browse** button to locate the directory.
3. If you want the system to continue the transfer if one or more templates fail to transfer, select the **Continue On Error** option. If one or more templates fail to transfer, the system records transfer errors in its log files, bypasses the failed templates, and transfers the remaining templates.

If you do not select this option, the system stops the transfer process if one template fails to transfer and only includes in the transfer those templates that transferred successfully.

4. If you want the system to overwrite any template of the same name that already exists in the database, select the **Overwrite Duplicates** option. The system does not display or log any errors.
If you do not select this option, any importing template with the same name as an existing template is ignored and the import process continues. The system does not display or log any errors.
5. Click **OK** to import the templates contained within the file you selected into the Teamcenter database.

The imported template names now exist in the database and display in the **Process Template** list.

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.

20.2

Linking tasks in a workflow process template

A link establishes the sequence by which peer-level tasks are executed, indicating that the task on the arrow end of the path cannot start until the task on the start end is completed.

| | |
|----------------|--|
| Explicit links | Manually created links, drawn from the predecessor task to the successor task. |
| Assumed links | Automatically created by the system if no explicit links have been created from the Start node by the time the template is set to the Available stage. |

When you put a workflow template in **Edit** mode and draw a single link from the **Start** node to another task node, assumed link behavior is disabled. The system does not draw assumed links, even if you leave tasks unlinked and change the workflow template to the **Available** stage. Any unlinked tasks are skipped when a workflow process based on the workflow template is initiated, and no error messages appear.

Caution

When you place workflow templates created before Teamcenter 8.3 and 8.1.1.1 in **Edit** mode, the system removes all links originating from the **Start** node. If this occurs, manually redraw any removed links.

20.2.1 Link tasks manually

Drawing a path between two tasks establishes the sequence in the execution of the tasks by declaring that the task on the arrow end of the link cannot start until the task on the start end of the link has been completed.

Manually drawing either success or failure paths between tasks creates explicit links between your tasks. Always explicitly link your tasks to ensure predictable results. Draw your success or failure path immediately after inserting tasks into the workflow process, before saving the workflow process or switching away from the Workflow Designer application. Saving the workflow process or switching applications before manually drawing paths prompts Teamcenter to automatically insert implicit links.

1. On the toolbar, click **Edit Mode** .

2. Click the task node you want to be the predecessor task.

Do not click the title bar of the task node; doing so begins a drag process.

3. Drag your cursor to the task node you want to be the successor task.

A link arrow follows the cursor as you drag. When your cursor moves over a task node, the node is highlighted.

4. Release the mouse button.

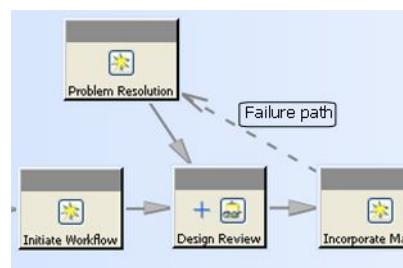
A link arrow connects the predecessor and successor nodes.

20.2.2 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.

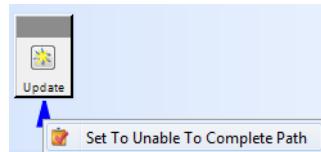
- **Backward branching**

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.

20.2.3 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 | Set to Unable to Complete |
| Review | Set to Reject |
| Route | Set to Reject |
| Condition | Set to Unable to Complete |
| Validate | Set to Error Path |
| EPM | Set to Unable to Complete |

20.3 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).
- Create a query against subprocesses (automatic only).

If there are multiple subprocesses, a query runs on the associated subprocesses and the results are used to branch accordingly. The query is typically configured to look at the root task's *result* attribute for all the subprocesses.

- 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> |

20.3.1 Configuring Condition tasks

Do not have a true path and false path converge on the **Finish** node. Paths are explicitly **AND** tasks and need a successor task at the merge point to complete. Typically, an **Or** task, which is specifically configured to require only one predecessor path to complete for it to start, is used to join the two paths. However, you can also use a **Generic** task or another kind of task.

Do not place a **Condition** task as the last task in a workflow process. The **Finish** node is not a task and should not be linked as a successor task to the **Condition** task.

20.3.2 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 can enter **Production** to indicate the process flowing into a production-ready branch.

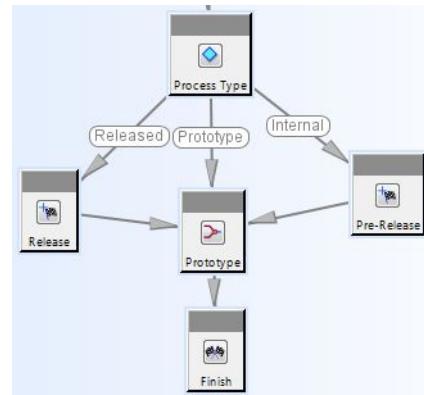
Note

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 the **Condition** task as required for your process.

Example

After creating a production-ready branch, you can create **Design** and **Release** branches by creating additional succeeding tasks and creating additional customized flow lines from the **Condition** task.

20.4 Using Validate tasks

The **Validate** task branches a workflow along two or more paths. The path followed is determined by whether specified errors occur during a workflow. Use this task to design workflows around anticipated errors (such as checked out targets), unexpected errors (such as failed scripts or failure of custom handlers), or to track any and all workflow errors.

Configure the **Validate** task by defining one or more *success* and *failure* paths flowing from the task. The success path is followed if no error occurs. The failure path is followed when errors occur.

When errors occur, you determine if the failure path is followed when:

- Any error occurs.
- Only when an error you specify on a list of error codes occurs.

Note

In the context of the **Validate** task, *workflow error* means any error generated by a workflow handler.

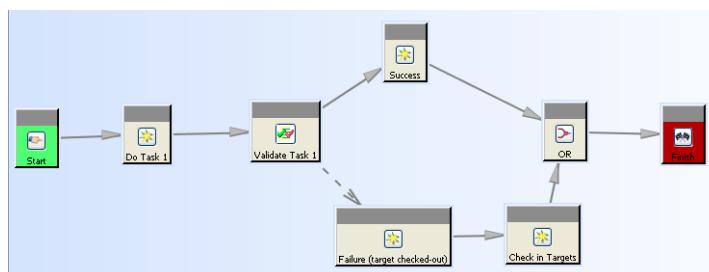
Configure the task to follow a failure path by **pairing** a workflow handler and an error code. Place a handler to be validated on the **Validate** task and then add the respective error code to the path's error list (or set the path to fail on *any* error).

20.4.1 Validate task example: Close gaps in your workflow

At Design, Inc., employees check out documents that are targets of workflows and sometimes neglect to check them back in. Teamcenter does not allow users to initiate a workflow process on a target that is checked out. However, at Design, Inc., no business rules prevent users from checking out targets *after* a workflow process is initiated. When the workflow reaches the review stage, and the required targets are checked out, the workflow cannot complete.

In this example, this situation is anticipated and the **Validate** task is used to provide a correction. The task is placed before the review stage of the workflow and configured to verify that all targets are checked in. If so, a success path is followed. If not, the workflow follows a *failure* path that includes an additional **Do** task assigned to a manager. The **Do** task instructs the manager to get the targets checked in, and then complete the **Do** task. After the error condition is corrected, the **Do** task's success path traverses back into the main workflow.

The **Validate** task is configured to validate whether targets are checked in by placing the **EPM-assert-targets-checked-in** rule handler on the **Start** action, and specifying the **target-checked-out** error in the error list.



When the workflow is run, either the success or failure path is followed, depending on whether the **RES_OBJECT_IS_RESERVED** error is initiated.

20.4.2 Validate task behavior

The **Validate** task's behavior depends upon how its failure path is configured and what errors are received.

| Failure criteria you specified | Error thrown (if any) | Task behavior |
|--|--------------------------------|--|
| Fail if any error | Any error | Failure path is followed. |
| Fail if error on error list occurs | Error on error list | Failure path is followed. |
| Fail if error on error list occurs | Error <i>not</i> on error list | Workflow process halts. Task remains in Started state and an error appears. |
| No failure path configured | Any error | Workflow process stops. Task remains in Started state and an error appears. |
| Regardless of whether failure path was configured, and whether errors occurred | No errors occur | Success path followed. If no success path was configured, workflow process stops. |

20.5 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

20.6 Summary

The following topics were taught in this lesson:

- Import and export process templates.
- Link tasks in a workflow.
- Define and create a failure path.
- Set **Condition** paths.
- Use a **Validate** task to branch a workflow.

Lesson

21 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*.
- 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*.
- You configured *access permissions*.
- You configured *projects*.
- You created *workflow process templates*.

Appendix

A Additional LOV extensions

Purpose

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

Objectives

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

- Create an interdependent LOV.
- Create a batch LOV.

Help topics

Additional information for this lesson can be found in:

- *Creating lists of values (LOVs)*

A.1 Create an interdependent LOV

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

Note

To begin, you must already have a cascading LOV.

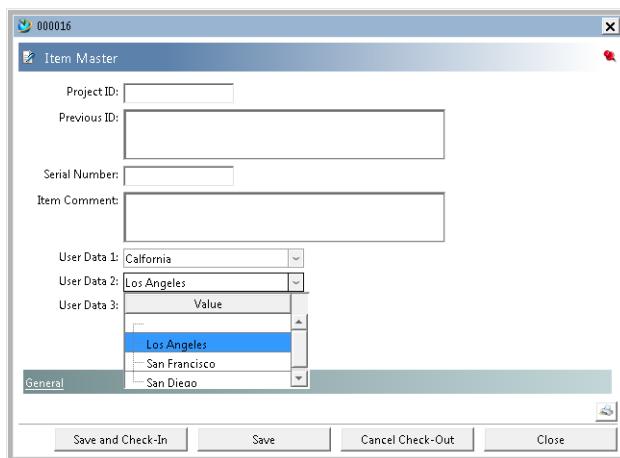
Follow these steps:

1. Right-click the cascading LOV in the **LOV** folder, choose **Open**, and select the **Show Cascading View** check box on the LOV.
2. Ensure that the cascading LOV is attached to a property in the **LOV Attachments** table.
3. Select the attachment in the **LOV Attachments** table and choose **Edit**. The **Interdependent LOV** dialog box appears.
4. Perform the following steps in the **Interdependent LOV** dialog box:
 - a. Select the property and click **Attach** to attach additional properties to it. 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.
 - b. To add a property to describe the attachment, click **Attach Description**. This adds the description of the LOV to the value displayed to the user.
 - c. Click **Finish**.

A.1.1 Test the interdependent LOV

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:



Interdependent LOV in the rich client user interface

Follow these steps:

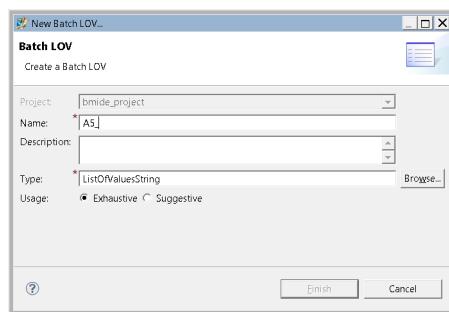
1. Click the button next to the **User Data 1** box to select the state.
2. Click the button next to the **User Data 2** box to select the city.
3. Click **Save and Check-In**.

Note

Open the form and 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.

A.2 Batch lists of values

Defining a batch list of values (LOV) allows you to directly update the LOV values in the Teamcenter database rather than storing them in the template. You can use the **bmide_manage_batch_lovs** utility to update the values for externally managed LOVs from XML files.



New Batch LOV

When you click **Finish**, the Business Modeler IDE creates the following:

1. The new batch LOV appears in the **Batch LOV** folder.
2. A message like the following is displayed in the **Console** view:

Sample data and localization files for your LOV: B5_TestLOV, that can be used as input to `bmide_manage_batch_lovs` utility, can be found at
project\output\Samples\LOVS\B5_TestLOV

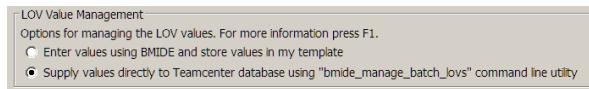
Sample LOV files are placed in the *project\output\Samples\LOVS* folders. These are visible from the **Navigator** view or Windows Explorer. These are sample XML files with sample values in a format required for the **bmide_manage_batch_lovs** utility.

Note

You may store your external LOV files in a different location. The files should be maintained in source control management.

LOV Value Management

Upon opening the new batch LOV, the **LOV Value Management** box shows that the **Supply values directly to Teamcenter database using "bmide_manage_batch_lovs" command line utility** is selected.



LOV Value Management box

To use the **bmide_manage_batch_lovs** utility, you must:

- Place the values in an XML file and accompanying localization XML file. The XML files should conform to the batch utility schema.
- Run the **bmide_manage_batch_lovs** utility, for example:

```
bmide_manage_batch_lovs  
-u=username -p=password -g=group -option=update  
-file=file location\B5_TestLOV.xml.
```

A.2.1 Create batch LOVs

Use the following procedure to create the **Batch LOV**.

1. Choose one of these methods:
 - On the menu bar, choose **BMIDE→New Model Element**, type **Batch LOV** in the **Wizards** box, and click **Next**.
 - Open the **Extensions\LOV** folders, right-click the **Batch LOV** folder, and choose **New Batch LOV**.
2. Type the **Name** and **Description** for the **Batch LOV**.
3. Select the **Type**.

Note

The ability to store LOV values in the database as batch LOVs is available only on the following LOV types: **ListOfValuesChar**, **ListOfValuesDate**, **ListofValuesDouble**, **ListOfValuesInteger**, and **ListOfValuesString**.

4. In the **Usage** section, click one of **Exhaustive**, **Suggestive**, or **Range**.
5. Click **Finish**.

Two results are observed in the Business Modeler IDE.

- a. The new batch LOV appears in the **Batch LOV** folder.
- b. Sample LOV files are placed in the *project\output\Samples\LOVS* folders. These are visible from the **Navigator** view or Windows Explorer.

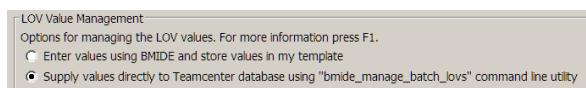
Note

This only creates the **Batch LOV**. After the **Batch LOV** is deployed from the Business Modeler IDE, additional steps are needed to create the properly formatted XML files and the **bmide_manage_batch_lovs** utility.

A.2.2 Create the externally managed LOV values

These steps are performed outside the Business Modeler IDE and will directly update LOV values in the Teamcenter database (rather than storing them in the Business Modeler IDE template). You can use the **bmide_manage_batch_lovs** utility to update the values for externally managed LOVs from XML files.

To be an externally managed LOV, the **LOV Value Management** box in the batch LOV must have the selection **Supply values directly to Teamcenter database using "bmide_manage_batch_lovs" command line utility**.



To update LOV values to be used by your batch LOV, perform the steps:

1. Ensure that the LOV has been deployed to the Teamcenter server.
2. Create or modify the batch LOV XML files.
 - a. Create or modify the batch LOV XML file with your values.
 - b. Create or modify the localization files in the **lang** folder with your values.

Note

Extract LOV values from the external ERP system and place the values in an XML file and accompanying localization XML file. The XML files should conform to the batch utility schema.

3. Run the **bmide_manage_batch_lovs** utility, for example:

```
bmide_manage_batch_lovs
-u=username -p=password -g=group -option=update
-file=project\output\Samples\LOVS\B5_TestLOV\B5_TestLOV.xml.
```

4. After update of externally managed LOVs, if you are using client cache at your site, you must run the **generate_client_meta_cache** utility with the **generate_lovs** command to update the LOV cache stored on the server.

A.2.3 Convert an LOV managed in a template to an externally managed LOV

Typically, values for LOVs are stored in a template, which can then be deployed to a server. But you can also externally manage LOVs and use the **bmide_manage_batch_lovs** utility to update the externally managed LOVs in the database.

1. Open an LOV that is managed in a template.
2. In the **LOV Value Management** box of the LOV, select **Supply values directly to Teamcenter database using “bmide_manage_batch_lovs” command line utility**.



The following dialog box appears.

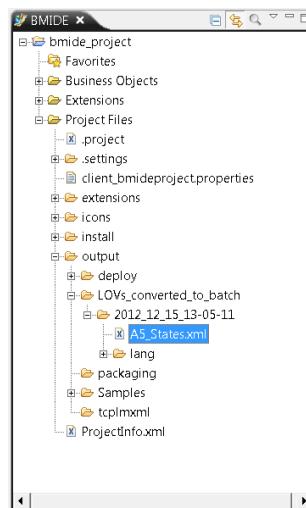


3. Click **OK**.

The LOV values, subLOV attachments, and localizations are saved in an XML file.

4. Click the **Navigator** tab and open the *project\output\LOVs_converted_to_batch\date-and-time* folders to find the XML file. An XML file containing localizations for the LOV is created in a **lang** subfolder.

If you convert multiple LOVs, separate XML files are created for each LOV.



5. Choose **BMIDE→Save Data Model**, then **BMIDE→Package Template Extensions**, and install the template to the server. This deletes all the LOV values, localizations, and subLOV attachments in the database for the LOVs that were converted.
6. Run the **bmide_manage_batch_lovs** utility to update the values in the XML files to the database, for example:

```
bmide_manage_batch_lovs.bat -u=username -p=password -g=dba
-option=update -file=A5_test_LOV.xml
```

7. Now that the LOV values are stored in the database, you can manage the values through the **bmide_manage_batch_lovs** utility.

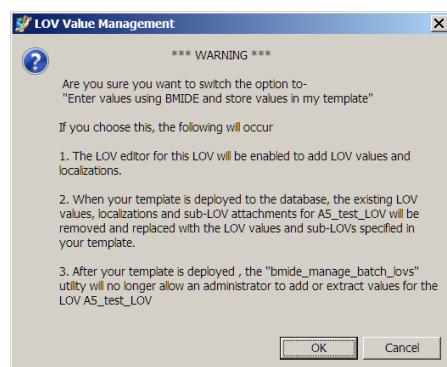
A.2.4 Convert an externally managed LOV to an LOV managed in a template

Externally managed LOVs can be converted to LOVs managed in a template.

1. Open an LOV that is externally managed.
2. In the **LOV Value Management** box of the LOV, select **Enter values using BMIDE and store values in my template**.



The following dialog box appears.



3. Click **OK** in the dialog box.

The **Add** button in the user interface is enabled to allow you to add values to the LOV.

4. Add values to the LOV.

Because the LOV values for this LOV are still in the database, you can extract them using the **bmide_manage_batch_lovs** utility. To extract LOV values, subLOV attachments, and localizations for specific LOVs in the database, enter a command like the following on a single line:

```
bmide_manage_batch_lovs -u=username -p=password -g=dba
-option=extract -lovs=BatchLOV_LOV1,BatchLOV_LOV2
-file=BatchLOV_extracted.xml
```

You can then use the extracted values as the basis for new values to enter in the template-managed LOV.

5. Choose **BMIDE→Save Data Model**, then **BMIDE→Package Template Extensions**, and install the template to the server.

Note

After updating the template to the server, these changed LOVs are not available for extraction or update using the **bmide_manage_batch_lovs** utility.

A.2.5 Considerations for externally managing LOVs

Instead of managing LOVs in a template, you can manage LOVs externally through the use of the **bmide_manage_batch_lovs** utility.

The following cases are supported:

- You can only use the following types of LOVs: **ListOfValuesChar**, **ListOfValuesDate**, **ListofValuesDouble**, **ListOfValuesInteger**, and **ListOfValuesString**
- You can only use the **Exhaustive** and **Suggestive** LOV usages. You cannot use the **Range** usage.
- You can attach an externally managed LOV as a subLOV to a LOV that is managed in a Business Modeler IDE template. This can be done using the Business Modeler IDE but not using **bmide_manage_batch_lovs** utility.
- An externally managed LOV can have subLOVs that are either managed in a Business Modeler IDE template or externally managed. This can be done only through the **bmide_manage_batch_lovs** utility but not using the Business Modeler IDE.

The following cases are not supported:

- When a **ListOfValuesString** type LOV is externally managed, referencing a property of another class is not allowed because the LOV values are updated through the utility.
- You cannot convert an LOV that you do not own in your template. For example, COTS LOVs cannot be converted to externally managed LOVs, and vice versa.
- Interdependent LOV attachments are not supported on externally managed LOVs.
- An LOV cannot be converted to an externally managed LOV if its value is referenced in any other element like a verification rule or note type
- An LOV cannot be converted to an externally managed LOV if it is used as a based-on LOV in a filter LOV.

A.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

A.4 Summary

The following topics were taught in this lesson:

- Create an interdependent LOV.
- Create a batch LOV.

Appendix

B Compound property configuration

Purpose

The purpose of this lesson is to define *compound properties* to display Teamcenter information in more convenient locations for end users.

Objectives

After you complete this lesson, you should be able to define a compound property:

- On an item revision from a form property.
- In a Structure Manager column from an item revision.
- In a Structure Manager column from checked-out dataset.

Help topics

Additional information for this lesson can be found in:

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

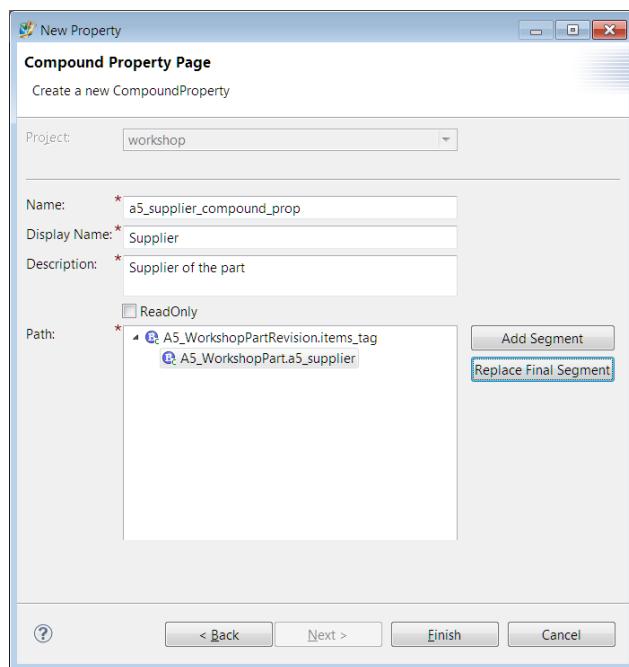
B.1 Compound properties

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).

The display object and source object are related by one or more Teamcenter relations and reference properties. The relationship between the display object and source object can be a reference relation, a GRM relation, or a combination of the two. Compound properties behave as a property of the display object type. Compound property rules can be inherited by children of the source business object.

Each Teamcenter business object has a set of predefined properties and associated properties. Depending on the business and process requirements of your company, properties may need to be defined for the business objects.

Compound properties allow you to add new properties to business objects without writing custom code.

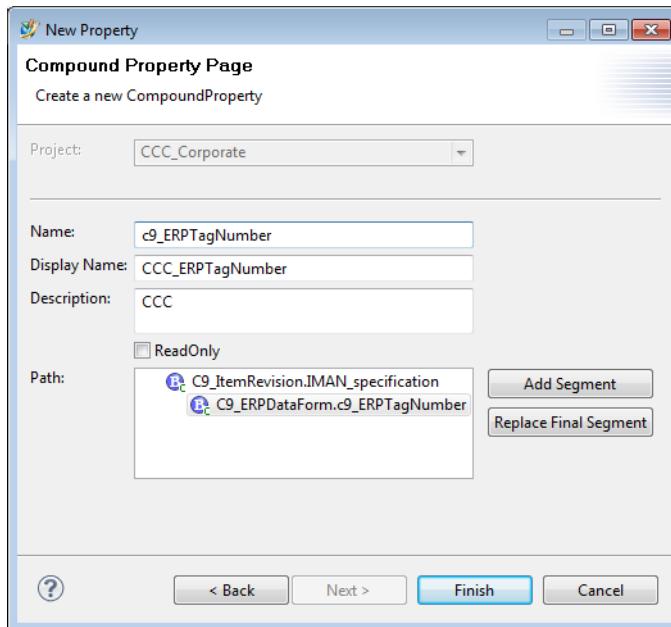


Key points

- Once configured, compound properties apply to properties as displayed in both the rich client and thin client interfaces.
- Compound properties defined on base business objects are inherited by sub-business objects. For example, a compound property defined on the **CCC_CommonItemRevision** business object is inherited by the **CCC_ItemRevision** business object.

B.1.1 Working with compound properties

A compound property uses **Relation** and **Reference** properties to traverse from the source to the destination object.



By using **Add Segment**, 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.

Rule pathRule path

The table shows an example to traverse the relation and child objects to find the target property.

safety_code (compound property)

| Object | Traverse | Child object | Target property |
|--------|------------------|----------------|-----------------|
| MyPart | IMAN_requirement | MySupplierForm | safety_code |

B.1.2 Add a compound property

Adding a compound property requires a path to display the source object's property on the display object, if the path exists for the two objects. The path traverses relation(s), reference(s) and business objects persistent business object properties.

1. In the **Business Objects** view, browse to and open the business object to which you want to add the compound property. Click the **Properties** tab in the resulting view.
2. Click the **Add** button to the right of the properties table.
3. Under **Property Types**, select **Compound**. Click **Next**.
4. Perform the following steps in the **Compound Property Page** dialog box:
 - a. Type the **Name** you want to assign to the new compound property.
 - b. Type the **Display Name** as you want it to appear in the user interface.
 - c. In the **Description** box, type a description of the compound property.
 - d. (Optional) Select the **ReadOnly** check box.
 - e. Click the **Add Segment** button.
5. Perform the following steps in the **Compound Property Segment** dialog box:
 - a. Select the first segment of the compound property. Click **Next**.
 - b. At the **Choose a business object** prompt, select the business object for the next segment. Click **Finish**. The **Compound Property Page** dialog box appears, showing the compound property path thus far.
6. 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**.
7. To add the last segment, click the **Add Final Segment** button. In the **Compound Property Segment** dialog box, select the property for the final segment.

B.1.3 Example – one segment rule path

To define a compound property, the segments of the path must be determined based on the data objects and relation between them. In the final segment, the data object property must be chosen.

This example shows a compound property to be defined on **MyPart**.



safety_code (compound property)

supplier (compound property)



safety_code

supplier



MyPartRevision

Rule path

The table shows how to traverse the relation and child objects to find the target property.

safety_code (compound property)

| Object | Traverse | Child object | Target property |
|---------------|-------------------------|-----------------------|--------------------|
| MyPart | IMAN_requirement | MySupplierForm | safety_code |

supplier (compound property)

| Object | Traverse | Child object | Target property |
|---------------|-------------------------|-----------------------|-----------------|
| MyPart | IMAN_requirement | MySupplierForm | supplier |

B.1.4 Example – two segment rule path

This example shows a compound property to be defined on **MyPart** where the value requires two segments to reach the **MyMaterialForm**.



weight (compound property)



MyMaterialForm (IMAN_specification relation to MyPartRevision)

weight

Rule path to weight (compound property)

The table shows how to traverse the relation and child objects to find the target property.

There are two segments as numbered in the first column.

Compound Property Segment

| | Object | Traverse | Child object |
|---|--------|---------------|----------------|
| 1 | MyPart | revision_list | MyPartRevision |

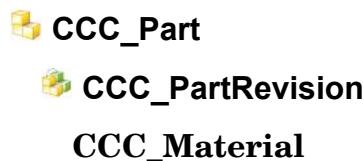
Final segment

| | Object | Traverse | Child object | Target property |
|---|----------------|--------------------|----------------|-----------------|
| 2 | MyPartRevision | IMAN_specification | MyMaterialForm | weight |

B.1.5 Example – Structure Manager rule path

This example shows a compound property to be defined on **BOMLine**, which is shown in Structure Manager. The **CCC_Material** is a property of the **CCC_PartRevision**.

My Teamcenter



Structure Manager

| Object | Type | CCC_Material |
|--------|------------------|--------------|
| Assm0 | CCC_PartRevision | |
| Comp1 | CCC_PartRevision | Steel |
| Comp2 | CCC_PartRevision | Steel |

Rule path

The table shows how to traverse the relation and child objects to find the target property.

CCC_Material (compound property)

| Object | Traverse | Child object | Target property |
|---------|-------------|------------------|-----------------|
| BOMLine | bl_revision | CCC_PartRevision | CCC_Material |

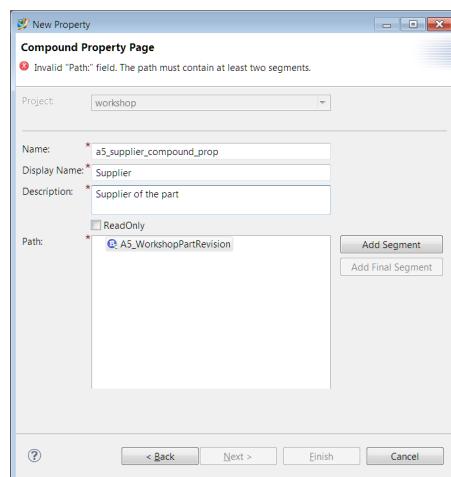
B.2 Add a compound property example steps

You can use a compound property to display a custom property from a form on a business object. A compound property uses **Relation** and **Reference** properties to traverse from the source to the destination object.

1. In the **Business Objects** folder, browse to the business object to which you want to add the compound property.
 - a. Search for a business object with the **Find** button.
 - b. Right-click the business object, choose **Open**.
 - c. Click the **Properties** tab in the resulting view.
2. Click the **Add** button to the right of the properties table.
3. Under **Property Types**, select **Compound** and click **Next**.

Using the compound property dialog boxes

1. Provide the compound property information.



- a. In the **Name** box, type the name of the new compound property.
- b. In the **Display Name** box, type the name as you want it to appear in the user interface. Enter the display name so it is identical to the display name on the originating property.
- c. In the **Description** box, type a description of the compound property.
- d. (Optional) Select the **ReadOnly** check box.

2. Add a compound property segment.

- a. Click the **Add Segment** button.

The Add Compound Property Segment wizard runs, displaying the **Choose a property** message.

- b. Select the property on the target business object to hold the compound property, such as **items_tag**. Use the **Reference**, **Relation**, and **Runtime** check boxes to filter out the type of properties to display.

- c. Click **Next**.

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

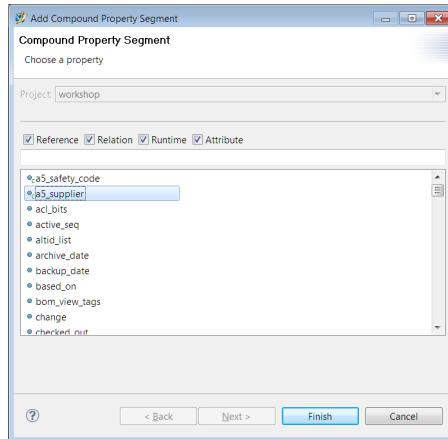
- d. Select the business object that is the source for the property.

- e. Click **Finish**.

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

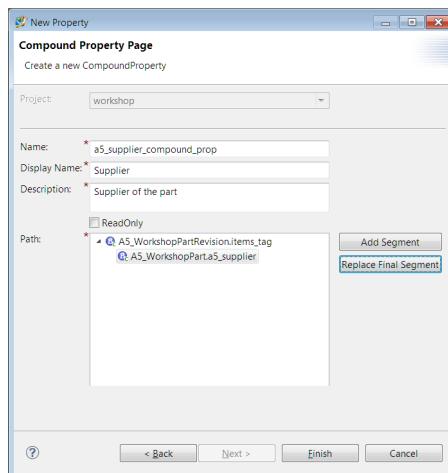
To add more segments to the compound property, click the last segment in the **Path** pane and click **Add Segment**.

3. Add the final segment.
 - a. Click the **Add Final Segment** button.
 - b. Select the property you want to use for the final segment of the compound property.



- c. Click **Finish**.

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



- d. When you are done adding segments in the **Compound Property Page** dialog box, click **Finish**. The new compound property appears in the property table.
- e. (Optional) After you add a property, change characteristics of the property by using property constants.

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.

B.3 Summary

The following topics were taught in this lesson:

- On an item revision from a form property.
- In a Structure Manager column from an item revision.
- In a Structure Manager column from checked-out dataset.

Appendix

C Additional rules and operations

C.1 Additional rules and operations

Purpose

The purpose of this lesson is to define rules for Generic Relationship Management (GRM) and operation extensions.

Objectives

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

- Define a create operation on an item to also create a form.
- Create Generic Relationship Management (GRM) rules.

Help topics

Additional information for this lesson can be found in:

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

C.1.1 Using operation extensions

To see extensions defined for business objects, right-click a business object, 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.

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.

C.1.1.1 Predefined extensions

Following are a few of the predefined extensions rules that can be viewed with the **Operations** tab.

- **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.

- **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.

- **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.

C.1.1.2 About extension attachments

To see extensions operations defined for business objects, right-click a business object, choose **Open**, and click the **Operations** tab in the resulting editor. The available operations for the business object are found in the **Operations** folder, and available operations for the properties are found in the **Properties** 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. Extension points include:

- **Pre-Condition**

Places limits before an action. For example, limit individual users by their work context to create only a certain item type. Pre-conditions are executed first in an operation dispatch. If any of the pre-conditions fails, the operation is aborted. Typical examples of pre-conditions are naming rules.

- **Pre-Action**

Executes code before an action. For example, add user information to the session prior to translation. Pre-actions are executed after pre-conditions and before the base action. If any of the pre-actions fail, the operation is aborted. A typical example is an initial value rule that needs to set an initial value before the save base action is invoked.

- **Base-Action**

Executes code for an action. The base action is the actual operation implementation, and cannot be replaced. **Base-Action** is used only for user exits operations.

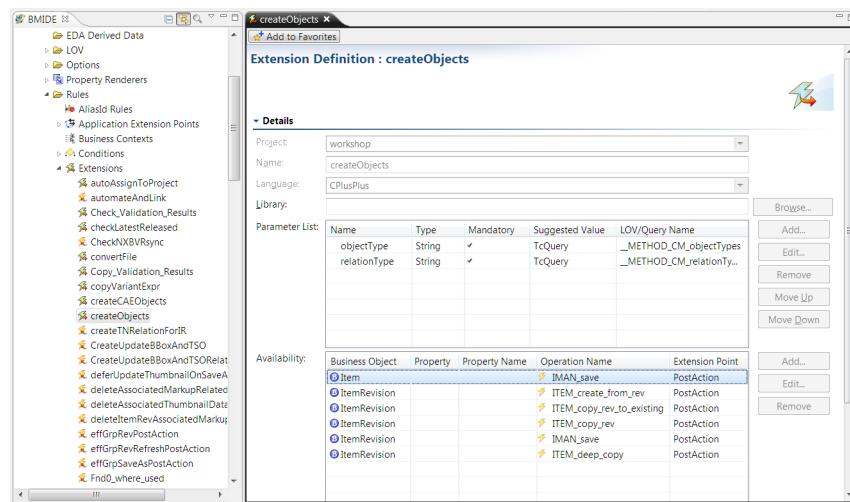
- **Post-Action**

Executes code after an action. For example, automatically start an item in a workflow. If any of the post-actions fail, the operation is aborted.

C.1.1.3 View predefined extensions to a business object

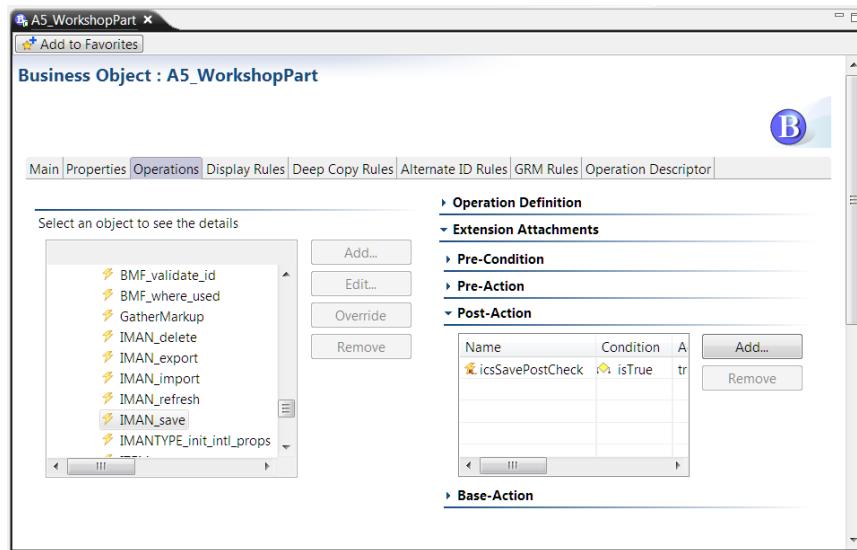
The details of the extension appear in a new **Extension Definition** view.

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.



Confirming availability of the extension

1. Open the **Advanced** perspective by choosing **Window→Open Perspective→Other→Advanced**.
2. In the **Extensions** view, expand the project and the **Rules\Extensions** folders.
3. Double-click the predefined extension definition you want to use, for example, **createObjects**.



Selecting an operation on the business object

4. 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** and **Legacy Operations** folders.
5. Select the operation in the **Operations** or **Legacy Operations** folder and click the **Extensions Attachments** link on the lower right side of the editor.

C.1.1.4 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 **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.
2. Select the operation in the **Operations** folder and click the **Extensions Attachments** link on the lower right side of the editor.
3. Click the **Add** button next to the extension point (**Pre-Condition**, **Pre-Action**, **Base-Action**, or **Post-Action**).

The Add Extension Rule wizard runs.

4. In the **Extension** dialog box, click the **Browse** button to the right of the **Extension** box and choose the extension.
5. 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.1.1.5 The **autoAssignToProject** extension

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.

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. When you assign an object to a project automatically, the project becomes the owning project.

The following object types can be configured for automatic assignment:

- Item and item revision subtypes
- 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.

Your current project or program is defined in the **User Settings** dialog box. You can choose **Edit→User Setting** to change your current project or program.

Note

To remove an object from the owning project, add the **TC_allow_remove_owning_project** preference and set it to **true**, which allows you to right-click the object and choose **Project→Remove**.

C.1.1.6 Using the **autoAssignToProject** extension

The **autoAssignToProject** extension automatically assigns the selected workspace object to the user's current project, as defined by the work context or user settings.

The following table describes the types and operations for which the **autoAssignToProject** extension is valid. The valid extension point is **PostAction**.

| Type | Operation |
|---|--|
| Item and all subtypes of item | IMAN_import IMAN_save ITEM_create_from_rev |
| Item revision and all subtypes of item revision | IMAN_save ITEM_baseline_rev ITEM_copy_rev ITEM_copy_rev_to_existing |
| Dataset and all subtypes of dataset | AE_save_dataset |
| Form and all subtypes of form | IMAN_save |

Configuring the **autoAssignToProject** internal extension for a business object has implications on the project propagation rules. Project propagation rules determine which secondary objects are assigned to a project when a primary business object is assigned. When there is a conflict between a propagation rule and the execution of the **autoAssignToProject** extension, the extension takes precedence.

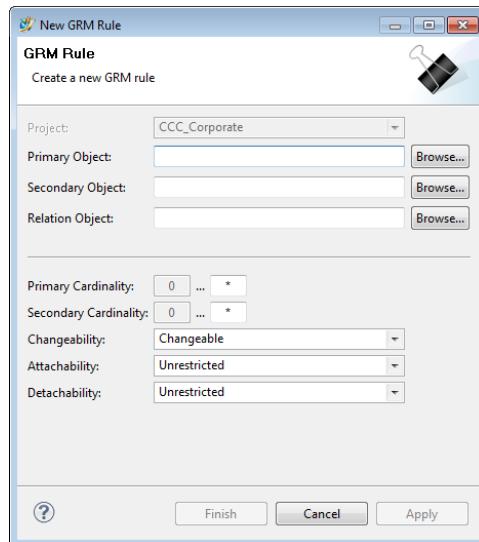
Note

If a current project is not specified for the user, this extension is ignored and the object is not automatically assigned. In addition, when the **autoAssignToProject** extension is configured for an item or engineering change order (ECO), the project name is preselected in the **Assign to Projects** page of the item or ECO create, revise, and save as dialog boxes.

If you want users to be able to remove objects from an owning project, you must create the **TC_allow_remove_owning_project** preference before using the **autoAssignToProject** extension. If this preference is not set, objects assigned to owning projects cannot be removed using the **Project→Remove** command.

C.1.2 Working with GRM rules

A Generic Relationship Management (GRM) rule applies *constraints* on the relationship between two Teamcenter objects.



You can use the GRM rules to limit what objects can be pasted to other 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.

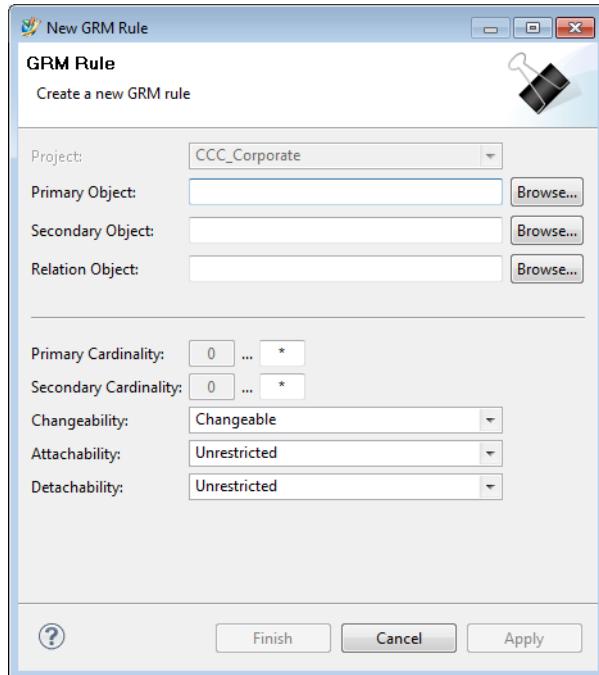
Example constraints

- Limit the number of objects to be pasted (cardinality).
- Allow unrestricted cut (detachability) or paste (attachability).
- Not allow an object to be cut (changeability).
- Not allow an object to be changed in any way (changeability).

Note

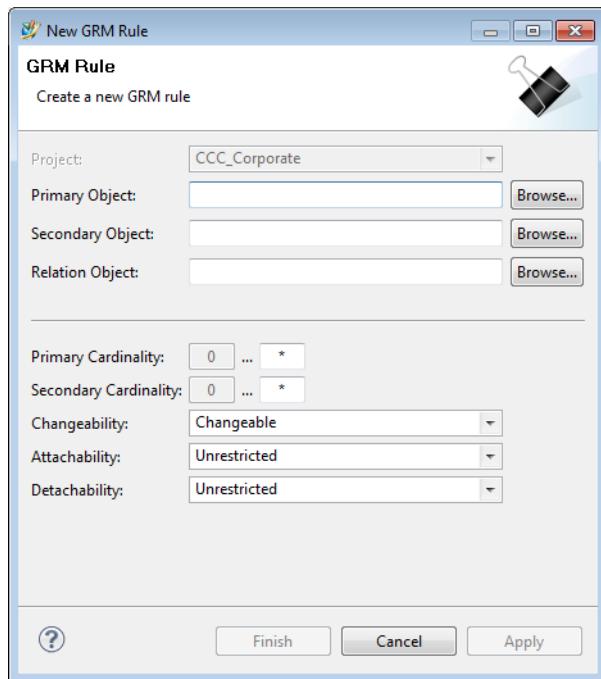
You can see the relations between business objects in the UML editor by right-clicking and choosing **Show→Relations**.

C.1.2.1 GRM rule definitions and constraints



| Setting | Description |
|------------------|---|
| Primary object | The primary business object in the relationship. |
| Secondary object | The secondary business object in the relationship. |
| Relation object | The relationship between the primary and secondary business objects. These relations are children of the IMANRelation business object. |

- Primary cardinality
The number of primary objects that can be related to a secondary object with the relationship. An asterisk (*) means an unlimited number.
- Secondary cardinality
The number of secondary objects that can be related to a primary object with the relationship. An asterisk (*) means an unlimited number.



Changeability

- Select **Changeable** if the relationships using this rule can be deleted or otherwise changed.
- Select **Add Only** if only new relationships can be made between the objects.
- Select **Frozen** if relationships cannot be changed in any way.

Attachability

- Select **Unrestricted** if all users can relate new objects using the rule.
- Select **Write Access Req.** if Access Manager rules should be used to evaluate if the relationship creation is allowed.

Detachability

- Select **Unrestricted** if all users can relate new objects using the rule.
- Select **Write Access Req.** if Access Manager rules should be used to evaluate if the relationship creation is allowed.

C.1.2.2 GRM rule inheritance

The GRM rule applies for all children of the primary and secondary objects.

For example:

ItemRevision business object is chosen as the primary object.

MSWord dataset is chosen as the secondary object

All children of these objects that have the relationship, inherit the rule. However, rules defined for a specific sub-business object take precedence over the relation rules defined for a parent object.

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.

C.1.2.3 GRM rule examples

Example 1

Create an **Item** and put a **MSWord** dataset on the **Item Revision** with the **IMAN_specification** relationship. Try to create another **MSWord** dataset on the same **Item Revision**. An error message states that you cannot do this because it violates **GRM** rule constraints that allow only one **MSWord** object to be related.

| GRM rule setting | Criteria |
|------------------------------|---------------------------|
| Primary object | Item Revision |
| Secondary object | MSWord |
| Relation object | IMAN_specification |
| Primary cardinality | 0 ... 1 |
| Secondary cardinality | 0 ... 1 |
| Changeability | Changeable |
| Attachability | Unrestricted |
| Detachability | Unrestricted |

Example 2

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.

C.1.2.4 Add a GRM rule

1. Select the file where you want to save the data model changes, for example, **rules.xml**.
2. In the **Business Objects** view, right-click a business object and choose **Open GRM Rules Editor**.

The **GRM Rules** editor displays the active GRM rules in the project.

3. To create a GRM rule between two objects, click the **Add** button.
The GRM Rule wizard runs.
4. Perform the remaining steps in the **GRM Rule** dialog box.

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.

Perform the following steps in the **GRM Rule** dialog box:

1. Use the **Browse** buttons to:
 - Select the **Primary Object** in the relationship.
 - Select the **Secondary Object** in the relationship.
 - Choose the **Relation Object** between the primary and secondary business objects.
2. Set **Primary Cardinality**.
3. Set **Secondary Cardinality**.
4. Set **Changeability** to one of **Changeable**, **Add Only**, or **Frozen**.
5. Set **Attachability** to one of **Unrestricted**, or **Write Access Req.**
6. Set **Detachability** to one of **Unrestricted**, or **Write Access Req.**
7. Click **Finish**.

The rule is created and appears in the table on the **GRM Rules** editor.

C.1.3 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

C.1.4 Summary

The following topics were taught in this lesson:

- Define a create operation on an item to also create a form.
- Create Generic Relationship Management (GRM) rules.

Appendix

D Rich client interface configuration

Purpose

The purpose of this lesson is to add codeless rich client configuration to the display of custom form properties with style sheets.

Objectives

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

- Display of custom form properties on a separate form tab.

Help topics

Additional information for this lesson can be found in:

- *Rich Client Interface Guide*

D.1 **Property display using XML style sheets**

Using XML style sheets enhances the display of forms and properties in the Teamcenter rich client and thin client interface. You can customize the display by editing the style sheet.

The advantages are:

- You can customize using configuration instead of coding.
- The customization affects both the rich client and thin client.

The style sheet is an XML document stored in a **XMLRenderingStylesheet** dataset. This gives more control to sites regarding how dialog boxes are displayed. The XML code allows sites to define a subset of properties to display, the display order, the user interface rendering components to be used, and more. Sites can use XML code to customize not only forms but also individual fields in the forms.

D.1.1 Using predefined style sheets

The Teamcenter installation provides several predefined style sheets that are registered to display the properties of the following objects:

- Folder
- Form
- Item
- Item revision
- Group
- User
- Role
- Site
- Group member
- Volume
- EPM job
- ImanFile
- Tool
- Reservation

The registration information is stored in the preference; each object type has two entries used to display regular properties, as follows:

- *business-object-name.RENDERING*

The value of this key is the dataset name or the dataset UID used to display properties of this type of object.

- *dataset-name/uid.REGISTEREDTO*

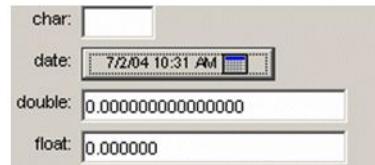
The value of this key is the type name for which this dataset is registered.

D.1.2 Display format

There are two display formats.

OneColumn

This is the format used in the current property display. Each row displays one property. This is the default format if no format is defined.



OneColumn display format

TwoColumn

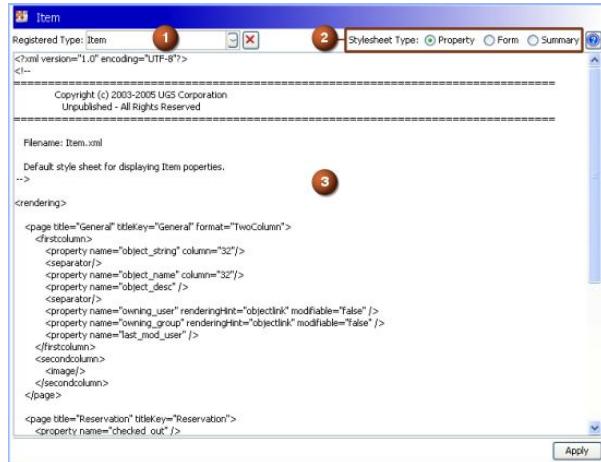
Each row displays two properties.



TwoColumn display format

D.1.3 Accessing the style sheet viewer

The following figure and table describe the viewer components.



Style sheet viewer

- | | | |
|---|--------------------------------|--|
| 1 | Registered Type list | Registers the XML style sheet to a type. |
| 2 | Stylesheet Type buttons | Applies the style sheet type to a particular type of object. |
| 3 | Text editor box | Allows you to modify the XML definition. |

To see the style sheet in the **Viewer** tab, select an **XMLEnRenderingStylesheet** dataset in My Teamcenter. Click the **Viewer** tab. If you have **DBA** access, you can use the viewer to create, modify, and register the XML definition for a certain type object. If you do not have **DBA** access, you can view the XML definition, but you cannot modify it.

Registration information style sheet is stored in a site preference. The scope of the preference is read as **ALL**, which allows you to manually modify the registration to apply to a specific, user, group, or role, and those preferences are read accordingly.

D.1.4 Examples of style sheet definitions

The following figure shows an example of an XML definition for item properties.

```
<?xml version="1.0" encoding="UTF-8"?>

<rendering>
  <page title="General" titleKey="General" format="TwoColumn">
    <firstcolumn>
      <property name="object_string" column="32"/>
      <separator/>
      <property name="object_name" column="32"/>
      <property name="object_desc" />
      <separator/>
      <property name="owning_user" />
      <property name="owning_group" />
      <property name="last_mod_user" />
    </firstcolumn>
    <secondcolumn>
      <image/>
    </secondcolumn>
  </page>
  <page title="Reservation" titleKey="Reservation">
    <property name="checked_out" />
    <property name="checked_out_user" />
    <separator/>
    <property name="checked_out_date" />
    <property name="checked_out_change_id" />
    <separator/>
    <property name="reservation" />
  </page>
  <page title="Project" titleKey="Project">
    <property name="proj_assign_mod_date" />
    <property name="project_ids" />
    <separator/>
    <property name="project_list" />
  </page>

  <page title="CCC_Props" titleKey="CCC_Props">
    <property name="CCC_Customer" />
    <property name="CCC_MakeBuy" />
  </page>

  <page title="All" titleKey="All">
    <all type="property"/>
  </page>
</rendering>
```

Item properties XML definition

D.1.5 Modify a predefined style sheet

To modify the predefined style sheets, you can either modify the existing dataset or create a new dataset using the **Save As** command.

1. In My Teamcenter, perform a search for the dataset you want to modify. The dataset type is **XMLRenderingStylesheet**.
2. To create a new dataset, select the **XMLRenderingStylesheet** dataset and save it as a new dataset by choosing **File→Save As**.
Teamcenter displays the new dataset.
3. Select the dataset and click the **Viewer** tab.
Teamcenter displays the contents of the XML file in the style sheet viewer.
4. Select the object type to which this style sheet will be registered from the **Registered Type** list.
5. Select one of the style sheet type options, either **Property** or **Form**.
6. Edit the XML file, as required.
7. Click the **Apply** button to save the modifications.
8. Select an object that uses the style sheet and display the properties of the object or open the form to verify the modifications to the style sheet.

D.1.6 Creating form preferences for new business objects

To create a style sheet for a new business object, you create the following preferences:

<dataset_name>.REGISTEREDTO

<type_name>.RENDERING

Note

The **<type_name>** represents a business object name.

For example, for a new business object named **MyItem**, you must create the following preferences to use a specific style sheet (dataset) for a **MyItem** business object:

- **MyItemDatasetSS.REGISTEREDTO**

Where **MyItemDatasetSS** is a **XMLRenderingStylesheet** dataset typically created with a **File→Save As** from an existing **XMLRenderingStylesheet** dataset.

- **MyItem.RENDERING**

Where **MyItem** is a new item business object.

D.1.7 Verify the registration of forms in the rich client interface

1. In one of the rich client applications, choose **Edit→Options**. Teamcenter displays the **Options** dialog box.
2. Click the **Index** link in the lower-right portion of the window.
Teamcenter displays the **Preferences** pane.
3. Type **form** in the **Search on preference name** box and click the **Search** button .
4. Click the **form.REGISTEREDTO** entry in the **Preferences list**.
Teamcenter displays the business objects to which the **Form** stylesheet can be applied in the right pane of the window.

If no rendering registration is found for the currently selected type, the system searches the hierarchy to determine if a rendering is registered to any of the parent classes or parent types. If one is found, that rendering is honored.

D.2 Teamcenter utility and preferences

This section describes the following Teamcenter utility and preferences:

- Teamcenter preference: **<dataset_name>.REGISTEREDTO**
- Teamcenter preference: **<type_name>.RENDERING**
- Teamcenter utility: **import_file**

<dataset_name>.REGISTEREDTO

DESCRIPTION

Teamcenter property displays are based on an XML style sheet. The style sheet must be created and registered by a system administrator. This preference is referenced by the style sheet; when an XML style sheet dataset is selected for display in the viewer, the dataset type defined in this preference is used. For each **type_name.RENDERING** preference defined, a corresponding **dataset_name.REGISTEREDTO** preference must be defined.

VALID VALUES

Accepts a single string as a value. It must be the type name defined in the corresponding **type_name.RENDERING** preference.

DEFAULT VALUES

None.

This preference must be imported or manually entered into the database.

SCOPE

Site preference.

<type_name>.RENDERING

DESCRIPTION

Teamcenter property displays are based on an XML style sheet. The style sheet must be created and registered by a system administrator. This preference determines the XML rendering for displaying the defined type's properties. If this preference is defined, the **<dataset_name>.REGISTEREDTO** preference must also be defined.

VALID VALUES

Accepts a single string as a value. It must be a valid dataset name, which contains rendering XML.

DEFAULT VALUES

None.

This preference must be imported or manually entered into the database.

SCOPE

Site preference.

import_file

DESCRIPTION

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.

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= p= g=**

The user, password and group. This user and group are the owning user and owning group.

f

Imports a single file into Teamcenter.

i

Imports multiple files into Teamcenter using a specified import file.

vb

Runs utility in verbose mode.

log

Creates a log of items and datasets created.

type

Defines the dataset type in Teamcenter, for example, **TEXT**.

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** type datasets have named references of **TEXT**. Each dataset type defines one or more named references to be associated with it.

de

Indicates that a dataset exists.

n

Specifies that a new dataset be added even if one with the same name exists.

e

Specifies that the dataset should be added if it already exists.

a

Specifies that the imported file be added as a named reference to the existing dataset.

r

Specifies that a new dataset revision be created and the existing named reference be replaced with the new one.

item

Specifies the name of the item containing the dataset that references the imported file.

revision

Specifies the item revision number and revision ID.

ie

Specifies behavior if the item already exists.

n

Specifies that the dataset is not added if the item already exists.

y

Specifies that the dataset may be added if the item already exists.

desc

Specifies a user-defined text description of an item that is created by the import function.

v

Specifies the full path of the Teamcenter volume where the imported file is placed.

EXAMPLES

- 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=MSExcel -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
```

- 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
```

D.3 **Summary**

The following topics were taught in this lesson:

- Display of custom form properties on a separate form tab.

Appendix

E Key preferences reference

E.1 Key preferences reference

The following are important preferences to be familiar with in Teamcenter:

Administration preferences

- **TC_current_role**

Specifies whether the user's membership in a role enforces access to objects.

Warning

If this is set to **false**, a member of the role has access to an object even when the role is not the user's **current** role. Be careful when testing a workflow, for instance, and recognize that a user could be getting permissions outside his current role.

Client preferences

- **WEB_default_site_server**

Sets the server URL to be included in Teamcenter mail messages.

- **ItemRevision_shown_relations**

Determines which relations display within an item revision.

Configuration preferences

- **DATASET_saveas_pattern**

Specifies a pattern used to generate and return the output name of nonmaster items during a **SaveAs** operation of their master item.

Data management preferences

- **AE_dataset_id_usage**

Determines whether dataset identification is enabled. If this is set to **ON**, datasets have both IDs and revisions.

Note

This means that the dataset can be revised independent of the item revision it may be related to, or even without it being related to any item revision.

File caching preferences

- **Fms_BootStrapUrls**

Determines which FMS server cache manages file downloads.

- **Transient_Volume_Installation_Location**
Specifies the node name of the transient volume.
- **Default_Transient_Server**
Specifies the default transient file server location.

Workflow preferences

- **EPM_enable_apply_template_changes**
Determines whether edits to a workflow template can be applied to all active workflow processes based on the template.
- **CR_allow_alternate_procedures**
Determines whether users can select alternate workflow processes from the **New Process Dialog** dialog box using the **Process Template Filter** list.
- **EPM_multiple_processes_targets**
Determines whether to allow multiple processes for the same objects.
- **EPM_resource_pool_restrict_subscription**
Determines whether users' subscription access to resource pools is restricted. If restricted, users can only subscribe to resource pools for groups/roles of which they are a member.
- **INBOX_hide_suspended_tasks**
Determines how a user's inbox displays tasks when they are in either the **Suspend** or **Resume** state.
- **SIGNOFF_required_group_and_role**
Determines whether a user must be logged in to the correct group/role to perform an assigned **perform-signoffs** task.

Appendix

F Additional Teamcenter security

Purpose

The purpose of this lesson is to provide an overview of the Teamcenter security applications and features.

- Authorized data access (ADA) compliance
- Intellectual property (IP) protection
- International Traffic in Arms Regulations (ITAR)
- Access Manager
- In-work (Workflow) data security

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*

F.1 **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.

F.2 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.

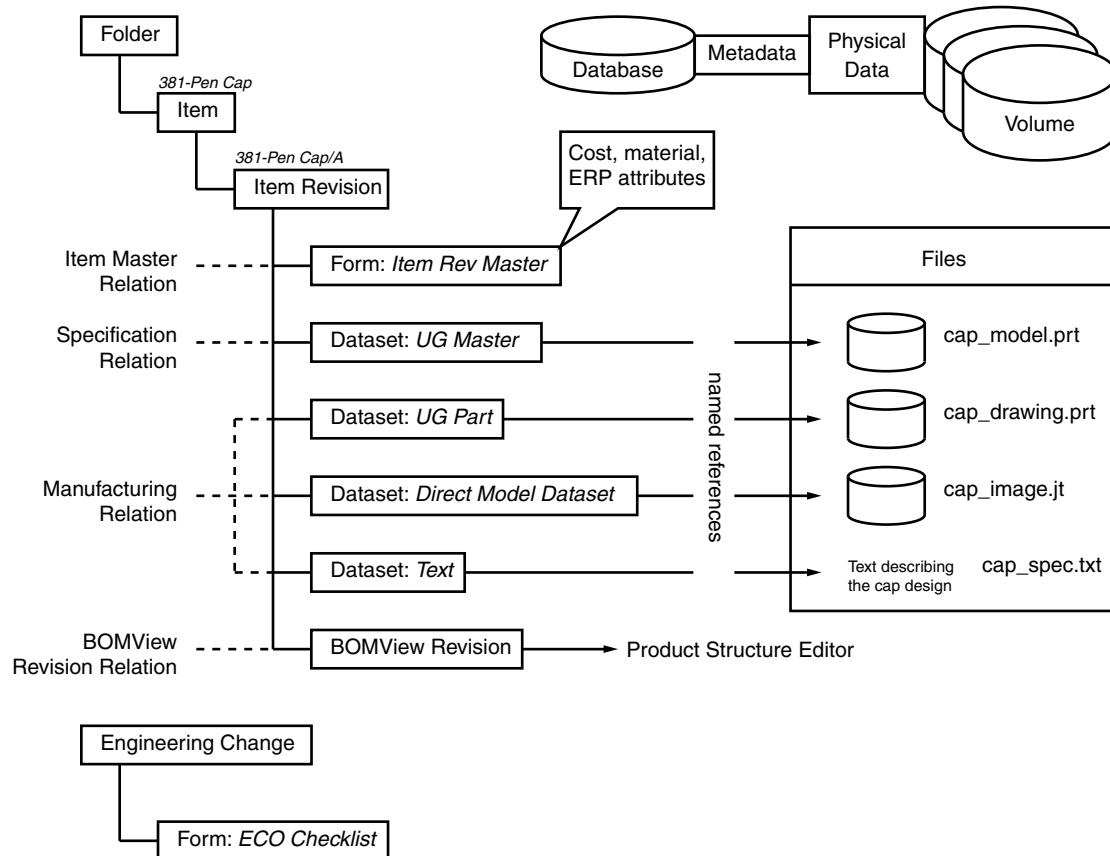
F.3 Basic concepts

Teamcenter security administration requires an understanding of:

- Object model hierarchy
- Authentication
- Authorization

F.4 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.



Note

Master forms inherit their permissions from their parent item or item revision. You can use the **TC_MASTERFORM_DELEGATE** environment variable to change the default behavior.

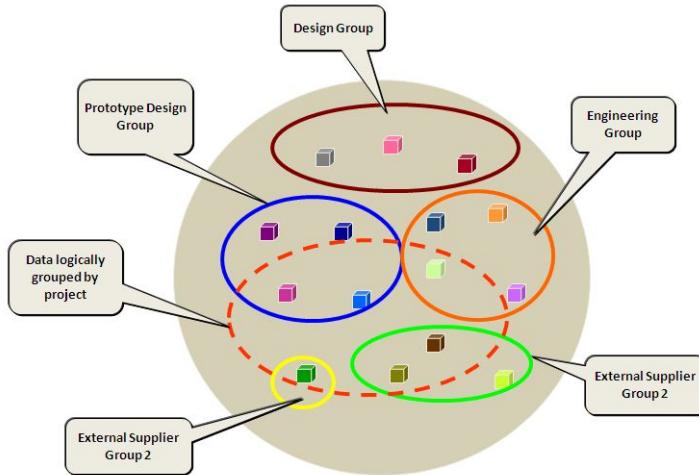
F.5 **Authentication**

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.

F.6 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.

F.7 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.

F.8 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

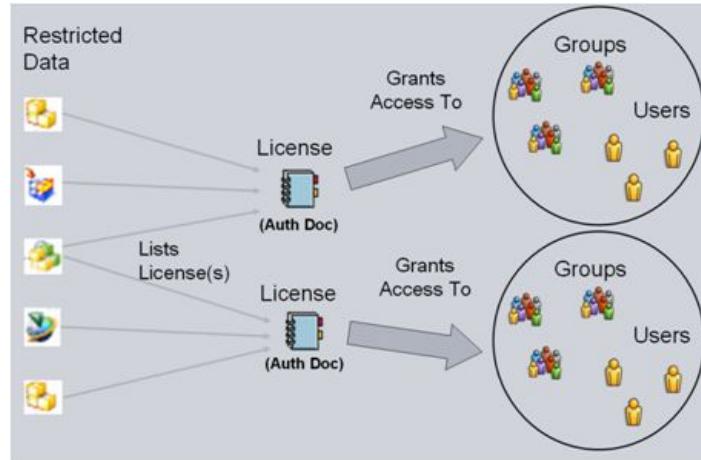
ACLs associated directly with the **In Job** condition are ignored when the rule tree is evaluated. Only ACLs associated with the workflow process are used to grant access.

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.

F.9 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), nondisclosure agreements, and contracts.

ADA assesses validation during any access attempt by restricted user.

F.10

Configuring authorized data access (ADA)

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.

F.10.1 More about configuring ADA

The ADA concepts described in the previous topics assume that data is stored in and accessed from within the Teamcenter environment. You can also configure logging and menu suppression (blocking) to be applied when classified data is loaded in Teamcenter Integration for NX. Logging provides an audit of actions performed on exporting data, and blocking suppresses NX menus to prevent geometric data from being exported outside of the NX/Teamcenter environment.

Note

In this context, *export* refers to performing an operation, such as creating a copy or printing data, that moves the data outside of the Teamcenter environment.

As a user with an ADA administrator role (**IP Admin** or **ITAR Admin**), you use the ADA License application to create and maintain licenses. Once created, access is either granted or denied to users and groups by associating the license directly with the data object.

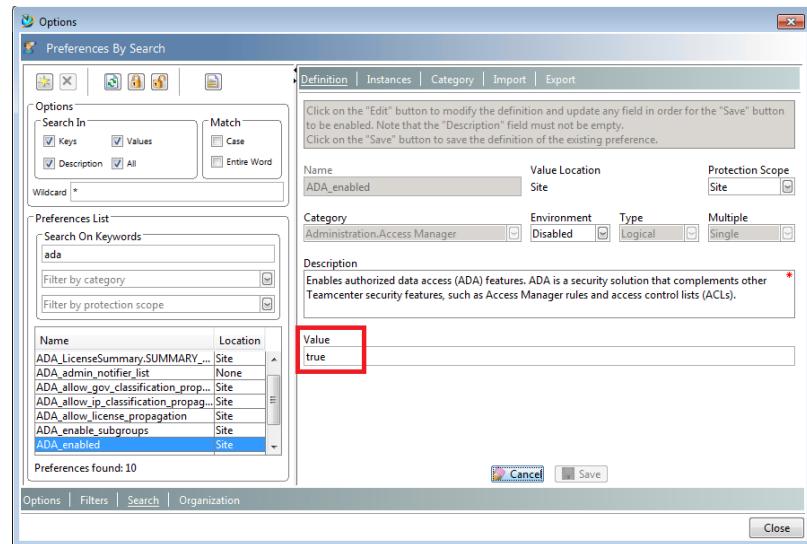
F.10.2 Basic tasks for configuring and administering ADA

The following basic tasks must be performed when configuring and administering authorized data access. Many of the tasks are the same. The differences are noted below and on pages that follow.

| ITAR restricted data | Intellectual property (IP) data |
|--|---|
| Enable authorized data access. | <i>Same task.</i> |
| Configure logging and blocking in NX. | <i>Same task.</i> |
| Define ITAR clearance and classification levels. | <i>Same task</i> , but using different preferences. |
| Assign users to the ITAR Admin role and grant ITAR Admin privileges. | <i>Same task</i> , but using the IP Admin role and IP Admin privileges. |
| Assign clearance levels, geographic location, nationality, and technology transfer certification dates to users. Assign geographic locations to sites. Assign nationality to groups. | Assign clearance levels to users. |
| Create ITAR licenses. | <i>Same task</i> , but IP licenses (different menu selections). |
| Assign government classification values to data objects. | Assign classification values to data objects. |
| Associate licenses with data objects. | <i>Same task.</i> |

F.10.3 Enable authorized data access

Authorized data access (ADA) features are enabled by default when you install Teamcenter. To enable ADA, set the **ADA_enabled** preference to **true**.

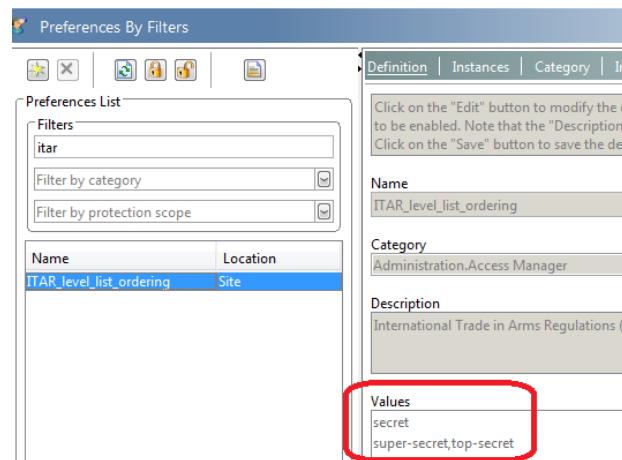


F.10.4 Define ITAR clearance and classification levels

Each site can define a list of ITAR/IP classification values and clearance levels that are assigned to data objects and users for ITAR/IP access evaluation.

The **ITAR** list is maintained in the **ITAR_level_list_ordering** preference.

The **IP** list is maintained in the **IP_level_list_ordering** preference. These are site preferences that must be set by a Teamcenter administrator.



Default Values

secret

super-secret,top-secret

Valid Values

Accepts multiple strings as value. Each string must provide valid ITAR/IP classifications for an object and the corresponding clearance levels on a user. Access Manager compares these values to determine user access rights to the object. String order determines classification level: the first entry is the lowest classification, the last entry is the highest.

Each entry *may* be a comma-separated list of equal-precedence classification levels. Thus, commas cannot occur in the classification values.

Scope

Site preference.

F.10.5 For ITAR/IP Admin users, assign the role and grant privileges

The **ADA_license_administration_privilege** preference specifies the privilege required to perform actions in the ADA License application, attach or detach license from workspace objects in My Teamcenter, or view audit logs of license activity from the ADA License application. The **ADA_license_administration_privilege** preference may be defined for a user either explicitly or in the context of a group or project.

Privileged users (or the owning user) are authorized to:

- Perform license modification and deletion.
- Classify data objects.
- Add and remove users from licenses.
- Set license expiration dates.

Default Value

ITAR_ADMIN

Valid Values

- **ITAR_ADMIN**

Allows users with the **ITAR Admin** privilege access to the ADA License application.

- **IP_ADMIN**

Allows user with the **IP Admin** privilege access to the Authorized Data Access application.

- **Administer_ADA_Licenses**

Allows users with the **Administer ADA Licenses privilege** access to the ADA License application.

Scope

Site preference.

Additional notes

Applying licenses to or removing licenses from objects requires that the user have **ITAR Admin** privilege on both the license and the object to which the license is being applied.

Users can be assigned to the **ITAR Admin** role or the **IP Admin** role by a Teamcenter administrator using the Organization application.

License modification is not controlled by Access Manager rules.

F.10.6 Assign geographic locations to sites and nationality to groups

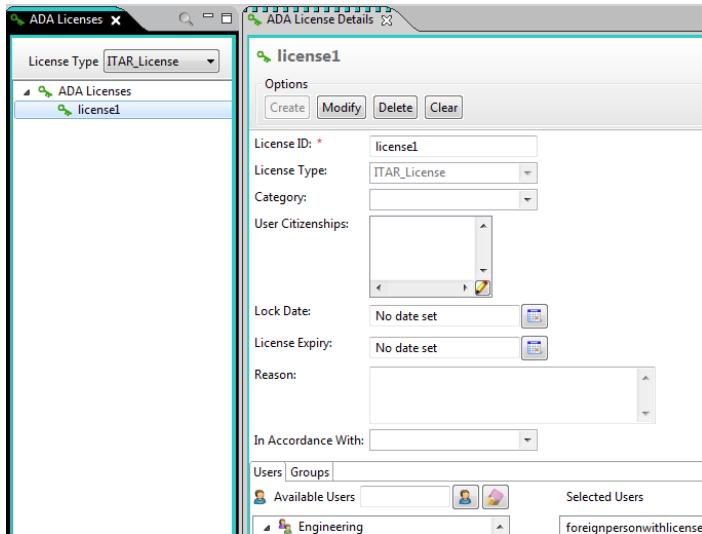
Assigning a geographic location to a site enables you to write access rules to control Multi-Site Collaboration privileges for exporting and transferring data. The **geography** attribute is set in the Organization application.

You can set the **nationality** attribute on Teamcenter groups, which enables you to use the group as the basis for selectively granting users access to restricted data based on ITAR licensing. Groups are created and maintained using the Organization application.

Note

Teamcenter groups are subject to hierarchical group behavior. Therefore, subgroups inherit the properties of the parent group. Siemens PLM Software recommends that users be made members of the most specific group possible to ensure the accuracy of key attributes, such as the nationality of the organization.

F.10.7 Create an authorized data access license



1. Select the **ADALicenses** node from the **Organization List** tree.
2. Type the license ID.
3. Select the license type from the **License Type** list:
 - **IP_License**
 - **ITAR_License**
 - **Exclude_License**
4. (Optional) Select a category for the license from the **Category** list.
5. Set the license expiration date by selecting a date and time using the **License Expiry** box. Click **Calendar** to display the calendar.
6. (Optional) Type a reason in the **Reason** box.
7. Select the users and groups that are allowed to access objects with the license attached.
 - (Optional) To allow access to specific users, click the **Users** tab, select the user name in the **Available Users** list, and click (>) to add the user to the **Selected Users** list.
 - (Optional) To allow access to specific groups and the associated subgroups, click the **Groups** tab, select the group name in the **Available Groups** list, and click (>) to add the group to the **Selected Groups** list.
8. Click **Create**.

F.10.8 Assigning government classification values to data objects

Access to classified data is determined by an evaluation of the user's clearance level and the classification level that is applied to the object.

| ITAR restricted data | Intellectual property (IP) data |
|---|--|
| The Gov Classification property specifies the classification level of an individual object and can be viewed and modified in the same manner as other object properties. | The IP Classification property specifies the classification level of an individual object and can be viewed and modified in the same manner as other object properties. Valid values for this property are derived from the IP_level_list_ordering preference. |

Note

The **Classification**, **Classified**, and **Classified in** properties apply to the Classification application that is used to categorize objects for reuse. They do not apply to classification for the purpose of authorizing data access.

F.10.9 Associate licenses with data objects

ITAR/IP licenses grant named users discretionary, limited-time access to classified data and are created and maintained by users who assigned to the **ITAR Admin/IP Admin** role. Users also have **Write** and **ITAR Admin/IP Admin** privileges to both the object and the license. **ITAR/IP** administrators attach licenses to data objects.

1. In the My Teamcenter tree pane, right-click the object to which you want to attach the license.
2. Choose **License**→**Attach** from the shortcut menu.
3. In the **Assign an Object to Licenses** dialog box, select the license that you want to attach to the object.
4. Click **OK**.

Note

- You must have privileges specified in the **ADA_license_administration_privilege** site preferences on both the object and the license to attach or remove licenses.
- You may want to set the **ADA_enable_subgroups** site preference to specifies whether or not subgroup members of the top-level groups can be authorized access to workspace objects with attached licenses when the top-level group is not authorized access.

F.11 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

F.12 **Summary**

The following topics were taught in this lesson:

- Authentication, authorization and authorized data access.
- Controlling access on intellectual property (IP) data.
- Controlling data access subject to International Traffic in Arms Regulations (ITAR) policies.
- Controlling access to working and in-process data.

Appendix

G Structure Manager revision rules

Purpose

The purpose of this lesson is to create and apply revision rules to select the appropriate revision of components in Structure Manager.

Objectives

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

- Define revision rules in Structure Manager.
- Import revision rules with the Teamcenter **plmxml_import** utility.

Help topics

Additional information for this lesson can be found in:

- *Structure Manager Guide*

G.1 Revision rules review

A revision rule is a set of parameters that determine which revision of an item is loaded into a particular Structure Manager window. With the **CCC Latest Working** as the revision rule, the product structure is configured with the **3540/B Bumper** component.

| 3500/A;1-Chassis ASM (View) - Latest Working - | |
|--|-----------|
| BOM Line | Item Type |
| 3500/A;1-Chassis ASM (/view) | Item |
| 3540/B;1-Bumper | Item |
| 3510/A;1-Pan | Item |
| 3530/A;1-P-Pack | Item |

Item revision configuration selects one of the revisions of an item is to be configured in a Structure Manager window (that is, meets the criteria in the revision rule).

A preset revision rule contains various parameters that have been defined by a privileged user and made available to other users.

The following graphic shows the revision rule setting is **CCC Latest Working**.

| |
|---|
| 4500/A;1-Model Car (View) - Latest Working - Date - "Now" |
|---|

This table shows the **CCC Latest Working** revision rule definition.

| Revision rule | Definition |
|--------------------|---|
| CCC Latest Working | Working () Has Status (Any Release Status, Configured Using Released Date) |

G.1.1 Revision configuration terms

The revision rule must always be set to determine which revision to load into Structure Manager.

Imprecise Assembly

A single-level assembly that has items as the components. The particular revision is determined by the revision rule settings.

Precise Assembly

A single-level assembly that has specific item revisions as the components. It is also subject to the revision rule, but the precisely specified item revision is configured by a precise entry in a revision rule.

Revision Rule

The parameters set by the user that determine which revision of an item is loaded into a particular Structure Manager window. A revision rule can also be saved as a Workspace object for use in other applications.

Rule Entry

A revision rule is made up of an ordered list of rule entries. Each type of rule entry is concerned with a particular type of configuration.

Override List

The user places item revisions in a workspace folder that is referenced from the revision rule and used to override the revision that would normally be loaded by the revision rule.

Release Status

An attribute assigned to an object after successfully going through a release. Item revisions can be configured according to their status. The status can optionally contain effectivity data for revision configuration.

Working Revision

A revision that can be freely changed by a user with write privileges. No record of intermediate states of a working revision is maintained by Teamcenter.

G.1.2 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 may contain any number of rule entries.

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.

You define each of the these criteria with a revision rule entry.

- 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.
- 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.

G.1.3 Revision Rule Definitions

Teamcenter provides multiple revision rules as shown.

Any Status, No Working

Selects the latest released revisions, no working revisions selected.

Any Status; Working

Selects the latest released revisions, if any exist, and then selects working revisions. This is useful when you want to configure a released structure, but want to be aware of items that will be used in the structure, but are not yet released.

Latest Working

Selects precise references if they exist, and for imprecise assemblies, selects the latest working revisions, if any exist, and then latest released revisions.

Latest by Alpha Rev Order

Selects the latest revisions according to the revision ID, sorted alphanumerically, regardless of whether they are working or released.

Latest by Creation Date

Selects the latest revisions according to the date they were created, regardless of whether they are working or released.

Precise Only

Selects the precise references to specific item revisions in precise assemblies.

Precise; Any Status

Selects the precise references to specific item revisions in precise assemblies. If imprecise assemblies are present, the latest revisions with **any status** are selected.

Precise; Working

Selects the precise references to specific item revisions in precise assemblies. If imprecise assemblies are present, only working revisions are selected.

Working; Any Status

Selects the latest working revisions, if any exist, and then selects the latest released revisions.

Working (Curr User); Any Status / Working (Curr Group); Any Status

Selects only the latest working revisions owned by the user (or the group) running the Teamcenter session is currently logged onto, if any exist, and then selects the latest released revisions. This is a simple and powerful rule allowing users to configure only the working data within their team.

During this lesson, you will create additional revision rules as shown.

CCC Released

Selects the latest revisions with a release status of **Released**, if any exist, and then selects the latest revisions with a release status of **Pre-Released**, if any exist, and then selects the latest working revision.

The following revision rule definitions are imported during the training class:

CCC Latest ECPending

Selects the latest revisions with a release status of **ECPending**, if any exist, and then selects the latest revisions with a release status of **Pre-Released**, if any exist, and then selects the latest working revision.

CCC Latest Released

Selects the latest revisions with a release status of **Released**, if any exist, and then selects the latest working revision.

CCC Latest Working

Selects the latest working revisions, if any exist, and then selects the latest released revisions.

CCC Released; Config by Date

Selects the latest revisions with a release status of **Released** according to the date they were created.

G.1.4 Configuring privileged and unprivileged users

User privileges determine if a user can create and modify revision rules or only apply rules created by others.

- **Privileged** users have access to all the revision rule menu commands.
- **Unprivileged** users have access only to a subset of the commands.
Unprivileged users cannot create or delete revision rules, and have only limited ability to modify revision rules without saving the changes.

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

Furthermore, you can restrict user access to revision rule menu commands with the Command Suppression application.

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**

Users may save new or changed revision rules.

Unprivileged users have access to the following commands:

- **View/Set Revision Rules**
- **Set Date, Set Unit, Set Override**, where appropriate

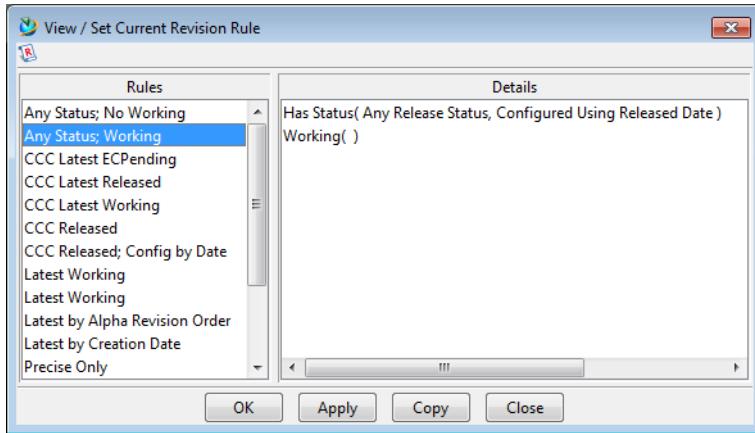
Note

Prior to Teamcenter 8, the following preference was recommended to determine which users can create, delete, and modify preset configuration rules.

```
TC_config_rule_users=
user1
user2
```

G.1.5 View or set a revision rule

The **Details** column displays the combined rule configuration information.



The values that may appear are:

- **Working (Owning User = *user*, Owning Group = *group*)**

This appears if the item revision is working and therefore configured by a **Working** entry in the revision rule. The owning user and owning group arguments appear only 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:

- o **Date (Today) or Date (1-Jan-2007)**

The date specified in the revision rule. Used to configure historical revisions with status.

- o **Unit No.(4)**

The unit number specified in the revision rule.

- o **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 appears if no revision meets the revision rule criteria. Teamcenter displays **???** to represent the revision.

Note

Use the **Rule Configured By** column in Structure Manager to interpret how a particular line is configured.

G.2 Revision rules setup

A revision rule is comprised of a sequential list of entries. Evaluation of the rule involves evaluating each of the entries, in top-to-bottom order, until a configured revision of the item is successfully obtained. Once a revision is found that satisfies an entry, the evaluation of entries below that entry is stopped.

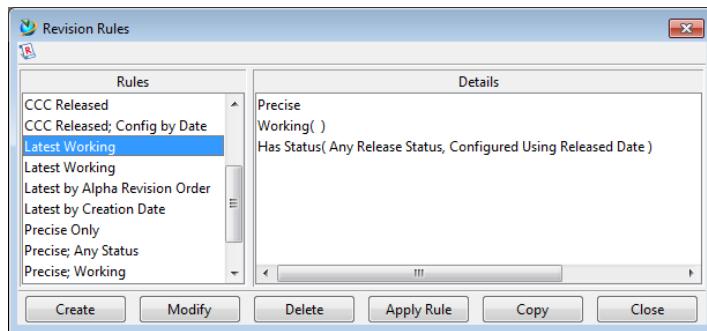
The screenshot shows a configuration dialog box titled 'Entries'. It contains the following fields:

- Name: Any Status; Working
- Description: (empty)
- Nested Effectivity
- Has Status(Any Release Status, Configured Using Released Date)
Working()

G.2.1 Create, modify, or delete revision rules

Revision rules are modified or created from the Structure Manager application.

Choose **Tools**→**Revision Rule**→**Create/Edit...** to create, modify, delete, and copy preset revision rules.

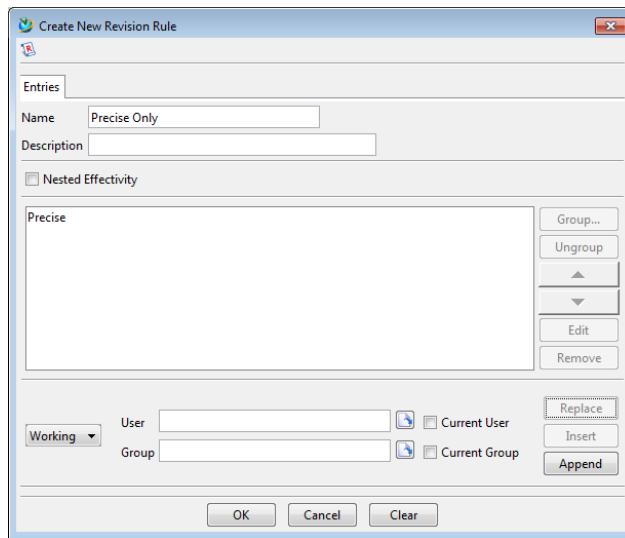


Revision Rules dialog box

- Click **Create** to display the **Create New Revision Rule** dialog box. Enter a name, set the criteria required, and then click **OK** to see this new revision rule added to the list in the initial dialog. Note that revision rule names must be unique.
- Click **Modify**, which displays the same dialog as **Create**, but prepopulates it with the revision rule name and the settings that correspond to the rule. These can then be modified as required. This also allows the name of the rule to be changed.
- Click **Delete** to remove a Revision Rule from the database.

G.2.2 Create new revision rule

The **Create New Revision Rule** dialog box allows users to create revision rules. Default Access Manager Rules shipped with Teamcenter define who can read and therefore use a revision rule after it is created.



Create New Revision Rule dialog box

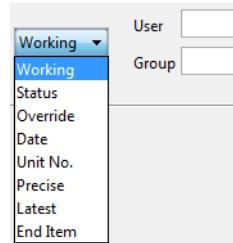
- The upper section allows a name to be entered, which is the name the user sees.
- The middle section is the area where entries are ordered and possibly grouped.
- The lower section is where rule entries are constructed and modified.

Note

- If the revision rule is created by a member of the **dba** group, the revision rule is available to all Teamcenter users (**World**).
- If the revision rule is created by any other user, only that user and system administrators can view/select the rule.

G.2.3 Working entry

A **Working** entry is used to select working item revisions (those without any release status).



The latest working revision of the item is selected according to the date it was created. The revision chosen can be made more specific using the following settings:

Note

More than one **Working** entry may be present within a revision rule. For example, a rule may try to configure the current user's working revision, and if none is found, then configure the current group's instead. If users changes their group, they must re-apply the revision rule to configure the appropriate revisions for the group they have changed to.

Owning User and Group

- **Owning User**

If an owning user is specified within a **Working** entry, the latest revision owned by the specified user is configured. The owning user can be also set to current, meaning that latest revision owned by the current Teamcenter user is configured.

- **Owning Group**

If an owning group is specified within a **Working** entry, the latest revision owned by the specified group is configured. The owning group can be also set to current, meaning that latest revision owned by the user's current Teamcenter group is configured.

G.2.4 Status entries

A **Status** entry is used to select item revisions that have been released with a particular status. The following settings are available for **Status** entries:

- **Any Release Status**

The latest item revision with status is configured, regardless of which status it is.

- **Selected Status**

The latest item revision with status of the selected type is configured. It allows you to configure a structure that only contains item revisions that have reached a specified status. These are selected from the list of available statuses in the list menu.

Note

More than one **Status** entry may be present in a revision rule.

Status with date or unit

- **Release Date**

The latest item revision is selected according to the date the revision was released (the date that the particular status was added).

- **Effective Date**

The latest item revision is selected according to effectiveness dates defined on the release status.

- **Effective Unit No.**

The latest item revision is selected according to unit numbers defined on the release status.

Note

Effective dates/units are defined by privileged users using the **Tools→Effectivity→Revision Effectivity...** command in the Structure Manager application.

G.2.5 Override entry

An **Override** entry allows particular item revisions to override those that would be selected by the other criteria. You add an **Override** entry if you want to create a rule referencing an override folder.

Typically you will leave the folder argument blank, which allows users to set an override folder at run time, using the **Tools→Revision Rule→Set Override Folder...** command.

In some cases, you may designate the override folder for special revision rules (for example, a group override folder).

As a Structure Manager user, you perform these actions:

1. Create a folder for your override item revisions.
2. Copy the item revisions to be used into your folder.
3. Set an override folder using the **Tools→Revision Rule→Set Override Folder...** command.

The item revisions in your designated folder will override the revision of the same item in the Structure Manager window.

G.2.6 Date and Unit entries

A **Date** entry is used when you want to create a revision rule that configures at a specific date or today. This prevents users from setting a date (using **Revision Rule→Set Date**) to configure by at runtime. If you want a revision rule that allows users to set the date, do not use a date entry.

A **Unit Number** entry is used to specify the unit number to match against when configuring item revision with status using **Unit No Effectivity**. This prevents users from setting a unit number (using **Revision Rule→Set Unit No**) to configure by at runtime. If you want a revision rule that allows users to set the unit number, do not use a **Unit Number** entry.

G.2.7 Precise and Latest entries

A **Latest** entry is used to select item revisions irrespective of whether they have been released. There is no differentiation between working revisions and those with status.

Latest entry settings

- **Creation Date**

The latest item revision is selected according to the date the revision was created.

- **Alphanumeric Revision ID**

The latest item revision is selected in alphabetic order of revision ID. This orders numeric Revision IDs alphanumerically, (for example, 1, 10, 2, and so on).

- **Numeric Revision ID**

The latest item revision is selected in numeric order of revision ID. Revisions with nonnumeric IDs are not configured.

A **Precise** entry is used to select the precisely specified item revisions in a precise BOM view revision. The entry has no effect on imprecise bills. If you apply a revision rule without a **Precise** entry to precise structures, they are dynamically configured. This can be very powerful, but can also give unexpected results that may not be desired.

Note

Any entry placed above the **Precise** entry in a revision rule overrides the precise references.

A revision rule containing only a **Precise** entry can be used to do purely precise configuration.

Note

If you want to configure a structure that contains both imprecise and precise assemblies, and you want to configure the precise references in the precise assemblies, add a **Precise** entry at the top of your revision rule. The other entries in the rule are used to configure the imprecise assemblies.

G.2.8 Ordering and grouping rule entries

Ordering rule entries

The order of entries within a revision rule is very important. The top-to-bottom order defines a precedence that affects the outcome of what item revision the rule configures. Entries at the top of the rule take precedence over entries below. As soon as an entry is capable of configuring an item revision, the entries below are not evaluated.

Grouping rule entries

To give entries equal precedence, they may be grouped together with brackets in the entry dialog box.

For example, consider the following rule entries:

```
Precise  
[ Has Status( Released, Configured Using Released Date ) OR  
Has Status( Pre-Released, Configured Using Released Date ) ]  
Working( )
```

Grouping these entries gives no precedence to item revisions with either of the defined statuses. The latest item revision with either of the statuses is configured.

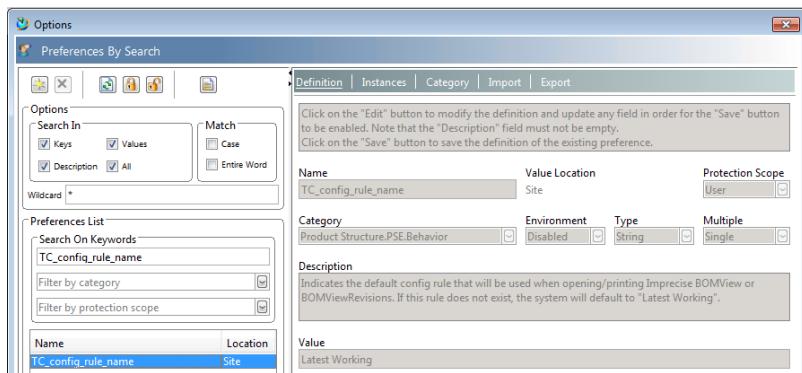
G.3 Setup considerations

Teamcenter is delivered with a base setup that allows the product to be used with the revision rule, **Latest Working**.

To use other revision rules, you may consider modifying the following preferences using the **Edit→Options→Search** menu command.

The following preference establishes the revision rule a user has in effect by default. The default can be changed to a different revision rule for the whole site, groups, roles, or users.

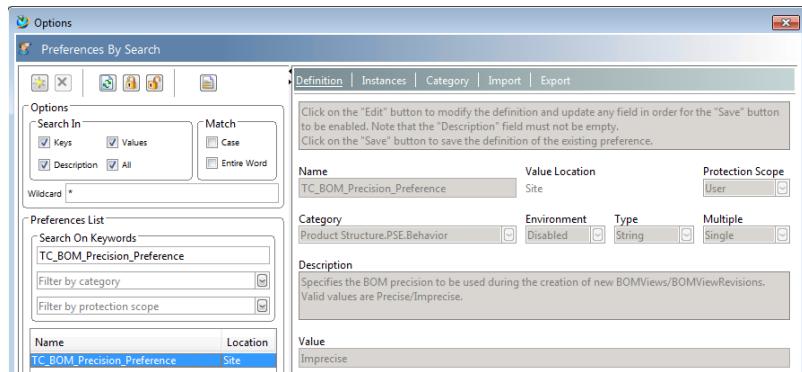
```
TC_config_rule_name=
Latest Working
```



G.3.1 BOM precision preference

The following preference defines the type of BOM (precise/imprecise) users create by default from Structure Manager.

TC_BOM_Precision_Preference=
Imprecise



G.4 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

G.5 **Summary**

The following topics were taught in this lesson:

- Define revision rules in Structure Manager.
- Import revision rules with the Teamcenter **plmxml_import** utility.

Appendix

H Administering workflow processes

H.1 Administering workflow processes

Purpose

The purpose of this lesson is to learn key elements of administering Teamcenter workflow processes.

Objectives

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

- Create a quick-release process.
- Release data using the **release_man** utility.
- Troubleshoot a workflow process.
- Understand workflow utilities.

Help topics

Additional information for this lesson can be found in:

- *Workflow Designer Guide*

H.1.1 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.

H.1.1.1 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 **EPM-create-status** and **EPM-set-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 **EPM-create-status** and **EPM-set-status** handlers.
- Create a process based on the **Empty Process** template and add the **EPM-create-status** and **EPM-set-status** handlers.

H.1.1.2 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 **EPM-create-status** and **EPM-set-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.

H.1.1.3 Batch release utility

Use the **release_man** utility to release objects in batch mode without creating workflow processes or 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.

H.1.2 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.

H.1.2.1 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. |

H.1.2.2 Cancel a workflow process

To cancel a workflow process and exit without process completion, use the **Actions→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.

H.1.2.3 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.

H.1.2.4 Delete a workflow process

To delete the entire workflow process after it has been initiated, use one of these procedures:

- In **My Worklist**, select a workflow task for which you are the responsible party, and then click **Delete**.
- In the **Impact Analysis** view, select the workflow task, and then click **Delete**.

Note

Deleting a task in the workflow deletes the entire workflow process.

H.1.3 Key workflow utilities

The following are important utilities to be familiar with when working with workflow processes:

- **tc_workflow_postprocess**

The **tc_workflow_postprocess** utility executes a specific action on a specific task in the workflow process from which the related background process was initiated.

For example, you can use this utility to evoke the complete action on the **Review** task in the workflow process from which a background tessellation process was initiated. This utility then prompts the workflow task to either execute an action (defined with the **-action** argument) or submit a decision (defined with the **-signoff** argument). This can be useful when the conditions to execute the action are not met until the background process completes. In these cases, the user has typically moved on to other tasks or ended the session.

- **purge_processes**

The **purge_processes** utility purges completed processes based on the last-modified date of the process. Objects such as e-mail messages that reference the process are not deleted. Use the **-f** argument to delete both in-process and completed processes and sever the references between objects and the processes.

- **migrate_wf_handlers**

The **migrate_wf_handlers** utility transforms the name and/or arguments of workflow handlers from one format to another. Task templates may have one or more workflow handlers associated with them. This utility transforms the handlers associated with templates that are active, as well as template versions that are obsolete but that are still referenced by uncompleted workflow processes or jobs.

H.1.4 Activities

Proceed to the activities for hands-on practice to reinforce the topics covered in this lesson.

If necessary, review the *How to use the activities* section at the top of the list of activities for this course. You should be using this activity set.

Teamcenter Application and Data Model Administration

Student Activities

MT25460 - Teamcenter 10.1

H.1.5 Summary

The following topics were taught in this lesson:

- Create a quick-release process.
- Release data using the **release_man** utility.
- Troubleshoot a workflow process.
- Understand workflow utilities.

Index

A

Access control entry (ACE) [12-8](#)
Access control list (ACL) [12-8](#)
Access control lists [12-9](#)
 Bypass [12-36](#)
 Item Revs [12-37](#)
 Items [12-36](#)
 Vault [12-36](#)
 Working [12-37](#)
Access Manager
 Adding rule [12-40](#)
 Basic concepts [12-4](#)
 Basic tasks [12-5](#)
 Cautions for using [12-47](#)
 Controlling revision rules [12-53](#)
 Export [12-43](#)-[12-44](#)
 Import [12-43](#)-[12-44](#)
 Interface [12-3](#)
 Modifying rule [12-42](#)
 Purpose [12-2](#)
Accessors
 Approver [12-23](#), [12-55](#), [19-3](#)
 Approver (Group) [12-55](#), [19-3](#)
 Approver (RIG) [12-55](#), [19-3](#)
 Approver Group [12-23](#)
 Approver RIG [12-23](#)
 Approver Role [12-23](#)
 Current project team [12-24](#)
 Current project teams [12-24](#)
 Group [12-22](#)
 Group administrator [12-22](#)
 Groups with security [12-22](#), [12-62](#)
 Groups with Security [12-62](#)
 Owner (owning user) [12-22](#)
 Owning group [12-22](#), [12-62](#)
 Owning Group [12-62](#)
 Project team [12-23](#)
 Project teams [12-23](#)

Public schedule [12-24](#)
Remote site [12-22](#)
Responsible party [12-23](#)
Responsible Party [12-55](#), [19-3](#)
Role [12-22](#)
Role in group [12-22](#)
Role in owning group [12-22](#)
Role in project [12-24](#)
Role in projects of object [12-24](#)
Site [12-22](#)
System administrator [12-22](#)
Task owner [12-23](#)
Task Owner [12-55](#), [19-3](#)
Task owning group [12-23](#)
Task Owning Group [12-55](#), [19-3](#)
User [12-22](#)
User Excluded [12-24](#)
User Has Government
 Clearance [12-24](#)
User Has IP Clearance [12-25](#)
User IP Licensed [12-25](#)
User IP Unlicensed [12-25](#)
User ITAR Licensed [12-25](#)
User ITAR Unlicensed [12-25](#)
User Over Government
 Clearance [12-25](#)
User Over IP Clearance [12-25](#)
User Under Government
 Clearance [12-25](#)
User Under IP Clearance [12-25](#)
World [12-22](#)
Account
 Generation [11-55](#)
Accounts
 infodba [11-18](#)
 Organization structure [11-21](#)
 System administration [11-17](#), [11-19](#)
ACE (access control entry) [12-8](#)

| | |
|-------------------------------------|----------------------|
| ACL | |
| Adding entries | 12-41 |
| Creating | 12-41 |
| ACL (access control list) | 12-8 |
| ACL, working | 12-51 |
| Action handlers | 19-5, 19-7 |
| Activating | |
| Group members | 11-53 |
| User accounts | 11-47 |
| Active extension files | 1-38, 1-40 |
| Adding | |
| Condition tasks | 20-9, 20-11 |
| Dataset business objects | 7-5 |
| Dataset named references | 7-7 |
| Existing users to a group/role | 11-42 |
| Item business objects | 2-18 |
| Lists of values (LOVs) | 5-9 |
| Naming rules | 9-5–9-6 |
| New roles to a group | 11-37 |
| New users to a group/role | 11-39 |
| Notes | 8-4 |
| Persistent properties | 2-23, 3-9, 4-8 |
| Projects | 1-25 |
| Relation properties | 6-5, 6-9 |
| Status objects | 8-6 |
| Tool objects | 7-10 |
| Units of measure | 8-8 |
| Adding rules | 12-40 |
| Administer ADA privilege | 12-21 |
| Administrative reports | |
| Admin-Employee Information | 16-7 |
| Admin-Group/Role | |
| Membership | 16-7 |
| Object Ownership | 16-7 |
| Objects by Status | 16-7 |
| AM Rules tab | 13-5 |
| Application banner | 1-11 |
| Approver | 12-23, 18-3 |
| Group | 12-23 |
| RIG | 12-23 |
| Role | 12-23 |
| Approver (Group) accessor | 12-55, 19-3 |
| Approver (RIG) accessor | 12-55, 19-3 |
| Approver accessor | 12-55, 19-3 |
| Array column | 2-25 |
| Array Length column | 2-25 |
| Assign to project privilege | 12-20 |
| Assigning default process templates | 19-10 |
| Attach | |
| Naming rules | 9-5 |
| Attaching | |
| LOVs | 2-32, 5-10 |
| Naming rules | 9-7 |
| Attachment | 18-3 |
| Reference object | 18-3 |
| Target object | 18-3 |
| Attribute Selection pane | 15-4 |
| Attributes | 19-2 |
| Data Model | 17-4 |
| Settings | 2-25 |
| Storage Type | 2-24 |
| Authentication | 11-11, F-6 |
| Authorization | F-7 |
| Authorized data access | F-10 |
| Authorized data access licenses | |
| Creating | F-18 |
| autoAssignToProject extension | C-9–C-10 |
| B | |
| Back and forward buttons | 1-11 |
| Backup | 1-43–1-46, 1-48–1-49 |
| Base-action | C-5 |
| Basic Access Manager concepts | 12-4 |
| Basic concepts | F-4 |
| Business Modeler IDE | 1-7 |
| Basic concepts about Report | |
| Builder | 16-4 |
| Basic tasks | 17-5 |
| Batch Print privilege | 12-20 |
| bmide_manage_batch_lovs utility | A-7 |
| BOWriterExport transfer mode | 17-11 |
| Browse mode | 18-7 |
| Business Modeler IDE | |
| Concepts | 1-7 |
| Configuring | 1-35 |
| Preferences | 1-35 |
| process | 2-14 |
| Purpose | 1-2 |

| | | | |
|--|----------------|--|-------------------|
| Template projects | 1-35 | Condition tasks, adding | 20-9 |
| User interface | 1-3-1-5 | Condition Usage report | 8-9, 8-12, 8-14 |
| Using | 1-9 | Conditions | 12-26 |
| Business object constants | 3-17 | Adding | 9-24 |
| Business object display rules | | Example | 9-22 |
| Adding | 9-14 | Examples | 9-27 |
| Reference | 9-13 | Group Nationality | 12-33 |
| Business objects | | Has Attribute | 12-26 |
| Creating | 3-7 | Has Bypass | 12-26 |
| Dataset | 7-5, 7-7 | Has Class | 12-26 |
| Form | 4-4, 4-6 | Has Description | 12-27 |
| Icon Overlay | 3-26 | Has Form Attribute | 12-28 |
| Icons | 3-24 | Has Government | |
| ImanRelation | 6-3 | Classification | 12-34 |
| In general | 1-9 | Has Item ID | 12-28 |
| Item | 2-18 | Has Name | 12-27 |
| Properties | 2-21-2-22 | Has No Government | |
| Relationships | C-11 | Classification | 12-34 |
| Bypass switch | 11-19 | Has No IP Classified | 12-35 |
| C | | Has Object ACL | 12-27 |
| Candidate Key column | 2-25 | Has Status | 12-27 |
| Cascading LOVs | 5-19 | Has Type | 12-26 |
| Cautionary statements for using | | In Current Program | 12-32 |
| rules | 12-47 | In Current Project | 12-31 |
| Change ownership privilege | 12-19 | In IC Context | 12-30 |
| Change privilege | 12-19 | In Inactive Program | 12-32 |
| Change privilege, guidelines | 12-52 | In Invisible Program | 12-32 |
| Changing user status | 11-44 | In Job | 12-27, 12-54, F-9 |
| CICO privilege | 12-20 | In Project | 12-31 |
| Class/Attribute Selection dialog | | Inactive Sequence | 12-29 |
| box | 15-4 | Introduction | 9-21 |
| Classes | | Is Archived | 12-28 |
| In general | 1-9 | Is GA | 12-30 |
| Reference | 2-7 | Is Local | 12-28 |
| Classification | 3-16 | Is Owned By Program | 12-32 |
| Clipboard button | | Is Program Member | 12-31 |
| Description | 1-12 | Is Project Member | 12-31 |
| Closure rules | 17-7 | Is SA | 12-30 |
| Compare Two Data Models | | Object Has IP Classification | 12-35 |
| report | 8-9-8-10, 8-13 | Operators | 9-26 |
| Compound properties | | Overview | 9-23 |
| Adding | B-4, B-8 | Owning Group | 12-29, 12-62 |
| Condition tasks | | Owning Group Has Security | 12-30, 12-61 |
| Adding | 20-11 | Owning Site | 12-29 |
| | | Owning User | 12-29 |

| | | | |
|--|-----------|--|--------------------|
| Signature | 9-25 | Project team | 12-24 |
| Site Location | 12-33 | Project teams | 12-24 |
| User Has Government | | | |
| Clearance | 12-34 | D | |
| User Has IP Clearance | 12-35 | Data | |
| User Is Excluded | 12-34 | In-process | 12-10 |
| User Is IP Licensed | 12-35 | Released | 12-10 |
| User Is ITAR Licensed | 12-33 | Working | 12-10 |
| User Location | 12-33 | Data model | |
| User Nationality | 12-32 | Deploying | 2-33–2-34 |
| Working with | 9-24 | Example | 2-17 |
| Configuring security for special project | | Inheritance | 2-12 |
| data | 12-66 | Merge preferences | 10-27 |
| Configuring supplier security | 12-65 | Packaging | 2-36 |
| Configuring supplier security for external | | Schema | 2-10 |
| data | 12-64 | Synchronize | 10-26 |
| Configuring supplier security for internal | | Data Model Documentation report .. 8-9, | |
| data | 12-63 | 8-12 | |
| Configuring the Business Modeler | | Data Model report | 8-9, 8-11 |
| IDE | 1-35 | Data sharing utilities | |
| Constants | | import_file | 14-18, D-13 |
| Property | 3-12, 4-9 | Dataset | |
| Controlling volumes' access | 11-26 | Tool | 7-9 |
| COTS (commercial off-the-shelf) | 1-32 | Dataset business objects | |
| COTS scope rules | 17-15 | Creating | 7-5 |
| Counters in naming rules | 9-9 | Reference | 7-2 |
| Create failure path | 20-8 | Dataset named references | |
| Create Operation Override report | 8-9, 8-12 | Adding | 7-7 |
| create_project utility | 13-17 | Datasets | |
| Creating | | Configure | 7-8 |
| Authorized data access licenses .. | F-18 | Viewing and opening | 7-8 |
| Business objects | 3-7 | Deactivating group members | 11-52 |
| Dataset business objects | 7-5, 7-7 | Deep copy rules | 9-15 |
| Groups | 11-27 | Adding | 9-18–9-19 |
| Item business objects | 2-18 | definitions | 9-16 |
| Persons | 11-29 | hierarchy | 9-17 |
| Preferences | 14-12 | Default local volume | 11-33 |
| Projects | 1-25 | Defining search criteria | 15-9 |
| Queries | 15-16 | Definition pane | 13-5 |
| Roles | 11-28 | Definition tab | 13-5 |
| Users | 11-31 | Delete | |
| Volumes | 11-22 | Workflow process | H-13 |
| Creating ACL | 12-41 | Delete privilege | 12-19 |
| Creating projects | 13-17 | Guidelines | 12-52 |
| Current | | Demote privilege | 12-19, 12-56, 19-4 |

| | |
|---|--------------|
| Deploying | |
| Live updates | 2-33 |
| Overview | 2-33 |
| Templates | 2-34 |
| Development environment | 1-22 |
| Displaying inactive group members | 11-54 |
| Displaying regular properties | D-3 |
| DisplayName business object constant | 3-17 |
| Dynamic LOVs | 5-21–5-22 |
| E | |
| Eclipse | |
| Downloading from IBM | 1-2 |
| ECO reports | |
| ECO Details Report | 16-7 |
| ECO Signoff Details | 16-7 |
| Editors | |
| UML | 2-10 |
| Effective ACL example | 12-17 |
| Enabled property constant | 3-13 |
| EPM-set-rule-based-protection handler | 12-54 |
| EPM-set-rule-based-protection handler | F-9 |
| Exclude licenses | F-11 |
| Export privilege | 12-19 |
| Exporting | |
| Preferences | 14-25 |
| Exporting JT files | 17-11 |
| Extension files | |
| Adding | 1-39 |
| Examples | 1-41–1-42 |
| Moving items into | 1-39 |
| Setting | 1-38, 1-40 |
| Extensions | |
| Working with | C-3 |
| External groups | 12-64 |
| Externally managed LOVs | A-7 |
| F | |
| Failure path | 20-7 |
| Favorites | 1-6 |
| Filter | 1-6 |
| Filter LOVs | 5-18 |
| Filtering | 11-3 |
| Find | 11-3 |
| Group | 11-48 |
| Inactive group members | 11-48 |
| Reload | 11-48 |
| Role | 11-48 |
| User | 11-48 |
| Finding objects | 2-6 |
| Form | |
| Example | 4-2 |
| Form business objects | 4-4 |
| Add properties | 4-7 |
| Creating | 4-6 |
| Hide properties | 4-10 |
| Storage class form | 4-5 |
| Form creation | 4-3 |
| G | |
| Generic process templates | |
| Create | 18-16 |
| Getting started | 17-2, F-2 |
| Getting started with Report Builder | 16-2 |
| global_transfer utility | 11-52 |
| GRM rules | |
| Adding | C-11 |
| Group | 12-22 |
| Administrator | 12-22 |
| Group Nationality condition | 12-33 |
| Group-level security | 12-61 |
| Configuring for suppliers | 12-63–12-65 |
| External groups | 12-64 |
| Hierarchical groups | 12-65–12-66 |
| Internal groups | 12-63 |
| Groups | 11-13, 12-61 |
| Activating members | 11-53 |
| Creating | 11-27 |
| Deactivating members | 11-52 |
| Defining roles | 11-12 |
| Displaying inactive members | 11-54 |
| Group member | 11-13 |
| Hierarchy | 11-15 |
| Managing | 11-49 |
| Removing members | 11-51 |

| | |
|--|--------------|
| Rule conditions | 12-61 |
| Subgroups | 11-14 |
| Suppressing display of inactive members | 11-54 |
| Groups with security | 12-22, 12-62 |
| Groups with Security accessor | 12-62 |
| H | |
| Handlers | 19-2 |
| Action | 19-5 |
| Rule | 19-5 |
| Handlers arguments | 19-5 |
| Has Attribute condition | 12-26 |
| Has Bypass condition | 12-26 |
| Has Class condition | 12-26 |
| Has Description condition | 12-27 |
| Has Form Attribute condition | 12-28 |
| Has Government Classification condition | 12-34 |
| Has Item ID condition | 12-28 |
| Has Name condition | 12-27 |
| Has No Government Classification condition | 12-34 |
| Has No IP Classified condition | 12-35 |
| Has Object ACL condition | 12-27 |
| Has Status condition | 12-27 |
| Has Type condition | 12-26 |
| Hide properties | 4-10 |
| Hierarchical group | |
| security | 12-65–12-66 |
| Hierarchical LOVs | 5-19 |
| Hierarchical preferences | 14-8 |
| Hierarchy | |
| Group | 11-15 |
| I | |
| IBM Eclipse | 1-2 |
| Icons | 3-24, 3-26 |
| IDs | 3-15 |
| IDs for items | 3-15 |
| Import files | 7-4 |
| Import privilege | 12-19 |
| import_export_reports utility | |
| Using | 16-8 |
| import_file utility | 14-18, D-13 |

| | |
|---|-------------------|
| Importing | |
| Backed up projects | 1-48 |
| Localization files | 3-21 |
| Preferences | 14-24 |
| Projects | 1-31 |
| Template files | 3-20 |
| Importing JT files | 17-11 |
| In Current Program condition | 12-32 |
| In Current Project condition | 12-31 |
| In IC Context condition | 12-30 |
| In Inactive Program condition | 12-32 |
| In Invisible Program condition | 12-32 |
| In Job condition | 12-27, 12-54, F-9 |
| In Project condition | 12-31 |
| In-process data | 12-54, F-9 |
| Inactivating user accounts | 11-45 |
| Inactive Sequence condition | 12-29 |
| InAnySchedule | 12-24 |
| Incorporate Latest Live Update Changes wizard | 10-22 |
| infodba | 11-18 |
| Inherited column | 2-25 |
| Initial value | |
| Column | 2-25 |
| InSchedule | 12-24 |
| Installing | |
| Templates | 2-38 |
| Templates to a production server | 2-35 |
| Instances | |
| Preferences | 14-14–14-15 |
| Interdependent LOVs | A-2–A-3 |
| Interface | 18-8 |
| Interfaces | |
| Rich client | 1-11 |
| Internal groups | 12-63 |
| Introduction to the IDE | 1-2 |
| IP Admin privilege | 12-20 |
| IP licenses | F-11 |
| Is Archived condition | 12-28 |
| Is GA condition | 12-30 |
| Is Local condition | 12-28 |
| Is Owned By Program condition | 12-32 |
| Is Program Member condition | 12-31 |
| Is Project Member condition | 12-31 |
| Is SA condition | 12-30 |

| | |
|--------------------------------|-----------|
| ITAR Admin privilege | 12-20 |
| ITAR licenses | F-11 |
| Item business objects | |
| Creating | 2-18 |
| Data model | 2-16, 3-4 |
| Extending | 3-2–3-3 |
| IDs | 3-15 |
| Item IDs | 3-15 |

J

| | |
|---|-------|
| Job | 18-3 |
| JTDataExportDefault transfer mode | 17-11 |
| JTDataImportDefault transfer mode | 17-11 |

K

| | |
|----------------|------|
| Keys | 3-15 |
|----------------|------|

L

| | |
|--|---------------------|
| Language Translations dialog box | 15-4, 16-3 |
| Libraries tab | 13-5 |
| Licenses | |
| Exclude | F-11 |
| IP | F-11 |
| ITAR | F-11 |
| Licensing | 11-11, F-6 |
| Links | |
| Fail | 20-7–20-8 |
| List of values (LOV) | |
| Add, remove or clear | 5-11 |
| Lists of values (LOVs) | |
| Adding | 5-9 |
| Attaching | 2-32, 5-10 |
| Batch | A-4 |
| Cascading | 5-13, 5-15, 5-19 |
| Db loaded | 5-17 |
| Dynamic | 5-21–5-22 |
| Externally managed | A-7 |
| Filter | 5-13–5-14, 5-18 |
| Hierarchical | 5-19 |
| In general | 1-9 |
| Interdependent | 5-13, 5-16, A-2–A-3 |

| | |
|---|------------|
| interface | 5-3 |
| Introduction | 5-2 |
| Overview | 2-31 |
| Types | 5-4, 5-7 |
| Usage types | 5-6 |
| Value types | 5-5 |
| Literal variables in naming rules | 9-9 |
| Live updates | 2-33 |
| Elements | 10-2, 10-4 |
| In general | 10-7 |
| Incorporate Latest Live Update Changes wizard | 10-22 |
| Performing | 10-18 |
| Preference | 10-16 |
| Process | 10-14 |
| Synchronize | 10-26 |
| Localization | |
| Button | 3-22 |
| Importing files | 3-21 |
| Localization button | 15-4, 16-3 |
| LOV Value Management box | A-4 |
| Lower Bound column | 2-25 |

M

| | |
|---|-------|
| make_user | 11-55 |
| Examples | 11-56 |
| make_user utility | 11-32 |
| Managing groups | 11-49 |
| Master form permissions | F-5 |
| Menu bar | 1-11 |
| Rich client | 1-11 |
| Modifiable property constant | 3-13 |
| Modifying | |
| Volume properties | 11-25 |
| Modifying an existing project | 13-14 |
| Multifield keys | |
| Configuring | 3-15 |
| Considerations | 3-16 |

N

| | |
|----------------------------|---------|
| Named references | 7-3–7-4 |
| Naming objects | |
| Name collisions | 1-27 |
| Naming rules | |
| Adding | 9-6 |

| | |
|--|-------------|
| Attaching | 9-7 |
| Counters | 9-9 |
| Create | 9-4 |
| In general | 9-3 |
| Literal variables | 9-9 |
| Revision | 9-10–9-11 |
| System variables | 9-8 |
| Navigation pane | 1-11 |
| Rich client | 1-11 |
| New Model Element wizard | 2-15 |
| Non-privileged project team member | 13-9 |
| Note types | |
| Adding | 8-4 |
| Reference | 8-3 |
| Nulls Allowed column | 2-25 |
| O | |
| Object Has IP Classification condition | 12-35 |
| Object model hierarchy | F-5 |
| Object-based protection | 12-8 |
| object_string property | 3-17 |
| Operation Descriptor tab | 2-26 |
| Options | |
| In general | 1-9 |
| Setting | 14-11 |
| Status | 8-5 |
| Tool | 7-9 |
| Working with | 8-2 |
| Options dialog box panes | 14-10 |
| Organization | |
| Account generation | 11-55 |
| Filter | 11-3 |
| Find | 11-3, 11-48 |
| Group hierarchy | 11-15 |
| Groups | 11-13 |
| Passwords | 11-10 |
| Persons | 11-8 |
| Roles | 11-12 |
| Users | 11-9 |
| Volumes | 11-16 |
| Organization List tree | 11-3 |
| Organization tree | 11-3 |
| Outline view | 2-13 |

| | |
|--|--------------|
| Overview | |
| Conditions | 9-23 |
| Template installation to a production server | 2-35 |
| Owner (owning user) | 12-22 |
| Owning group | 12-22, 12-62 |
| Owning Group accessor | 12-62 |
| Owning Group condition | 12-29, 12-62 |
| Owning Group Has Security condition | 12-30, 12-61 |
| Owning Site condition | 12-29 |
| Owning User condition | 12-29 |

P

| | |
|--|----------------|
| package_live_updates utility | 10-29 |
| Packaging extensions for installation | 2-36 |
| Panes | |
| Attribute Selection | 15-4 |
| Saved Queries tree | 15-4 |
| Saved query properties | 15-4 |
| Search Criteria | 15-4–15-5 |
| Parallel task and parallel process ACL conflict resolution | 12-58 |
| Passwords | |
| Examples | 11-10 |
| Restrictions | 11-10 |
| Performance | 17-12–17-14 |
| Permissions for master forms | F-5 |
| Persistent object manager (POM) | 2-7 |
| Persistent properties, adding | 2-23, 3-9, 4-8 |
| Persons | 11-8 |
| Creating | 11-29 |
| Platform of the IDE | 1-2 |
| PLM XML/TC XML Export Import Administration interface | 17-3 |
| PLM XML/TC XML Export Import Administration | 2-9 |
| PLMXML_export_packed_bom_<transfer-mode-name> preference | 17-14 |
| PLMXMLAdminDataExport transfer mode | 17-11 |
| POM schema | 2-7 |
| In rich client applications | 2-9 |

| | |
|---|--------------------|
| Introduction | 2-7 |
| Viewing | 2-8, 2-11 |
| Portfolio, Program and Project | |
| Management utilities | |
| create_project | 13-17 |
| update_project_data | 13-19 |
| Post-action | C-5 |
| Pre-action | C-5 |
| Pre-condition | C-5 |
| Preference | |
| Category | 14-5 |
| Description | 14-6 |
| Environment | 14-5 |
| Import and export | 14-23 |
| Location | 14-4 |
| Multiple | 14-6 |
| Name | 14-4 |
| Options dialog box | 14-9 |
| Protection Scope | 14-4 |
| Type | 14-6 |
| Value | 14-6 |
| Preferences | |
| Business Modeler IDE | 1-35 |
| <business-object>_default | |
| relation | 6-2 |
| Creating | 14-12 |
| Creating instances | 14-14–14-15 |
| Exporting | 14-25 |
| Hierarchical | 14-8 |
| Importing | 14-24 |
| Live Update | 10-27 |
| Managing | 14-2 |
| Merge | 10-27 |
| PLMXML_export_packed_bom | |
| <transfer-mode-name> | 17-14 |
| <relation-business-object>_relation | |
| primary | 6-2 |
| System | 14-8 |
| preferences_manager utility | 14-9 |
| Prerequisites for Workflow | |
| Designer | 18-7 |
| Primary application buttons | 1-12 |
| Rich client | 1-12 |
| Privileged team member | 13-8 |
| Privileged use commands | G-7 |
| Privileges | |
| Administer ADA license | 12-21 |
| Assign to project | 12-20 |
| Batch Print | 12-20 |
| Change | 12-19, 12-52 |
| Change ownership | 12-19 |
| CICO | 12-20 |
| Delete | 12-19, 12-52 |
| Demote | 12-19, 12-56, 19-4 |
| Export | 12-19 |
| Import | 12-19 |
| IP Admin | 12-20 |
| ITAR Admin | 12-20 |
| Promote | 12-19, 12-56, 19-4 |
| Publish | 12-19 |
| Read | 12-19 |
| Remote checkout | 12-20 |
| Remove from project | 12-20 |
| Subscribe | 12-19 |
| Transfer in | 12-20 |
| Transfer out | 12-19 |
| Translation | 12-20 |
| Unmanage | 12-20 |
| View/Markup | 12-20 |
| Write | 12-19 |
| Write Classification ICO | 12-20 |
| Process templates | |
| Create | 18-15 |
| Export | 20-2 |
| Import | 20-2 |
| Process Templates | 18-3 |
| Product structure maintenance utilities | |
| update_project_bom | 13-23 |
| Product Structure reports | 16-7 |
| Program-level security | |
| Default access rules | 12-72 |
| Project | |
| Modifying | 13-14 |
| Privileged team member | 13-8 |
| Project administrator | 13-8 |
| Project team administrators | 13-8 |
| Security rule tree | 13-12 |
| Team | 12-23 |
| Teams | 12-23 |
| Project administrator | 13-8 |
| Project team administrators | 13-8 |
| Project-level rules | 12-75 |

| | |
|--|--------------------|
| Project-level security | |
| Access rules | 12-70, 13-11 |
| Default access rules | 12-72 |
| Rule placement | 12-74 |
| Project-level security tasks | |
| Applying Access Manager rules | 12-70, 13-11 |
| Projects | |
| Backing up | 1-43–1-44, 1-48 |
| Business Modeler IDE | |
| template | 1-35 |
| Create | 13-10 |
| Creating | 1-25 |
| Files | 1-28 |
| Importing | 1-31, 1-48 |
| Restore | 1-45 |
| Restore alternatives | 1-46 |
| Promote privilege | 12-19, 12-56, 19-4 |
| Properties | |
| Attribute | 3-8 |
| Business objects | 2-21–2-22 |
| Compound | 3-8 |
| Compound property | B-2 |
| Data Model | 17-4 |
| In general | 1-9 |
| Make visible or required | 2-27, 2-29 |
| Persistent | 2-23, 3-9, 4-8 |
| Reference | 3-8 |
| Relation | 3-8, 6-5, 6-9 |
| Runtime | 3-8 |
| Properties display | |
| Customizing using XML style sheets | D-2 |
| Property constants | |
| Behavior | 3-11 |
| Enabled | 3-13 |
| Modifiable | 3-13 |
| Reference | 3-10, 3-12, 4-9 |
| Update | 3-14 |
| Property sets | 17-8 |
| Protecting Teamcenter data | 12-6 |
| Public schedule | 12-24 |
| Publish privilege | 12-19 |
| Purpose of the IDE | 1-2 |

Q

Queries

| | |
|-------------------------------------|-------------|
| Attribute Selection | 15-17 |
| Based on existing definitions | 15-13 |
| Creating | 15-4, 15-16 |
| Creating using hints feature | 15-14 |
| Definition window | 15-4 |

Query

| | |
|-----------------------------|-------|
| Custom item type | 15-19 |
| Query Builder | 2-9 |
| Quick search | |
| Rich client | 1-11 |
| Quick-release process | H-4 |

R

| | |
|----------------------|-------|
| Read privilege | 12-19 |
| Recovery | 1-49 |

Reference

| | |
|-------------------------------------|------|
| Business object display rules | 9-13 |
| Classes | 2-7 |
| Dataset business objects | 7-2 |
| Note types | 8-3 |
| Unit of measure | 8-7 |

| | |
|------------------------------|------|
| Reference Class column | 2-25 |
|------------------------------|------|

| | |
|------------------------|------|
| Reference object | 18-3 |
|------------------------|------|

Relation

| | |
|-----------------------------------|-----|
| Properties | 6-9 |
| Relation properties, adding | 6-5 |

Relationships

| | |
|---------------------------------------|-------|
| Business objects | C-11 |
| Reloading the Organization tree | 11-48 |
| Remote checkout privilege | 12-20 |
| Remote site | 12-22 |
| Remove from project privilege | 12-20 |

Removing

| | |
|---------------------------|-------|
| Members from groups | 11-51 |
|---------------------------|-------|

| | |
|----------------------|-----|
| Report Builder | 2-9 |
|----------------------|-----|

| | |
|-----------------------|------|
| Concepts | 16-4 |
| Getting started | 16-2 |
| User interface | 16-3 |

| | |
|---------------|----------|
| Reports | 8-9–8-11 |
|---------------|----------|

| | |
|----------------------------------|------|
| Admin-Employee Information | 16-7 |
|----------------------------------|------|

| | |
|------------------------------|------|
| Admin-Group/Role | |
| Membership | 16-7 |
| Admin-Object Ownership | 16-7 |

| | |
|---------------------------------------|-----------------|
| Admin-Objects by Status | 16-7 |
| ECO Details Report | 16-7 |
| ECO Signoff Details | 16-7 |
| PS - BOM Structure | 16-7 |
| Required | |
| Business object properties | 2-27, 2-29 |
| Resource pool subscription | 18-19 |
| Responsible party | 12-23 |
| Responsible Party accessor . . | 12-55, 19-3 |
| Restore data | 1-49 |
| Revision naming rules | 9-10-9-11 |
| Revision rules | |
| Controlling access | 12-53 |
| Structure Manager criteria | G-18 |
| Understanding | G-4 |
| Viewing information | G-8 |
| Revision Rules Setup | G-11 |
| Rich client | |
| Back and forward buttons | 1-11 |
| Clipboard button | 1-12 |
| Information center | 1-12 |
| Menu bar | 1-11 |
| Navigation pane | 1-11 |
| perspective banner | 1-11 |
| Primary application buttons | 1-12 |
| Search field | 1-11 |
| Secondary application buttons . . | 1-12 |
| Toolbar | 1-11 |
| Rich client interface | 1-11 |
| Role | 12-22 |
| In group | 12-22 |
| In owning group | 12-22 |
| In project | 12-24 |
| In projects of object | 12-24 |
| InAnySchedule | 12-24 |
| InSchedule | 12-24 |
| Roles | 11-12 |
| Adding new | 11-37 |
| Creating | 11-28 |
| Rule handlers | 19-5, 19-8 |
| Rule tree | 12-11 |
| Conditions | 12-26 |
| Rule tree precedence | 12-11 |
| Rules | 9-2 |
| Adding | 12-40 |
| Business object display | 9-13-9-14 |
| Cautionary statements | 12-47 |
| Compound property | B-8 |
| Deep copy | 9-15, 9-17-9-19 |
| Definition | 12-7, 12-12 |
| Example | 12-36 |
| GRM | C-11 |
| In general | 1-9 |
| Modifying | 12-42 |
| Naming | 9-3 |
| Revision naming rule | 9-10-9-11 |
| Subbranch precedence | 12-11 |
| Syntax | 12-13 |
| Tips for using | 12-45 |
| Tree | 12-11 |
| Verifying the effect of | 12-48 |
| Viewing | 12-49 |
| Rules-based protection | 12-7, 12-12 |

S

| | |
|---|--------------|
| Saved Queries tree pane | 15-4 |
| Saved query properties pane | 15-4 |
| Search Class button | 15-4 |
| Search criteria | |
| Class attributes | 15-10, 15-18 |
| Defining | 15-9 |
| Search Criteria pane | 15-4 |
| Search field | 1-11 |
| Searching for | |
| Objects | 2-6 |
| Secondary application buttons | 1-12 |
| Security Rules pane | 13-5 |
| Security Services | 11-11 |
| Server | |
| Connection profiles | 1-35-1-36 |
| Deploying changes to | 2-33-2-34 |
| Set active extension files | 1-38, 1-40 |
| Setting | |
| Options | 14-11 |
| Site | 12-22 |
| Site Location condition | 12-33 |
| Source Class column | 2-25 |
| Source Control Management (SCM) | |
| systems | 1-34 |
| Status | 8-5 |

| | |
|--|------------------------------|
| Adding | 8-6 |
| Structure Manager | |
| revision rule criteria | G-18 |
| Subgroups | 11-14 |
| Subscribe privilege | 12-19 |
| Suppressing display of inactive members in groups | 11-48, 11-54 |
| Synchronize data model | 10-26 |
| System administrator | 12-22 |
| System preferences | 14-8 |
| System variables in naming rules . . . | 9-8 |
| T | |
| Tabs | |
| Operation Descriptor | 2-26 |
| Target object | 18-3 |
| Task | |
| Delete | H-13 |
| Task handlers | |
| Creating | 19-5 |
| Task owner | 12-23 |
| Task Owner accessor | 12-55, 19-3 |
| Task owning group | 12-23 |
| Task Owning Group accessor | 12-55, 19-3 |
| Task templates | |
| Handlers | 19-5 |
| Tasks | 18-3, 18-6 |
| Attributes | 19-2 |
| Linking manually | 20-6 |
| Team members | |
| Privileged team member | 13-8 |
| Project administrator | 13-8 |
| Project team administrators | 13-8 |
| Teamcenter interfaces | |
| Rich client | 1-11 |
| Teamcenter preferences | 1-35 |
| Teamcenter reporting and analytics | 2-9 |
| Teamcenter security applications . . . | F-3 |
| Templates | |
| Deploying | 2-33–2-34 |
| In general | 1-33 |
| Installing | 2-38 |
| Packaging | 2-36 |
| TIEImportDefault transfer | |
| mode | 17-11 |
| TIEPDXExportDefault transfer | |
| mode | 17-11 |
| TIEUnconfiguredExportDefault transfer | |
| mode | 17-11 |
| Tips for using rules | 12-45 |
| Tool | 7-9 |
| Toolbar | 1-11 |
| Tools | |
| Adding | 7-10 |
| Transfer in privilege | 12-20 |
| Transfer mode best practices | 17-10 |
| Transfer mode objects | 17-6, 17-9 |
| Transfer out privilege | 12-19 |
| Translation privilege | 12-20 |
| Type identifiers | 17-12 |
| Type-level rules | 12-75 |
| U | |
| UML editor | |
| Using | 2-10 |
| Unique column | 2-25 |
| Units of measure | |
| Adding | 8-8 |
| Reference | 8-7 |
| Unloading objects | 17-13 |
| Unmanage privilege | 12-20 |
| Unprivileged user commands | G-7 |
| update_project_bom utility | 13-23 |
| update_project_data utility | 13-19 |
| Updating BOM items in projects | 13-23 |
| Upper Bound column | 2-25 |
| User | 12-22 |
| User Excluded | 12-24 |
| User Has Government | |
| Clearance | 12-24 |
| User Has Government Clearance | |
| condition | 12-34 |
| User Has IP Clearance | 12-25 |
| User Has IP Clearance condition . . | 12-35 |
| User interface | 17-3 |
| Business Modeler IDE | 1-3–1-5 |
| User interface, Report Builder | 16-3 |
| User IP Licensed | 12-25 |

| | | | |
|---|---|---------------------------------------|--|
| User IP Unlicensed | 12-25 | Organization User wizard | 11-39 , 11-42 |
| User Is Excluded condition | 12-34 | workflow | |
| User Is IP Licensed condition | 12-35 | actions | H-10 |
| User Is ITAR Licensed condition | 12-33 | cancel | H-11 – H-12 |
| User ITAR Licensed | 12-25 | resume | H-12 |
| User ITAR Unlicensed | 12-25 | troubleshooting | H-9 |
| User Location condition | 12-33 | Workflow | |
| User Nationality condition | 12-32 | Resume tasks | H-12 |
| User Over Government | | Workflow accessors | |
| Clearance | 12-25 | Approver | 12-55 , 19-3 |
| User Over IP Clearance | 12-25 | Approver (Group) | 12-55 , 19-3 |
| User Under Government | | Approver (RIG) | 12-55 , 19-3 |
| Clearance | 12-25 | Approver (Role) | 12-55 , 19-3 |
| User Under IP Clearance | 12-25 | Responsible Party | 12-55 , 19-3 |
| Users | 11-9 | Task Owner | 12-55 , 19-3 |
| Activating account | 11-47 | Task Owning Group | 12-55 , 19-3 |
| Adding existing | 11-42 | Workflow ACL examples | 12-57 |
| Adding new | 11-39 | Workflow ACLs | 19-2 |
| Changing status | 11-44 | Workflow Designer | |
| Creating | 11-31 | Definition | 18-2 |
| Inactivating account | 11-45 | Interface | 18-8 |
| Users, configuring | G-7 | Tasks | 18-6 |
| Utilities | | Workflow Designer interface | 18-7 |
| global_transfer | 11-52 | Workflow privileges | |
| make_user | 11-32 | Demote | 12-56 , 19-4 |
| package_live_updates | 10-29 | Promote | 12-56 , 19-4 |
| V | | Workflow process | |
| Verifying the effect of access | | Delete | H-13 |
| rules | 12-48 | Workflow tasks | 18-10 |
| View/Markup privilege | 12-20 | Workflow templates | 18-14 |
| Viewing rules | 12-49 | Create | 18-15 |
| Views | | Create generic process | |
| Outline | 2-13 | templates | 18-16 |
| Visible | | Enterprise process modeling | |
| Properties | 2-27 , 2-29 | (EPM) | 18-12 |
| Volumes | 11-16 | Exporting | 20-3 |
| Controlling access | 11-26 | Importing | 20-4 |
| Creating | 11-22 | Model a process | 18-14 |
| Definition | 11-16 | Process approaches | 18-4 |
| Modifying properties | 11-25 | Process terms | 18-3 |
| W | | Tasks | 18-10 |
| Wizards | | Workflow Designer interface | 18-7 |
| Organization Role wizard | 11-37 | Working ACL | 12-37 , 12-51 |
| World | | Working data | 12-50 , F-8 |
| | | World | 12-22 |

| | |
|---------------------------|-----------|
| Write Classification ICO | X |
| privilege | 12-20 |
| Write privilege | 12-19 |
| XML style sheets | |
| Using predefined | D-3 |

Reference tear-out pages

These reference tear-out pages are provided for your convenience.

Course agenda

| | | |
|--------------|------------------|--|
| Day 1 | Morning | |
| | Introduction | |
| | Course overview | |
| | Lesson 1 | Business Modeler IDE fundamentals |
| | Lesson 2 | BMIDE process and data model |
| | Afternoon | |
| | Lesson 3 | Item business object configuration |
| | Lesson 4 | Form business object configuration |
| Day 2 | Morning | |
| | Lesson 5 | LOV (list of value) extensions |
| | Lesson 6 | Relation business object configuration |
| | Lesson 7 | Dataset business object configuration |
| | Afternoon | |
| | Lesson 8 | Option extensions and BMIDE reports |
| | Lesson 9 | Rule extensions |
| Day 3 | Morning | |
| | Lesson 9 | Rule extensions (continued) |
| | Lesson 10 | Data model live updates |
| | Afternoon | |
| | Lesson 11 | Organization |
| | Lesson 12 | Access Manager |
| Day 4 | Morning | |
| | Lesson 13 | Projects to control access |
| | Lesson 14 | Managing preferences |
| | Lesson 16 | Report Builder definitions |
| | Afternoon | |
| | Lesson 16 | Report Builder definitions |
| | Lesson 17 | PLM XML import and export |
| Day 5 | Morning | |
| | Lesson 18 | Introduction to Workflow Designer |
| | Lesson 19 | Building workflow process templates |
| | Afternoon | |
| | Lesson 20 | Workflow sharing and additional paths |
| | Lesson 21 | 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 | Default data value | Data value |
|-----------------------|---------------------------------------|-------------------|
| OS User ID | <i>student</i> | |
| OS Password | <i>Supplied by instructor</i> | |
| Teamcenter User ID | <i>jgordon</i> | |
| Teamcenter Password | <i>jgordon</i> | |
| Teamcenter User Name | <i>Gordon, Jack</i> | |
| Student Home location | <i>D:\users\student\student_files</i> | |
| TC_ROOT location | <i>D:\tc_root</i> | |
| TC_DATA location | <i>D:\tc_data</i> | |
| Connection Profile | <i>Teamcenter Training</i> | |
| Installed Node ID | <i>tc_TC101svr</i> | |
| | | |
| | | |
| | | |
| | | |
| | | |

Student profile

PLM Software
www.siemens.com/plm



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/date that you work with

Figure 3. A 3D visualization of the experimental geometry.

Reason for training _____

Please verify and add to this list of training for OA Areas, ImagineRTE Participants, Technicians or Dimensional Mgmt./Visualizations. Medium means Instructor-led (IL), Online (OL), or Self-paced (SP)

| Software | From whom | When | Course name | Medium |
|----------|-----------|------|-------------|--------|
|----------|-----------|------|-------------|--------|

Table 1. Summary of the main characteristics of the four groups of patients.

Table 1. Summary of the main characteristics of the four groups of patients.

Table 1. Summary of the main characteristics of the four groups of patients.

For more information about the study, please contact Dr. John Smith at (555) 123-4567 or via email at john.smith@researchinstitute.org.

Table 1. Summary of the main characteristics of the four groups of patients.

Other CAD/CAM/CAE /PDM software you have used

Please check (✓) your ability/knowledge level in the following areas:

Subject **None** **Novice** **Intermediate** **Advanced**

CAD assemblies
CAD parts
CAM

PDM – usage

PDM – system management ○ ○ ○ ○
PDM – customization ○ ○ ○ ○

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:

1. ...clearly explained the course objectives.....
2. ...was knowledgeable about the subject.....
3. ...answered my questions appropriately.....
4. ...encouraged questions in class.....
5. ...was well spoken and a good communicator.....
6. ...was well prepared to deliver the course.....
7. ...made good use of the training time.....
8. ...conducted themselves professionally.....
9. ...used examples relevant to the course and audience.....
10. ...provided enough time to complete the exercises.....
11. ...used review and summary to emphasize important information.....
12. ...did all they could to help the class meet the course objectives.....

| | STRONGLY DISAGREE | DISAGREE | SOMEWHAT DISAGREE | SOMEWHAT AGREE | AGREE | STRONGLY AGREE |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| 1. ...clearly explained the course objectives..... | <input type="checkbox"/> |
| 2. ...was knowledgeable about the subject..... | <input type="checkbox"/> |
| 3. ...answered my questions appropriately..... | <input type="checkbox"/> |
| 4. ...encouraged questions in class..... | <input type="checkbox"/> |
| 5. ...was well spoken and a good communicator..... | <input type="checkbox"/> |
| 6. ...was well prepared to deliver the course..... | <input type="checkbox"/> |
| 7. ...made good use of the training time..... | <input type="checkbox"/> |
| 8. ...conducted themselves professionally..... | <input type="checkbox"/> |
| 9. ...used examples relevant to the course and audience..... | <input type="checkbox"/> |
| 10. ...provided enough time to complete the exercises..... | <input type="checkbox"/> |
| 11. ...used review and summary to emphasize important information..... | <input type="checkbox"/> |
| 12. ...did all they could to help the class meet the course objectives..... | <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.....
2. The computer equipment was reliable.....
3. The software performed properly.....
4. The overhead projection unit was clear and working properly.....
5. The registration and confirmation process was efficient.....

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:

1. The training material supported the course and lesson objectives
2. The training material contained all topics needed to complete the projects.....
3. The training material provided clear and descriptive directions.....
4. The training material was easy to read and understand.....
5. The course flowed in a logical and meaningful manner.....

6. How appropriate was the length of the course relative to the material?..... Too short Too long Just right

Comments on course and material: _____

Student:

1. I met the prerequisites for the class (I had the skills I needed)
2. My objectives were consistent with the course objectives.....
3. I will be able to use the skills I have learned on my job.....
4. My expectations for this course were met.....
5. I am confident that with practice I will become proficient.....

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.