

Teamcenter Systems Engineering Help



User's Manual

- > Installing the Architect/Requirements Client with Office
- > Integration Using the Architect/Requirements Main Window
- > Using the Search Module



Project Administrator's Manual

- > Overview of Projects
- > Managing Users



System Administrator's Manual

- > Systems Architect/Requirements Management Architecture
- > Systems Architect/Requirements Management Components
- > Configuring Architect /Requirements



API Reference

- > Overview
- > Examples of using API's



Server Installation Manual for Windows

- > Installing Architect/Requirements
- > Upgrading the Installation

Teamcenter 11.1 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management User's Manual

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	3.0	March 2003
B	3.0.1	May 2003
C	4.0	December 2003
D	4.1	February 2004
E	5.0	July 2004
F	6.0	March 2005
Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
G	2005	September 2005
H	2005 SR1	June 2006
I	2007	December 2006
J	2007.1	April 2007
K	2007.2	September 2007
L	2007.3	January 2008
M	8	January 2009
N	8.0.1	June 2009
O	8.1	October 2009
P	8.2	October 2010
Q	9	July 2011
R	9.1	May 2012
R1	9.1.5	January 2014

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	10.0	January 2015
T	10.1	September 2016
U	11.1	March 2018

This edition obsoletes all previous editions.

User's Manual Contents

- Manual History 4**
- Contents 7**
- Preface..... 17**
 - Audience 17
 - Conventions 17
 - Revision Marks 17
 - Browser and Dialog Window Examples 17
 - Names and Values 18
 - Submitting Comments..... 18
 - Proprietary and Restricted Rights Notice..... 18
- Chapter 1: Introducing Architect/Requirements..... 19**
 - Requirements Identification and Allocation 20
 - Identification..... 20
 - Allocation..... 21
 - Requirements Management..... 22
 - Object Types 23
 - Object Properties..... 25
 - Project Alignment 26
- Chapter 2: Installing the Architect/Requirements Client with Office Integration..... 27**
 - Introduction..... 27
 - Architect/Requirements Components 28
 - Prerequisites for Installing the Architect/Requirements Client with Office Integration..... 28
 - Limitations of Architect/Requirements Support for Microsoft Office Integration 30
 - Supporting Multiple Installations of Microsoft Office 30
 - Constraints for Supporting Microsoft Office Integration 30
 - Windows Client Hardware..... 31
 - New Visio 2013 and Visio 2016 Stencil and Diagram Formats Not Supported..... 31
 - Installing the Architect/Requirements Client with Office Integration 33
 - Troubleshooting the ICE_JNIRegistry.DLL Java console error..... 35
 - Using the Live Office Interface 37
 - Prerequisites for Using the Live Office Interface 37
 - Post Installation Procedures for Using the Live Office Interface 38
 - Using the Live Office Diagnosis..... 42

Uninstalling the Architect/Requirements Client	43
Chapter 3: Using the Architect/Requirements Main Window	45
Starting a New Architect/Requirements Session	45
Elements of the Architect/Requirements Main Window	48
Customizing the Main Window Display	50
Toolbar	51
Rearranging Button Groups Within the Toolbar Area	55
Displaying Button Groups in Floating Windows	55
Module Bar	57
Disabling or Enabling the Module Bar	57
Hiding or Showing the Module Bar	57
Resizing the Module Bar	57
Address Bar	58
Disabling or Enabling the Address Bar	58
Hiding or Showing the Address Bar	59
Resizing the Address Bar	59
Navigation Tree	60
Expanding a Project or a Folder	60
Disabling or Enabling the Navigation Tree	61
Disabling or Enabling the Root Node	61
Hiding or Showing the Navigation Tree	61
Switching to the Navigation Tree	61
Resizing the Navigation Tree	62
Content Table	63
Hierarchical Content Table	64
Architect/Requirements Recycle Bin	68
Disabling or Enabling the Content Table	70
Hiding or Showing the Content Table	70
Switching to the Content Table	70
Resizing the Content Table	71
Notebook Pane	72
Properties Tab	74
Attachments Tab	75
Links Tab	77
Connectivity Tab	83
Preview Tab	85
Where Used Tab	86
Versions Tab	88
Using Tabs in Floating Windows	90
Disabling or Enabling the Notebook Pane	93
Hiding or Showing the Notebook Pane	93
Switching to the Notebook Pane	93
Resizing the Notebook Pane	94
Viewing Objects in a Hierarchy	95
Filtering Objects in a Hierarchy	97

Showing and Hiding Filters	97
Filter Toolbar	98
Filter Choices	99
Filter Operators	101
Filtering Data	101
Saved Views	102
Printing a Selected Panel.....	102
Viewing Architect/Requirements System Properties.....	103
Working With the Client Log File	105
Setting Client Debugging Options	107
Changing Your Architect/Requirements Password.....	110
Exiting Architect/Requirements.....	110
Chapter 4: Maintaining a Project.....	111
Overview of Projects.....	111
Opening a Project.....	112
Opening a Folder.....	113
Creating a Folder.....	115
Copying Objects.....	119
Copying Object URLs.....	122
Moving Objects.....	123
Renaming an Object.....	125
Using Groups to Maintain Objects.....	126
Creating a Group.....	127
Adding Group Members	128
Removing Group Members.....	129
Working With Shortcuts	131
Creating a Shortcut	133
Navigating From a Shortcut to the Master Object	134
Hiding and Showing the Children of a Master Object.....	134
Replacing Shortcuts With Copies of the Master Objects.....	135
Sending Object Data by E-Mail	135
Exporting Objects	138
Exporting Objects to an AP233 STP File	138
Exporting Objects to an AP233 XML File	140
Exporting Objects to an Architect/Requirements XML File	142
Exporting Objects to Microsoft Office Excel	143
Exporting Objects to Microsoft Office Word	146
Exporting Objects for importing into Teamcenter.....	150
Importing Objects	152
Importing Objects From an AP233 STP File.....	152
Importing Objects From an AP233 XML File.....	154
Importing Objects From an Architect/Requirements XML File.....	156
Importing Objects From Microsoft Office Excel.....	158
Deleting Objects.....	162
Shortcut Objects.....	163
Deleting a Child Object	163

Restoring Objects.....	163
Emptying Your Architect/Requirements Recycle Bin.....	165
Chapter 5: Managing Requirements.....	167
Overview of Requirements.....	167
Creating a Requirement Object.....	168
Requirement Subtype Assignment and Inheritance.....	168
Creating a Requirement at the Top Level of a Folder.....	169
Creating a Sibling of an Existing Requirement.....	169
Creating a Child of an Existing Requirement.....	170
Creating Requirements and Content by Import From Microsoft Office Word.....	171
Document Parsing by Outline Levels Only.....	171
Document Parsing by Outline Levels and Keywords.....	171
Style Sheet of the Import Folder.....	173
To import requirements from Microsoft Office Word:.....	173
Entering and Changing Requirement Content in Microsoft Office Word.....	175
Content Elements.....	175
Content Formatting.....	175
Page Setup.....	177
Editing Requirement Content in Microsoft Office Word.....	177
Using OLE Objects in Word.....	178
Referencing Information in Requirement Text Edited in Microsoft Word.....	178
Editing Requirement Content in Text Property Cells.....	179
Enabling Text Property Editing for a Requirement.....	182
Organizing Requirements in a Hierarchy.....	184
Positions of Promoted Requirements.....	184
Positions of Demoted Requirements.....	184
Number Property Values of Promoted and Demoted Requirements.....	185
Promoting and Demoting Requirements.....	185
Viewing Requirement Content.....	186
Comparing the Content of Different Requirements.....	187
Setting the Comparison Mode.....	187
Comparing Two Requirements.....	188
Comparing the Requirements in Two Folders.....	189
Chapter 6: Constructing System Views With Building Blocks and Diagrams.....	191
Building Blocks.....	191
Creating a Building Block.....	192
Organizing Building Blocks in a Hierarchy.....	195
Live Visio Diagrams.....	197
User Actions in Live Visio Diagrams.....	197
Creating a Diagram.....	201
Creating a Child Diagram.....	203
Editing a Diagram.....	204
Copying a Microsoft Office Visio Diagram to Live Visio.....	209
Changing a Diagram Type.....	211

Disconnecting a Live Visio Diagram.....	213
Viewing a Diagram if Microsoft Office Visio Is Not Installed	214
Data Dictionaries and Data Definitions	215
Creating a Data Dictionary	215
Attaching a Data Dictionary to a Diagram.....	216
Creating Data Definitions	217
Ports and Connections.....	220
Creating Ports	220
Creating Connections in the Architect/Requirements Client.....	221
Working With Connections in a Live Visio Diagram.....	223
Attaching a Data Definition to a Connection.....	228
Detaching a Data Definition From a Connection.....	230
Viewing the Connections for an Object.....	231
Working With Trace Links in a Live Visio Diagram.....	231
Creating a Trace Link in a Live Visio Diagram.....	232
Copying a Trace Link to a Live Visio Diagram.....	232
Chapter 7: Showing Object Relationships With Trace Links	233
Overview of Trace Links	233
Creating Trace Links.....	234
Linking Objects Within Architect/Requirements	234
Linking to an Object in Another Teamcenter Product.....	237
Viewing Object Relationships	238
Viewing Objects in a Defining or Complying Path	238
Viewing Trace Links for an Object	241
Navigating to a Linked Object	242
Deleting Trace Links for an Object.....	243
Generic Links.....	244
Customizing Generic Links	244
Chapter 8: Recording Supplementary Information With Notes	245
Overview of Notes	245
Attaching a Note to an Object.....	246
Creating a Plain Text Note.....	247
Creating a Rich Text Note	248
Creating a Note in Microsoft Office Word.....	250
Editing a Note	252
Enabling Text Property Editing	255
Editing Text Property Cells.....	257
Viewing Note Content	259
Chapter 9: Working With Object Properties.....	261
Overview of Object Properties.....	261
System-Defined Properties	261
User-Defined Properties	263
Property Values.....	263
Releasing Reservation of Objects	264

Customizing Views of Property Columns.....	265
Conditions That Determine the Column View in the Content Table.....	266
Applying a View in the Content Table	267
Applying a View in the Notebook Pane	268
Creating a Column View	269
Setting the Default View for a Folder.....	270
Setting the Default View for Your User Name.....	271
Modifying a View Locally.....	272
Modifying a View in the Database	276
Filtering Property Tables	277
Editing the Properties of a Selected Object.....	279
Property Editing for Multiple Object Selections.....	282
Conditions of Property Display for Multiple Selections.....	282
Editing Properties for Multiple Objects.....	283
Editing Properties in Table View Cells.....	286
Calculating Numeric Property Values With Formulas	288
Updating Properties From an AP 233 STP File	290
Using the Live Excel Interface.....	292
Types of Live Excel Sessions	292
User Actions in Live Excel Files	294
Editing Properties in a Live Excel Session With the Architect/Requirements Client	295
Editing Properties in an Independent Live Excel Session	302
Creating Objects in Live Excel.....	305
Creating Trace Links in Live Excel	306
Disconnecting a Live Excel Spreadsheet.....	307
Attaching a Spreadsheet to an Object	308
Storing Property Values Through Reference Links	309
Reference Link Concepts.....	309
Creating a Plain Text Reference Link.....	318
Creating a Full Content Reference Link	319
Viewing Reference Links	321
Deleting Reference Links	323
Chapter 10: Working With Versions	325
Overview of Versions and Variants.....	325
Enabling Versions for a Project	326
Filtering Versions by Effectivity Rule	328
Freezing Objects	329
Unfreezing Objects	330
Creating Versions.....	331
Creating Variants	332
Viewing a Version Tree	333
Deleting a Version or a Variant	334
Creating a Baseline	335
Viewing a Baseline	337
Chapter 11: Using the Search Module	339

Overview of the Search Module	339
Activating the Search Module.....	340
Search Results	341
Search Results Dialog Window	342
Saving, Recalling, and Running Searches and Reports	344
Exporting a Report to Microsoft Office Excel.....	347
Exporting a Report to Microsoft Office Word.....	349
Exporting a Report to Microsoft Office Visio	350
Order of Search Results	352
Basic Search View	353
Intermediate Search View.....	358
Advanced Search View	362
Switching to the Advanced Search View.....	364
Basic Pane.....	364
Query View Pane.....	364
Modify Pane.....	365
Query Edit Pane.....	366
Constructing Advanced Queries	367
Advanced Query Statements.....	370
Chapter 12: Using the Change Management Package	379
Overview of Change Management.....	379
Submitting Changes for Approval	380
Approving or Rejecting a Change.....	381
Modifying a Change Approval Object.....	383
Viewing a Change Approval Object	384
Viewing a Change Log	386
Appendix A: Glossary.....	389
Appendix B: System-Defined Properties in the Systems Engineering and Requirements Management Module.....	395
Overview of System-Defined Properties	395
Table of System-Defined Properties	396
Appendix C: Live Office Interface Frequently Asked Questions.....	403
General.....	403
What are the prerequisites to use the Live Office Interface?	403
Can I install Microsoft Office 2013, and Office 2016 on the same machine?.....	403
What are the Architect/Requirements default addins?.....	403
How do I disable Architect/Requirements addins in Microsoft Office application?	404
What are the registry entries created by Architect/Requirements for the Live Office Interface?.....	404
What should I do if the Architect/Requirements Live Office Interface does not work on my machine after I install the latest version of the Architect/Requirements client? OR How to fix the communication error message when I export to live Excel from Architect/Requirements?	405
What should be the security settings in the Microsoft Office Excel, Microsoft Office Word, and Microsoft Office Visio in order to use the Architect/Requirements Live Office Interface?.....	405

Where can I find the log file for Live Office Interface?	405
Microsoft Office Word Troubleshooting	406
What do I do if the formatting is lost while exporting requirements to Microsoft Word?	406
How to fix problems Importing Microsoft Word Document?	406
Microsoft Office Excel Troubleshooting	407
What should I do if the live Excel functionality does not work on my system? The diagnostic tool does not suggest any errors. The security and registry settings are correct.	407
How to fix the missing Architect/Requirements toolbar in Microsoft Office Excel? OR How to view the Architect/Requirements menu when I open a saved live Excel workbook? OR What should I do if the Microsoft Office Excel Add-ins do not load after I open a saved live Excel workbook?	407
I made changes in the saved independent live Excel workbook after I connected to the Architect/Requirements server. I cannot see these changes in the Architect/Requirements client. What could be the reason?	408
In what format should I save my live Excel workbook to retain the live behavior after I open the saved live Excel sheet?	408
I edited a requirement (a Microsoft Office Word document) from a live Excel workbook using the Architect/Requirements Open menu in the live Excel sheet. However, the changes are not reflecting in the Architect/Requirements client. What could be the issue?.....	408
Microsoft Office Visio Troubleshooting.....	408
I open a saved live Visio diagram from my local disk, connected to Architect/Requirements and changed the property of object, but I cannot see the property value updated in Architect/Requirements client. What could be the issue?.....	408
What could be the issue if adding a master shape in live Visio diagram does not create type or subtype in Architect/Requirements?	408
What should I do if I cannot open at least one stencil object when I try to create a new live Visio diagram?.....	409
How can I map more than one Architect/Requirements property to a Microsoft Office Visio master shape?.....	409
How can I identify the sequence of the sub shapes in a group shape?.....	409
While creating or opening Visio diagram, some of the shapes appear outside the Visio page. What should I do?.....	409
Troubleshooting for the Architect/Requirements Client.....	410
The client installation of Systems Architect/Requirements Management is failing. What could be the possible cause?.....	410
How do I unregister and register the Architect/Requirements Client and Microsoft Office?.....	410
What should I do if I cannot create a live Visio diagram from the Architect/Requirements client and the Visio Live—>Create Diagram menu is disabled?	411
I am not able to open a Microsoft Office Word document from the Architect/Requirements client. I get several errors related to com.inzoom.comjni.ComJniException.eComError. How I can get rid of those?	411
How to fix the several pop up messages I get related to vtable call when I launch the Architect/Requirements client?.....	411
Where can I find the log file for the Architect/Requirements client?	412
What are the log files that I can use for troubleshooting?	412

Appendix D: Changing the maximum memory available for rich client.....413
Index.....415

Preface

This manual is a user's reference for Teamcenter Systems Architect/Requirements Management 11.1. Systems Architect/Requirements Management belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual is for project team members who use Systems Architect/Requirements Management (Architect/Requirements) to create, manage, trace, and analyze requirements information throughout the entire life of a product. This manual provides both conceptual information and step-by-step instructions for specific tasks.

This manual assumes that you are familiar with your project and your product development process, that you understand general computer terminology and the Microsoft Windows operating system, and that you have experience with Microsoft Word, Microsoft Excel, Microsoft Internet Explorer, and Microsoft Office Visio.

Conventions

This manual uses the conventions described in the following sections:

Revision Marks

Technical changes are marked by a bar adjacent to the changed text.

Browser and Dialog Window Examples

The examples of browsers and dialog windows in this manual may appear different from those you see on your screen:

- The examples reflect Systems Architect/Requirements Management as initially installed at your site. Your enterprise may customize the browsers and dialog windows such that they appear different from those in the examples.
- The examples reflect individual Systems Architect/Requirements Management modules. If you install additional modules, your dialog windows and browsers reflect the additional modules.
- The examples reflect Systems Architect/Requirements Management installed on a Windows platform.

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **initsid.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
<i>Italic</i>	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In initsid.ora , <i>sid</i> identifies a varying portion of the name (a unique system ID). For example, the name of the file might be: initBlue5.ora
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

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

© 2018 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Chapter 1: Introducing Architect/Requirements

This chapter presents an overview of Architect/Requirements, discussing the basic concepts of its purpose and application.

Insufficient planning and misunderstood expectations affect more than two-thirds of all product development projects.¹ Architect/Requirements solves both problems in the most crucial phase of development, the decision-making stage, when plans and expectations are consolidated as requirements.

Requirements describe the product that the customer will buy. They communicate the customer's specifications to the various disciplines involved in the product's development. To ensure that the finished product meets those specifications, developers follow the requirements throughout the development process. When the product conforms to all requirements, it is ready for delivery, and more importantly, it has the functions and the quality that the customer demands.

Projects that inadequately communicate requirements to developers are prone to run over budget and behind schedule, with late-cycle changes and heroic integration efforts. In turn, these conditions can lead to functional deviations and inferior quality, and ultimately to a product that the customer may reject.

In avoiding such difficulties, two principles are vital. Requirements must be identified at the project's inception, so that problems are revealed and understood before the actual development begins. After identification, requirements must be connected to the product design, and that connection must be maintained through all successive stages—while drawings, parts, and assemblies are developed, tested, and changed.

Incorporating both principles, Architect/Requirements provides for:

- Identification of requirements in the initial stage of product development.
- Early allocation of requirements to development teams and product components.
- Ongoing management of requirements.

¹ Forrester Report, March 2000, in Collaborative Solutions for Product Lifecycle Management brochure (M-1000), published March 2002 by EDS PLM Solutions.

Requirements Identification and Allocation

Architect/Requirements extends customer involvement to all phases of the development process. Through Teamcenter integration, the customer can actively influence the design from requirements identification and allocation to product testing and acceptance.

The following sections describe the requirements identification and allocation process.

Identification

The first step in requirements identification is to gather information for source requirements. Such information may be gathered from telephone conversations, meetings, regulatory agencies, standards organizations, and external documents.

The next step is to analyze that information, looking for issues, ideas, and keywords expressed by the customer. By those guidelines, irrelevant material can be separated and discarded. The remaining information becomes the content for the source requirements.

In Architect/Requirements, content can be added to a new requirement with Microsoft Word. In addition, documents can be imported from Word to create multiple requirements and content simultaneously. For more information, see [Creating a Requirement Object, Entering and Changing Requirement Content in Microsoft Office Word](#), and [Creating Requirements and Content by Import From Microsoft Office Word](#) in chapter 5, *Managing Requirements*.

Then, additional requirements are derived from the source requirements. This step involves creating more detailed requirements that can be traced back to the originating sources. Trace links, from source requirements to derived requirements to other derived requirements, establish the order of precedence among requirements. For more information, see [Overview of Trace Links](#) in chapter 7, *Showing Object Relationships With Trace Links*.

Also in the identification process, requirements are organized according to their intended implementation. This organization partitions the requirements around designs, so that records can be generated for reference in comparing changes with previous structures. For more information, see chapter 4, [Maintaining a Project](#).

To record the organizational structure:

- Table views in Architect/Requirements can be exported to Microsoft Excel. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- Requirements can be exported to Microsoft Word to capture their content and their structure within a folder. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.

Allocation

Following identification, requirements are ready for allocation among the development teams and product components. Allocation connects requirements to the product design through trace links to objects in Teamcenter products:

- In Architect/Requirements, requirements link to one another within a project.
- In Engineering Process Management, requirements link to diagrams, parts, and assemblies.
- In Teamcenter Enterprise, requirements link to product modules.
- In Project, requirements link to tasks in a program schedule.

For more information, see [Creating Trace Links](#) and [Linking to an Object in Another Teamcenter Product](#) in chapter 7, *Showing Object Relationships With Trace Links*.

The trace links to and from each Teamcenter product show how requirements affect the individual objects, the overall development in that discipline, and the dependencies among the disciplines. These complementary relationships increase efficiency by:

- Reducing latency of design and decisions.
- Minimizing the need for changes late in the development process.
- Preventing repetition of past mistakes.
- Enabling process measurement and improvement.
- Promoting accurate design documentation.
- Notifying downstream disciplines of decisions based on the requirements, immediately and in context.

Teamcenter integration delivers maximum effectiveness in the development process. Throughout the process, the customer's influence ensures that problems are discovered before the customer receives the product, and that the finished product is one which the customer will buy. Because requirements are communicated without delay, development disciplines can adjust to design changes as they occur. With requirements actuating development decisions, organizations that use Teamcenter products can respond rapidly to fluctuations in market demand.

Requirements Management

To manage requirements, an organization may use spreadsheets, linked documents, custom databases, and document-oriented tracing tools. Typical problems in such methods are that requirements are isolated on individual computers with limited access, stored in databases with little resemblance to the product structure, or maintained through complicated user interfaces with significant learning curves.

Architect/Requirements simplifies requirement development and access, and substantially reduces the learning curve. Requirements are developed with Microsoft Office Word, Microsoft Office Excel, and Microsoft Office Visio, tools that are readily available in most organizations. For authorized users, requirements are always accessible in a central database that matches the product structure. In addition, the Architect/Requirements user interface is modeled on Microsoft Windows Explorer, with parallel concepts and user actions. Team members who are familiar with Word, Excel, Visio, and Windows Explorer can quickly become productive with Architect/Requirements.

Table 1-1 relates key concepts of Windows Explorer to Architect/Requirements.

Table 1-1. Microsoft Windows Explorer Model in Architect/Requirements

Windows Explorer	Architect/Requirements
Disk volumes Physical data storage devices to hold folders and files	Projects Logical data boundaries to hold folders, requirements, building blocks, and groups
Folders Organization of files within a disk volume	Folders Organization of requirements, building blocks, and groups within a project
Files Units of content managed within folders	Requirements, building blocks, and groups Units of content managed within folders
Permissions Rules for user access to disk volumes, folders, and files	Security profile Rules for user access to projects, folders, requirements, building blocks, and groups
Properties Named values defining the characteristics of a disk volume, folder, or file	Properties Named values defining the characteristics of a project, folder, requirement, building block, or group
Search Find objects that meet specified criteria, such as name, text content, type, or other properties	Search Find objects that meet specified criteria, such as name, text content, type, or other properties
Create Create new folders and files	Create Create new folders, requirements, building blocks, and groups

Table 1-1. Microsoft Windows Explorer Model in Architect/Requirements

Windows Explorer	Architect/Requirements
Cut, Copy, Paste Move or duplicate folders and files	Cut, Copy, Paste Move or duplicate folders, requirements, building blocks, and groups
Undo Reverse previous actions	Undo Reverse previous actions

Though it supports many aspects of systems engineering, Architect/Requirements is mainly a requirements management tool. Therefore, to most users the principal object is the requirement, one of the built-in object types in Architect/Requirements. For each object type, a distinct set of properties defines all objects of that type.

Object Types

Projects are superior to all other objects in Architect/Requirements and represent the highest level of organization. Each project prescribes a boundary for user access and for customization of object types and properties. Furthermore, each project defines a hierarchy in which the other types of objects reside.

In the Systems Engineering and Requirements Management module, those object types are the following:

- *Folder*

In a project hierarchy, the primary level is reserved for folders. As folders contain files and other folders in Microsoft Windows, folders contain requirements, building blocks, groups, and other folders in Architect/Requirements. For more information, see chapter 4, [Maintaining a Project](#).

- *Requirement*

Requirements are created and organized in folders. Within each folder, multiple levels of organization allow flexibility in structuring requirements. For example, all requirements in a folder can reside at the top level. Or, they can be organized in a hierarchy of parent, child, and sibling requirements. Each requirement is a separately managed object, with specific properties and access control. Content is created and edited in Microsoft Word. For more information, see chapter 5, [Managing Requirements](#).

- *Building block*

Building blocks are created and organized in folders. As requirements can be structured within a folder, building blocks can be organized in multiple levels of parents, children, and siblings, or they can all reside at the top level. Building blocks graphically illustrate the hierarchical relationships of elements in a product or system. For example, such hierarchies may reflect design elements in a product, tasks in a work breakdown structure, or job functions in an organizational chart. Diagrams attached to building blocks are created and edited in live Visio, the Architect/Requirements interface with Microsoft Office Visio. For more information, see chapter 6, [Constructing System Views With Building Blocks and Diagrams](#).

- *Group*

A group consists of references to existing objects within a Architect/Requirements project. Each reference associates one object as a *member* of the group. Member objects remain in their various locations and can be maintained from the group without the need to switch from one location to another.

Members can be folders, requirements, building blocks, notes, and other groups. A given object can belong to any number of groups, and can be removed from one or more groups at any time. For more information, see [Using Groups to Maintain Objects](#) in chapter 4, *Maintaining a Project*.

- *Note*

Notes are well suited for recording information that relates only to certain objects, rather than to all objects of a type. For example, such information may convey the rationale for a decision, the minutes of a meeting, or an informal discussion about a particular requirement.

Notes can be attached to folders, requirements, building blocks, and groups. Content is created and edited in Microsoft Word. For more information, see chapter 8, [Recording Supplementary Information With Notes](#).

- *Diagram*

Diagram objects contain diagrams that are created and edited through live Visio, the Architect/Requirements interface with Microsoft Office Visio. Each live Visio diagram is interactively synchronized with the Architect/Requirements database, and is associated with a special stencil that contains shapes representing the Architect/Requirements object types.

Objects can be created, modified, and deleted in the database by adding, modifying, and deleting the corresponding shapes in the diagram. The diagram is updated dynamically when members of the diagram owner are created, modified, and deleted in the database. Diagram objects are typically attached to building blocks, but can be attached also to folders, requirements, and groups. For more information, see chapter 6, [Constructing System Views With Building Blocks and Diagrams](#).

- *Spreadsheet*

To store equations in the Architect/Requirements database, users can create spreadsheet objects containing worksheets from existing Microsoft Excel files. Equations in worksheet cells are preserved when the spreadsheet is opened, rather than being overwritten by updated property values from the database.

Spreadsheets can be attached to folders, requirements, building blocks, and groups. Equations and other content can be edited in Excel. For more information, see [Attaching a Spreadsheet to an Object](#) in chapter 9, *Working With Object Properties*.

- *Trace link*

A trace link establishes a directional relationship between two objects. Each trace link indicates which object precedes, or defines, the other in the relationship. The defining and complying objects can reside within the same Architect/Requirements project, or the objects can reside in different projects.

Trace links can be created between folders, requirements, building blocks, and groups. Trace links also can be created from objects in Architect/Requirements to objects in other Teamcenter products. For more information, see chapter 7, [Showing Object Relationships With Trace Links](#).

- *Connection*

This object type is used in diagrams created through live Visio, the Architect/Requirements interface with Microsoft Office Visio. For more information, see [Live Visio Diagrams](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

Object Properties

For each object type, a distinct set of properties governs the nature and behavior of all objects of that type. Architect/Requirements assigns predefined properties automatically to every object. In addition, a project administrator can define custom properties to fit the organization's business needs.

Each property's value is defined as one of the following:

- *Choice*

The value is selected from a predefined list of choices. A list can be *single-choice*, allowing only one selection, or *multiple-choice*, allowing any number and combination of selections.

- *Date*

The value is a calendar date and can be defined to include the time of day.

- *Numeric*

The value is a number and can be defined to allow floating-point formats.

- *Text*

The value is any string of plain text.

For more information, see chapter 9, [Working With Object Properties](#) and appendix B, [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

Project Alignment

Each site that uses Architect/Requirements can align its projects with its own functional or organizational entities according to its needs. For example, a site might use only one project for all development work. Or, a site might use multiple projects, with each project relating to one of the following:

- Enterprise organizational boundaries: a division, a department, or a work team.
- Product or product line activities that cut across many organizational or functional groups.
- Individual contracts or activities commonly referred to in the local environment as projects.
- Major subsystems of a product.

Although multiple projects allow greater flexibility, each site should evaluate the trade-offs, and then develop a plan for using projects in Architect/Requirements. In defining the scope of each project, the plan should address issues such as the following:

- Which users can modify the information in the project, and which can only view the information.

Users can be granted project access privileges at several broad levels, with corresponding restrictions. In addition, user permissions can be imposed for individual objects through security profiles. Each user can be granted access to any number of projects, with different privileges and permissions for each project.

- Customization of object types and user-defined properties within the project.

At project scope, an Architect/Requirements project administrator defines custom object properties, property values, and object subtypes. Those properties, values, and subtypes apply only within a single project. Therefore, each project can be customized for a particular purpose, process, or product.

Chapter 2: Installing the Architect/Requirements Client with Office Integration

This chapter describes the components, prerequisites, and the installation of the Architect/Requirements Client with Office Integration. It also describes the usage of the Live Office Interface.

Introduction

The Architect/Requirements Client with Office Integration installation wizard resides on the Architect/Requirements server. You use Microsoft Internet Explorer or Mozilla Firefox to log in to the server and run the wizard. When the installation is complete, the client starts automatically.

Architect/Requirements Components

The Architect/Requirements software consists of the following components:

- The database server component runs the Versant service for the Architect/Requirements database. Your enterprise administrator installs this component.
- The Web server component provides the business logic for Architect/Requirements. Your enterprise administrator installs this component.
- The client component presents the Architect/Requirements user interface on a local computer. You install the client on your computer. For more information, see [Installing the Architect/Requirements Client with Office Integration](#), later in this chapter.

The client component includes the installation of the Architect/Requirements live interface with Microsoft Office. For more information, see [Using the Live Office Interface](#), later in this chapter.

Prerequisites for Installing the Architect/Requirements Client with Office Integration

You must install the following prerequisites before you install the Architect/Requirements Client:

Microsoft .NET 4.6 for Windows 7

The Architect/Requirements client requires .NET 4.0 CLR runtime support. .NET 4.0 CLR runtime is provided with Windows 10, but you need to add a NET 4.6 package for Windows 7 32-bit or 64-bit installation. You can download and install the Microsoft .NET Framework 4.6 package from the following URL:

<https://www.microsoft.com/en-us/download/details.aspx?id=48137>

Primary Interop Assemblies Redistributable

If .NET 4.6 is not included and Microsoft Office is installed before installing .NET, you must install a Primary Interop Assemblies Redistributable package from Microsoft. The **Microsoft Office 2010 Primary Interop Assemblies Redistributable** supports Office 2013, and Office 2016. For more information on installing the Primary Interop Assemblies, see:

[http://msdn.microsoft.com/en-us/library/kh3965hw\(v=vs.100\).aspx](http://msdn.microsoft.com/en-us/library/kh3965hw(v=vs.100).aspx)

Oracle 32-bit JRE

Due to changes in Oracle's JRE redistribution policy, Siemens PLM Software does not redistribute JREs. You can obtain the 32-bit JRE installer from your IT department or download it directly from the Oracle Web site:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

For information about version of Oracle JRE certified for your platform, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.



You need to install the JRE again even if you have the 32-bit JRE 1.8.0_161 installed.

To install the JRE

1. Uninstall all previously installed JREs.
2. Install 32-bit Java SE 1.8.0_161 .
3. Ensure that the 32-bit Java SE 1.8.0_161 **bin** directory is inserted at the beginning of the user's **PATH** variable.

Other prerequisites

Before you install the Architect/Requirements Client with Office Integration:

- You must know the URL of your site's home page for Architect/Requirements.
- You must be a registered Architect/Requirements user and must know your user name and password.
- You must have Microsoft Internet Explorer or Mozilla Firefox installed.

For information about version of Internet Explorer or Mozilla Firefox, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.



You require Power User or Administrator privileges for installing the Architect/Requirements Client with Office Integration.

After the client installation is performed by a user with Power User or Administrator privileges, a different user without these privileges can launch the Architect/Requirements Client and work with it to the extent the user's privileges allow.



All the Windows registry entries and the DLL files for the Client with Office Integration are registered in the **HKEY_LOCAL_MACHINE** root keys. Hence, you must have write permissions for these root keys.

- You must remove any previous release version of the live Office interface. Multiple versions of the live Office interfaces on a single machine are not recommended. Every time you start the Architect/Requirements client, the live Office interface with the same version as the Architect/Requirements client is registered for use.



If you have questions about the home page or your user name or password, consult your Architect/Requirements system administrator or project administrator.

You may also need your proxy server settings from Microsoft Internet Explorer. These settings are available through the **Connections** tab in the Internet Options dialog window. Click the **LAN Settings** button to display the Local Area Network (LAN) Settings dialog window, where the settings are displayed under **Proxy server**.

Limitations of Architect/Requirements Support for Microsoft Office Integration

Architect/Requirements provides limited support for Microsoft Office 2013 and Office 2016.

Supporting Multiple Installations of Microsoft Office

Architect/Requirements does not support the installation of multiple versions of Microsoft Office, although it is allowed by Microsoft in certain cases. To prevent stability issues in the Architect/Requirements Live Office interface, avoid installing multiple versions of Office. When more than one version is installed, there is no reliable way for the Architect/Requirements client to launch a specific version. Conflicts occur with the different versions of Office add-ins.

The Architect/Requirements client integrates with Office Excel, Word, and Visio. You must have the same version of Office Excel, Word, and Visio. You can have Office 2013 or Office 2016 installed. However, you cannot have a mixed installation of different versions of Office products. For example, an installation of Excel 2013 and Visio 2016 on the same computer is not supported. To install different versions of Office products, use a virtual machine.

Constraints for Supporting Microsoft Office Integration

Architect/Requirements provides integration with versions 2013 and 2016 of the Microsoft Office Professional editions subject to the following constraints:

- **Supported Microsoft Office Packages**

Product	Professional	Office 365
Office	32-bit Office Professional Plus	32-bit Office 365 ProPlus
Visio	32-bit Visio Professional	32-bit Visio Pro for Office 365

- **Limitations of Support for Microsoft Office Packages**

All Packages	Office 365 Packages
Only versions 2013 and 2016 are supported.	Only the PC device is supported.
64-bit formats are not supported.	Offline installation is supported for all Office 365 packages included above.
Home and student editions are not supported	Over-the-web installation is supported for Office 365 ProPlus version 2016, but not for version 2013.

- **Limitations of Support for Microsoft Office features**

Microsoft Office and Visio include features that are added, removed, and changed between different versions. While Architect/Requirements integration functions are certified to work with the packages listed above, it is possible that it may not work with all combinations of Office and/or Visio features. Siemens PLM Software recommends that you verify business solutions on a test environment before using them in a production environment.

Feature changes for Microsoft Office and Visio

Release	Office Professional Plus	Visio Professional
2013	* Changes in Office 2013 at https://technet.microsoft.com/en-us/library/cc178954(v=office.15).aspx	** What's New in Visio 2013 at https://support.office.com/en-us/article/What-s-new-in-Visio-11656535-2345-491d-bb9a-bef0141180f7
2016	* What's new and improved in Office 2016 at https://support.office.com/en-us/article/What-s-new-and-improved-in-Office-2016-29d7e38e-ef06-4d9c-a476-03d896928b2f	** What's New in Visio 2016 at https://support.office.com/en-us/article/What-s-new-in-Visio-2016-798f4f39-2833-486b-9ae9-55162672102e

* - Applies also to Office 365 ProPlus (Excel and Word)

** - Applies also to Visio Pro for Office 365

Windows Client Hardware

The minimum hardware requirement for running Office 2013 or Office 2016 is significantly greater than the minimum hardware requirement for Office 2010. If you are upgrading to Architect/Requirements 11.1 and also upgrading to Office 2013 or Office 2016, you may need additional hardware. If you plan to add Office 2013 or Office 2016 to your existing Windows 7 or Windows 10 system, ask your IT department to verify if the hardware is sufficient.

System Requirements references for Microsoft Office and Visio

Release	Office Professional Plus	Visio Professional
2013	* https://technet.microsoft.com/en-us/library/ee624351(v=office.15).aspx	** https://products.office.com/en-us/visio/microsoft-visio-faq-diagram-software?legRedirect=true&CorrelationId=d383e71d-bc7d-4ea1-af39-28bf8a2ecad3# , search for system requirements for Visio and click the link.
2016	* Open https://products.office.com/en-us/office-system-requirements , search for Office Professional Plus 2016 and click the link.	** Open https://products.office.com/en-us/office-system-requirements , search for Visio 2016 and click the link.

* - Applies also to Office 365 ProPlus (Excel and Word)

** - Applies also to Visio Pro for Office 365

New Visio 2013 and Visio 2016 Stencil and Diagram Formats Not Supported

Architect/Requirements 11.1 supports integration with Visio 2013 and Visio 2016, however, the new formats for diagrams (**VSDX**) and stencils (**VSSX**) are not supported.

- Offline copies of Visio diagrams integrated with Architect/Requirements must remain in **VSD** format. To avoid unnecessary re-work, use Visio's **Save As** feature to save offline copies of Visio diagrams in the **VSD** format only. There is currently no plan to provide support for Visio's **VSDX** format with Architect/Requirements.
- The Architect/Requirements stencil import capability continues to support the **VSS** format. You must not import stencils formatted in the new **VSSX** format into Architect/Requirements. To import a stencil formatted in the new **VSSX** format, save it as a **VSS** file.
- When you save a Visio diagram for offline use that you need to synchronize with Architect/Requirements later, ensure that the files are saved in the **VSD** format.

To save the diagram outside Architect/Requirements on a local or shared drive:

- . Select the diagram in the Architect/Requirements rich client.
- . Open the diagram double clicking it.
You can also click the right mouse button and select **Open** from the context menu.
- . Click **File** in Visio window.
- . Click **Save As** to display the folder selection dialog.
- . Select a folder or click **Browse**.
Visio displays the **Save As** dialog.
- . Type the file name in the **File name** box.
- . In the **Save As** dialog, ensure that **Visio 2003-2010 Drawing (*.vsd)** is selected in the **Save as** type drop down menu.
- . Click **Save**.
- . To open the diagram and synchronize it with Architect/Requirements, click **File** and select **Visio Live**→**Open** to display the **Open** dialog.
- When you select the Visio stencils to include a Live Visio diagram, you must select stencils stored in the **VSS** format only.

To add stencils to your Live Visio diagram:

- . When creating a Live Visio diagram, select any stencil included in the **Diagram inputs** dialog.
These stencils are always in the **VSS** format. These are stored in the Architect/Requirements database, and are maintained by your Architect/Requirements Project Administrator.
- . When the Live Visio diagram is open for editing, it is possible to add shapes from Visio stencils outside Architect/Requirements. These stencils must be in **VSS** format. For example, if you select **Open Stencil** from the **STENCIL** ribbon, you must select files with a ***.vss** extension only.

For more information on the use of Visio stencils and diagrams, see [Constructing System Views With Building Blocks and Diagrams](#).

Installing the Architect/Requirements Client with Office Integration

Before you install the Architect/Requirements client with Office integration, see the [Prerequisites for Installing the Architect/Requirements Client with Office Integration](#).



If you are installing the Architect/Requirements client with Office integration as a part of upgrade process from a previous version, you must uninstall the previous version of Architect/Requirements client installed on your computer.

For information on uninstalling the Architect/Requirements client, see [Uninstalling the Architect/Requirements Client](#).



Because your enterprise can customize Architect/Requirements, some steps in this procedure may vary from the steps for your particular site. If you have questions about installing the client, consult your system administrator.



- The 32-bit version of the Java Runtime Environment (JRE) must be installed. If necessary, you can install the required Java software in this procedure.

The **JRE.Version** configuration option's default value includes only the 1.8 version. However, sites upgrading from previous Architect/Requirements releases should uninstall all previously installed JREs from their client systems, install JRE 1.8, and edit their **JRE.Version** option to remove all 1.6 and 1.7 versions and add at least one 1.8 version.

For information about version of Java Runtime Environment (JRE) and Java Plug-in supported, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- The Architect/Requirements live interface with the Microsoft Office is installed as part of the Architect/Requirements Client with Office Integration installation. For more information, see [Prerequisites for Using the Live Office Interface](#), later in this chapter.
1. Open the Architect/Requirements home page in a browser window and click the **Launch Teamcenter systems engineering** link.

For information on the supported browsers, open the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.



If you are installing Architect/Requirements on Windows 10, launch the browser as the administrator.

To launch the browser as administrator, right click the browser icon in the directory that the browser is installed and select **Run as administrator**.

The browser may prompt you to allow execution of the **Java™ Web Launcher** on your computer. You must allow this in order to proceed with the launch:

- Microsoft Internet Explorer will prompt with a dialog asking “Do you want to allow this website to open a program on your computer?”. The user should click the “Allow” button to proceed. un-checks the box labeled “Always ask before opening this type of address”.
- Mozilla Firefox will prompt with a “Launch Application” dialog. The user should click the “Open Link” button to proceed. The choice can be remembered if the user checks the box labelled “Remember my choice”.

Depending on whether Security Services is enabled on the Architect/Requirements server, one of two login pages is displayed:

- The Teamcenter Systems Engineering and Requirements Management login page is displayed if Security Services is not enabled. Enter your Architect/Requirements user name and password, select a language, and click **Log In**.
- The Teamcenter Login page is displayed if Security Services is enabled. Enter your Security Services user name and password, select a language, and click **Log In**.



If you do not have Power User or Administrator privileges, the client installation fails and displays a message to contact your local IT department. If you obtain the required privileges, the client installation procedure should be followed from the beginning.

Once logged in, Java may prompt you as to whether you want to run the **Teamcenter Systems Engineering** application. Select “Run” to proceed.

A confirmation message prompts you to install the client.

2. Click **Yes** to display the Proxy Server Settings dialog window, and then do one of the following:
 - To connect directly to the Architect/Requirements server, click **OK**.
 - To connect to the Architect/Requirements server through a proxy server, fill in the **Host address** and **Port** fields and click **OK**. If necessary, check the **Use proxy authorization** check box and enter your proxy server information.



If your browser settings specify an automatic configuration script, you must add configuration information to the Java control panel. This step is necessary if the **Use automatic configuration script** check box is checked in the Local Area Network (LAN) Settings dialog window.

The configuration information is in the file whose location is shown in the **Address** field. From this file, you need your proxy server's HTTP address and server port number.

Opening this file is an advanced operation. If you have questions about the file or the information, consult your system administrator.

After obtaining the information, do the following:

- In the Windows Control Panel, open the Java Control Panel, and then click the **Proxies** tab.
- Clear the **Use browser settings** check box, fill in the **Proxy Address** and **Port** fields for **HTTP**, and then click **Apply**.

3. In the InstallAnywhere wizard, click **Next** to display a window showing the installation directory path, and then do one of the following:
 - To change the installation drive or folder, do one of the following:
 - Enter the new path directly in the **Directory Name** field.
 - Click **Browse** to display the Select a Directory dialog window, navigate to and select the new installation drive or folder, and then click **Open** to enter the new path in the **Directory Name** field.
 - For installation in the default drive and folder, leave the **Directory Name** field unchanged.
4. Click **Next** to display a window with summary information for review.
5. Click **Next** to start the installation.

The wizard displays a progress indicator, followed by a window stating that the wizard has successfully installed the client.

6. Click **Finish** to exit the wizard.
7. The Architect/Requirements main window is displayed

For more information, see [Elements of the Architect/Requirements Main Window](#), in chapter *Using the Architect/Requirements Main Window*.



- If you logged in through Security Services (step 1), and if other Teamcenter applications are installed, you can launch those applications without logging in to each one individually.
- The wizard installs a Microsoft Word template for parsing requirements by keywords in an import document. For more information, see [Creating Requirements and Content by Import From Microsoft Office Word](#) in chapter 5, *Managing Requirements*.

Troubleshooting the ICE_JNIRegistry.DLL Java console error

If the **Web Application Configuration** variable **RegeditEnabled** is set to **false** and you install or launch the Architect/Requirements client, the following error is logged in the Java console.

```
ERROR You have not installed the DLL named 'ICE_JNIRegistry.DLL'.
       no ICE_JNIRegistry in java.library.path
```

To resolve the error, add the *TCSE_CLIENT_LOC* environment variable to the computer running the Architect/Requirements client. Append the value of the *TCSE_CLIENT_LOC* to the *Path* variable.

To add the TCSE_CLIENT_LOC variable and edit the Path:

1. Open the Windows **Control Panel** and double click the **System** icon.
2. Click **Advanced system settings**.
3. On the **Advanced** tab, click **Environment Variables**.
4. In the **User variables** section, click **New**.
5. Type *TCSE_CLIENT_LOC* as the **Variable name**.

6. Type the Architect/Requirements client install location as the **Variable value**;
for example, **C:\Program Files\SiemensPLM\Teamcenter\SystemEngineering\Release_11.1**
7. Select the **Path** variable and click **Edit**.
You need local administrator privileges to edit the *Path* variable if it is present in the **System variables** section.
8. Add the following text at the end of the existing path:
;%TCSE_CLIENT_LOC%
9. Click **OK** three times to apply the changes.

Using the Live Office Interface

The live Office interface enables you to create, edit, view, and manipulate objects in the Architect/Requirements through certain Microsoft Office products. The interface consists of Architect/Requirements add-ins and templates for:

- Microsoft Office Excel. For more information, see [Using the Live Excel Interface](#) in the chapter 9, *Working With Object Properties*.
- Microsoft Office Visio. For more information, see [Live Visio Diagrams](#) in the chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Microsoft Office Word.



The live Office interface is installed as part of the Architect/Requirements Client with Office Integration installation. For more information, see [Architect/Requirements Components](#) earlier in this chapter.

For more information about customizing the live interface with Microsoft Office, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

For information on troubleshooting the live Office interface, see [Using the Live Office Diagnosis](#) and the appendix C, [Live Office Interface Frequently Asked Questions](#).

Prerequisites for Using the Live Office Interface

For the successful installation of the Architect/Requirements live Office interface, the following must be installed on your computer:

- Microsoft Internet Explorer, or Mozilla Firefox.

For information about version of Internet Explorer or Mozilla Firefox, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- Microsoft Office.

For information about version of Microsoft Office, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.



- Only the **.xlsm** or **.xlsx** file type supports the full capability of live Excel. Siemens PLM Software recommends that you assign the **.xlsm** or **.xlsx** file type to all live Excel files that you save outside the Architect/Requirements database, for example, on a local drive.
 - If you save live Excel files as the **.mhtml** file type, they become static and lose the live capability permanently.
- Microsoft Office Visio, if you intend to use the live Visio capability in the live Office interface.

For information about version of Microsoft Office Visio, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- Microsoft .NET Framework.

For information about version of .NET Framework, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

64-bit .NET Framework is required for running Architect/Requirements Office Interface on a 64-bit platform.

-

To work with a non-US locale for regional settings with an English version of Microsoft Office, the Multilingual User Interface (MUI) Pack for that language or locale should be installed for Microsoft Office. The MUI Pack provides features that allow you to change the language of your Office user interface and online Help.

The MUI Pack is a prerequisite for the Microsoft Office APIs to work correctly with Architect/Requirements when the US version of Microsoft Office is used in a non-US locale. It should be installed regardless of whether the MUI Pack for another language for the user interface is used or not.

-

The MHTML libraries are required for the functioning of Architect/Requirements because the client interacts with Microsoft Office in MHTML format. The MHTML libraries are packaged with Outlook Express in Microsoft Windows, and with Windows Mail in Microsoft Windows Vista.

Without the MHTML libraries, the Architect/Requirements integration with Microsoft Office fails. The content in the preview pane displays MHTML markup tags.



When you open a locally saved copy of your Live Excel workbook or Live Visio diagram, Architect/Requirements asks you if you want to connect to the server and launches the default browser (Internet Explorer or Mozilla Firefox) from which you can log in to Architect/Requirements. However, if you close the browser window without logging in, in case of Internet Explorer, a message stating that you are not connected to the server is displayed immediately. In case of Mozilla Firefox, this message is displayed after a considerable delay giving the impression that the Live Excel or Live Visio application is hung.

Post Installation Procedures for Using the Live Office Interface

After completing the live Office installation:

- To use the live Visio capability of the live Office interface, you must enable .NET Framework programmability support for Microsoft Office Visio. For more information, see [Enabling .NET Framework Programmability Support for Microsoft Office Visio](#), later in this chapter.
- You must configure Microsoft Office to work with the live Office interface. For more information, see [Configuring Microsoft Office for the Live Office Interface](#), later in this chapter.

Enabling .NET Framework Programmability Support for Microsoft Office Visio



Microsoft .NET Framework and Microsoft Office Visio must be installed on your computer.

For information about version of Microsoft .NET Framework and Microsoft Office Visio, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

1. In the Windows Control Panel, double-click **Add/Remove Programs** to display the Add/Remove Programs dialog window.
2. In the **Currently installed programs** pane, select your Microsoft Office Visio installation, and click **Change** to run the Microsoft Office Visio Setup program.
3. In the Maintenance Mode Options window, select **Add or Remove Features**, and then click **Next** to display the Advanced Customization window.
4. With **Microsoft Office** expanded, click the plus sign (+) to the left of **Microsoft Office Visio**.
5. To the left of **.NET Programmability support**, click the down arrow, and then select **Run from my computer** from the pop-up list.



The **.NET Programmability support** option is available if you have installed Microsoft .NET Framework version 2.0 and 3.0 or 3.5.

6. Click **Update** to display a progress indicator, followed by a message stating that Microsoft Office Visio has been updated successfully.

If a message asks for the Microsoft Office Visio CD-ROM or network location, insert the CD-ROM or enter the location. Click **Next** to complete the setup.

Configuring Microsoft Office for the Live Office Interface

You must enable the live Office add-ins and templates to work with Microsoft Office. Also, you must set up .NET options.

Enabling Live Office Add-ins and Templates in Microsoft Office



In the Microsoft Office Trust Center, you can choose a higher security setting for the templates and add-ins.

1. In Microsoft Office Excel, Visio, or Word, click **File**→**Options**.
The Excel Options, Visio Options, or Word Options dialog window is displayed.
2. Select **Trust Center** in the left pane, and then select **Trust Center Settings** in the right pane.
The **Trust Center** dialog box is displayed.
3. Select **Add-ins** in the left pane, and then do one of the following in the right pane:
 - For standard security, clear the **Require Application Add-ins to be signed by Trusted Publisher** check box.
 - For higher security, check the **Require Application Add-ins to be signed by Trusted Publisher** check box.
4. Select **Macro Settings** in the left pane, and then select **Disable all macros except digitally signed macros** in the right pane. The unsigned macros would not work in this case.
5. Click **OK** twice to accept the changes.



When higher security is enabled and you open an existing file in Excel, Visio, or Word, the Message Bar displays a security warning stating that some active content has been disabled. To enable this content and add Siemens PLM Software to your list of trusted publishers, do the following:

1. In the Message Bar, click **Options**.

The Microsoft Office Security Options dialog window is displayed.

2. Under **Macros & ActiveX**, select **Enable this content**.
3. Under **Add-In**, select **Enable all code published by this publisher**.

When you click **OK**, the dialog window closes and the Message Bar is removed.

Setting Up .NET Options in Microsoft Office



Setting up .NET options is not applicable to Microsoft Office 365 ProPlus.

1. In the Windows Control Panel, double-click **Programs and Features** to display the Add/Remove Programs dialog window.
2. In the **Uninstall or change a program** pane, select your Microsoft Office installation, and click **Change** to run the Microsoft Office Setup program.
3. Select **Add or Remove Features**, and then click **Continue**.
4. On the **Installation Options** tab, do the following with **Microsoft Office** expanded:
 - Expand **Microsoft Excel**, and then select the **Run from my computer** option for **.NET Programmability support**.
 - Expand **Microsoft Word**, and then select the **Run from my computer** option for **.NET Programmability support**.

5. Click **Continue**.

The Configuration Progress window is displayed, followed by a message window stating that the configuration is complete.

6. Click **Close**.

If a message asks for the Microsoft Office CD-ROM or network location, insert the CD-ROM or enter the location. Click **Next** to complete the setup.

Using the Live Office Diagnosis

The Architect/Requirements diagnostic tool enables you to verify that your local machine is properly configured to run the Architect/Requirements interfaces with Microsoft Office.

To run the Architect/Requirements diagnostic tool, open the Architect/Requirements home page in Internet Explorer. Then, click the **Administrative Tools** link. On the Administrative Tools page, click the **Diagnostic Tools** link and on the Diagnostic Tools page, click the **Live Office Diagnosis** link.

To select the applications to be diagnosed, you can check the **Diagnose All** check box or the individual check boxes next to the applications. Then, click the **Diagnose** button to generate the diagnostic information for the selected applications. The diagnostic information contains a green checkmark on success or if no problem is found. If any error or problem is found, you can click the **Suggested Fix** button to display the suggested fix in a message box.

You can click the **Generate Report** button to display the diagnostic information in a browser or click the **Email Report** button to E-mail the diagnostic information as an attachment to the recipients.

You can click the question mark at the top right corner of the Architect/Requirements diagnostic tool window to display the related help information, such as the frequently asked questions.

Table 2-1 lists the applications and their corresponding diagnostic information displayed in the Architect/Requirements diagnostic tool.

Table 2-1. Live Office Diagnostic Information

Application	Diagnostic Information
System	<ul style="list-style-type: none">• Logged-in user name and the system name.• Installed operating system version.• System regional language and date format settings.• Installed and current .Net Framework versions.• Values of the TEMP and TMP environment variables.• .Net interoperability support for Microsoft Office.• Registration status of the IzmJniComAx library.• Installed live Office interface version.
Microsoft Office Excel	<ul style="list-style-type: none">• Installation status and version of Microsoft Office Excel.• Installation and registration status of .Net Interoperability support for Microsoft Office Excel.• Registration status and load behavior of the Architect/Requirements Excel add-in and import add-ins.• Security level of Microsoft Office Excel.• Status of the Architect/Requirements live Office interface files required for the working of the Excel add-in.

Table 2-1. Live Office Diagnostic Information

Application	Diagnostic Information
Microsoft Office Word	<ul style="list-style-type: none">● Installation status and version of Microsoft Office Word.● Installation and registration status of .Net Interoperability support for Microsoft Office Word.● Registration status and load behavior of the Architect/Requirements Word add-in.● Security level of Microsoft Office Word.● Status of the Architect/Requirements live Office interface files required for the working of the Word add-in.
Microsoft Office Visio	<ul style="list-style-type: none">● Installation status and version of Microsoft Office Visio.● Installation and registration status of .Net Interoperability support for Microsoft Office Visio.● Registration status and load behavior of the Architect/Requirements Visio add-in.● Security level of Microsoft Office Visio.● Status of Enable Automation for Microsoft Office Visio.● Status of COM add-ins for Microsoft Office Visio.● Status of the Architect/Requirements live Office interface files required for the working of the Microsoft Office Visio add-in.
	 <p>The diagnostic tool does not flag the synchronization issue related to the local cache folders used for Microsoft Office Visio files. The issue is observed when there is a conflict with stencil downloads. As a workaround, delete the local cache folders and try the synchronization again. To remove the local cache, open the following folder in Windows Explorer and delete the contents of the folder:</p> <pre data-bbox="574 1398 1117 1425">%APPDATA%\TcRequirements\VisioStencils</pre>

Uninstalling the Architect/Requirements Client

In some situations, you may want to uninstall the Architect/Requirements client from your computer. You must uninstall Architect/Requirements client if you are upgrading to a new version of Architect/Requirements.



Elevated user privileges (power user or administrator) are required for uninstalling the Architect/Requirements client.



The Architect/Requirements client uninstaller can fail if there is a problem in locating a JRE. In such a case, insert the **LAX_VM** argument to explicitly specify the JRE used to run the uninstaller. For example, if your 32-bit JRE is installed in **C:\Program Files (x86)\Java\jre\bin**, and you are uninstalling the Architect/Requirements client version 10.0, the following command uninstalls the Architect/Requirements web application:

```
"Uninstall TcSE Client with Office Integration Release_10" LAX_VM "C:\Program Files (x86)\Java\jre\bin\java"
```

To uninstall Architect/Requirements client

1. Open the Windows **Control Panel**.
2. Open **Programs**→**Programs and Features**→**Uninstall or change a program**.
3. Select **TcSE Client with Office Integration**.



The exact name of the Architect/Requirements depends on the version of Architect/Requirements that you have installed.

4. Click **Uninstall/Change**.
5. Follow the instructions on the uninstall wizard to uninstall the client.

Chapter 3: Using the Architect/Requirements Main Window

This chapter describes the major elements of the Architect/Requirements main window and contains instructions for using those elements.

Starting a New Architect/Requirements Session

If the Architect/Requirements client is installed on your computer, you start a new session by logging in to the Architect/Requirements server through Microsoft Internet Explorer or Mozilla Firefox. For more information, see [Installing the Architect/Requirements Client with Office Integration](#).



Because your enterprise can customize Architect/Requirements, some steps in this procedure may vary from the steps for your particular site. If you have questions about starting a new session, consult your system administrator.



- You must know the URL of your site's home page for Architect/Requirements and your user name and password. If you have questions about logging in, consult your Architect/Requirements system administrator or project administrator.
- The Architect/Requirements client on your computer may be an earlier version. If a new version is available, you can upgrade your present version in this procedure.

1.

In Internet Explorer or Mozilla Firefox, open the home page for Architect/Requirements, and then click the **Launch Teamcenter systems engineering** link.

Depending on whether Security Services is enabled on the Architect/Requirements server, one of two login pages is displayed:

- The Teamcenter 11.1 Systems Engineering and Requirements Management login page is displayed if Security Services is not enabled.
- The Teamcenter Login page is displayed if Security Services is enabled.

2. Do one of the following:

-

On the Teamcenter 11.1 Systems Engineering and Requirements Management login page, enter your Architect/Requirements user name and password, select a language, and click **Log In**.



On the Teamcenter Login page, enter your Security Services user name and password, select a language, and click **Log In**.



If you log in through Security Services, and if other Teamcenter applications are installed, you can launch those applications without logging in to each one individually.



Siemens PLM Software recommends that users do not log in to multiple clients simultaneously using the same login credentials. In Architect/Requirements, a user must use only one client instance at a time.

If two Architect/Requirements clients are used simultaneously, locking errors are displayed on using user specific resources such as the **Recycle bin**. Changing user settings like effectivity affects both the clients.

If the latest version is installed, the Architect/Requirements main window is displayed (see [Elements of the Architect/Requirements Main Window](#)).

If a new version is available, an upgrade message is displayed.



Upgrades provide new and enhanced features and resolutions of previous problems. Siemens PLM Software recommends that you remove the earlier version and install the new one.



To remove the earlier version and install the new one, click **Yes**.

A progress indicator appears while files are downloaded. Then, the uninstaller wizard is displayed.

- Follow the subsequent instructions in the uninstaller, and click **Finish** when the removal is complete.

A progress indicator appears while files are downloaded. Then, the InstallAnywhere wizard for the client is displayed.

- Follow the subsequent instructions in the installer, and click **Finish** to display the Architect/Requirements main window (see [Elements of the Architect/Requirements Main Window](#)).
- To continue using the earlier version, click **No** to display the Architect/Requirements main window (see [Elements of the Architect/Requirements Main Window](#)).



You can also start a new Architect/Requirements session through:

- A Microsoft Office Excel file containing previously exported object data. For more information, see [Exporting Objects to Microsoft Office Excel](#).

- An object URL copied to Microsoft Office Outlook, Microsoft Internet Explorer, or other Microsoft Windows programs that support hyperlinks. For more information, see [Copying Object URLs](#).
- A trace link between an object in another Teamcenter product and an object in Architect/Requirements. For more information, see [Linking to an Object in Another Teamcenter Product](#).
- A file from a previous live Excel session. For more information, see [Using the Live Excel Interface](#).

Table 3-1 describes the elements of the Architect/Requirements main window.

Table 3-1. Elements of Architect/Requirements Main Window

Element	Description
Toolbar	Displayed below the pulldown menus at the top of the main window. Appearance and function are similar to a standard toolbar in a Microsoft Windows program. For more information, see Toolbar , later in this chapter.
Module bar	Displayed below the toolbar at the left border of the main window, extending to the bottom. Appearance and function are similar to the Outlook bar in Microsoft Office Outlook. For more information, see Module Bar , later in this chapter.
Address bar	Displayed to the right of the module bar and below the toolbar. Appearance and function are similar to the Address bar in Microsoft Internet Explorer and Microsoft Windows Explorer. For more information, see Address Bar , later in this chapter.
View list	Displayed to the right of the Address bar and below the toolbar. Appearance and function are similar to drop down lists in Microsoft Windows programs. For more information, see Applying a View in the Content Table in chapter 9, <i>Working With Object Properties</i> .
Navigation tree	Displayed to the right of the module bar and below the Address bar, extending to the bottom of the main window. Appearance and function are similar to the hierarchical Folders pane in Microsoft Windows Explorer. For more information, see Navigation Tree , later in this chapter.
Content table	Displayed to the right of the navigation tree in the upper pane. Appearance and function are similar to the columned pane in Microsoft Windows Explorer. For more information, see Content Table , later in this chapter.
Notebook pane	Displayed to the right of the navigation tree in the lower pane. Appearance and function are similar to a tabbed dialog window in a Microsoft Windows program. For more information, see Notebook Pane , later in this chapter.

Customizing the Main Window Display

Except for the toolbar, each major element can be disabled and enabled, by itself or in conjunction with one or more other elements. With this flexibility of display, the main window can be customized according to a specific purpose or your general preference.

For example:

- You can gain more space for the content table by disabling the navigation tree, the notebook pane, or both.

For more information, see [Navigation Tree](#), [Content Table](#), or [Notebook Pane](#), later in this chapter.

- If you work primarily in the Systems Engineering and Requirements Management module, you may want to keep the module bar disabled, occasionally enabling it for access to the Search module or Administration module.

For more information, see [Module Bar](#), later in this chapter.

Your last display settings are preserved in your next session. If an element is disabled when you exit Architect/Requirements, it is not displayed when you start a new session.

Toolbar

The Architect/Requirements main window displays the toolbar below the pulldown menus. Figure 3-2 shows the buttons on the Architect/Requirements toolbar.

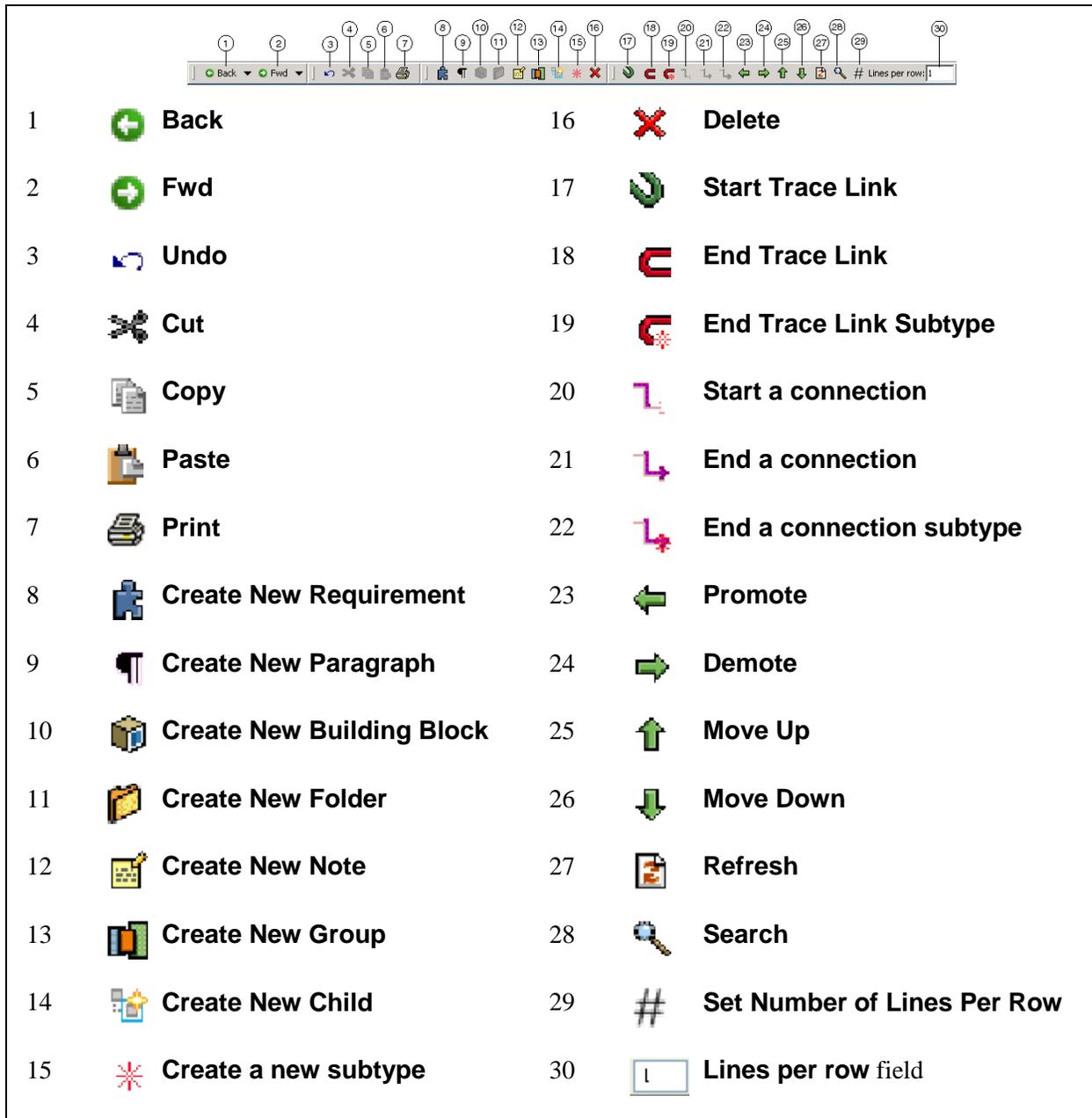


Figure 3-2. Buttons on Architect/Requirements Toolbar



- Toolbar buttons become available or dimmed according to the types of objects that you select.
- You can rest the pointer on a button to see its description in a tooltip.

Each toolbar button relates to an option on a pulldown menu. Table 3-2 describes the buttons on the Architect/Requirements toolbar.

Table 3-2. Buttons on Architect/Requirements Toolbar

Button	Related Menu Option	Action
Back	None	Navigates to the previous selection in the navigation tree. To move back more than one step at a time, click the down arrow to the right to display as many as ten previous navigation tree selections, and then select the project or folder to which you want to navigate. The currently selected project or folder is not displayed in the list.
Fwd	None	Navigates to the subsequent selection in the navigation tree. To move forward more than one step at a time, click the down arrow to the right to display as many as ten subsequent navigation tree selections, and then select the project or folder to which you want to navigate. The currently selected project or folder is not displayed in the list. This button is not enabled until the Back button is used to navigate to a previous selection.
Undo	Edit menu, Undo option	Reverses the previous action.
Cut	Edit menu, Cut option	Places the selected object or objects in the Clipboard.
Copy	Edit menu, Copy option	Places a duplicate of the selected object or objects in the Clipboard.
Paste	Edit menu, Paste option	Inserts the Clipboard contents in the selected location.
Print	File menu, Print option	Print the selected panel. For more information, see Printing a Selected Panel .
Create New Requirement	File menu, New → Requirement options	Creates a new requirement.
Create New Paragraph	File menu, New → Paragraph options	Creates a new paragraph, a system-defined requirement subtype.
Create New Building Block	File menu, New → Building Block options	Creates a new building block.

Table 3-2. Buttons on Architect/Requirements Toolbar

Button	Related Menu Option	Action
Create New Folder	File menu, New → Folder options	Creates a new folder.
Create New Note	File menu, New → Note options	Attaches a new note to the selected object.
Create New Group	File menu, New → Group options	Creates a new group.
Create New Child	File menu, New → Child options	Creates a new child of the selected folder, requirement, or building block.
Create a new subtype	File menu, New → Subtype options	Creates an object of the subtype that you choose in the Select Subtype dialog window, which is displayed when you click this button.
Delete	File menu, Delete option	Sends the selected object or objects to your Architect/Requirements Recycle Bin.
Start Trace Link	Edit menu, Links → Start Trace Link options	Starts a trace link from the selected object or objects.
End Trace Link	Edit menu, Links → End Trace Link options	Ends a trace link to the selected object or objects.
End Trace Link Subtype	Edit menu, Links → End Trace Link Subtype options	Ends a trace link subtype to the selected object or objects.
Start a connection	Edit menu, Connections → Start Connection options.	Starts a connection from the selected object or objects.
End a connection	Edit menu, Connections → End Connection options.	Ends a connection to the selected object or objects.
End a connection subtype	Edit menu, Connections → End Connection Subtype options.	Ends a connection subtype to the selected object or objects.
Promote	Edit menu, Promote option	Promotes the selected object to the next higher level in the hierarchy.
Demote	Edit menu, Demote option	Demotes the selected object to the next lower level in the hierarchy.

Table 3-2. Buttons on Architect/Requirements Toolbar

Button	Related Menu Option	Action
Move Up	Edit menu, Move Up option	Moves the selected object up by one position within its current level.
Move Down	Edit menu, Move Down option	Moves the selected object down by one position within its current level.
Refresh	View menu, Refresh option	Synchronizes the selected project or folder with the Architect/Requirements server.
Search	Tools menu, Search option	Activates the Search module.
Set the number of lines per row	View menu, Lines Per Row option	<p>Opens the Lines per row field, where you can enter the number of lines of text to display in the hierarchical content table. This feature is enabled when the Text property column is displayed, though the current setting is always effective.</p> <p>You can enter any number in the field. The default value is 1.</p> <p>In the View menu Lines Per Row option, you can choose 1 Line, 5 Lines, 10 Lines, or 20 Lines.</p>

To control the behavior and display of the toolbar, you can do the following:

- Rearrange button groups within the toolbar area.
- Display button groups in floating windows, independent of the toolbar area.

Rearranging Button Groups Within the Toolbar Area

Each group of buttons can be moved to different positions within the toolbar area.

To rearrange button groups within the toolbar area:

1. At the left of a button group, point to the vertical bar and hold down the left mouse button.
2. Drop the group in the new position within the toolbar area.

For each additional group that you want to move, repeat steps 1 and 2.

Displaying Button Groups in Floating Windows

Each group of buttons can be temporarily detached from the toolbar area and displayed in a separate floating window. These independent windows can be moved to any position on your screen, through the standard Microsoft Windows functions.

Also by standard functions, the windows can be closed to return them to the toolbar area. Floating toolbar windows cannot be resized, minimized, or maximized.

To display button groups in floating windows:

1. At the left of a button group, point to the vertical bar and hold down the left mouse button.
2. Move the pointer out of the toolbar area, and then release the mouse button.

For each additional group that you want to detach, repeat steps 1 and 2.

Module Bar

The module bar contains a button for each module of Architect/Requirements. By clicking a button, you gain access to one of the following modules:

- Search
- Systems Engineering and Requirements Management
- Administration

To control the display of the module bar, you can do the following:

- Disable or enable the module bar.
- Hide or show the module bar.
- Resize the module bar.

Disabling or Enabling the Module Bar

You can disable or enable the module bar at any time. Your last setting in the current session is preserved when you start a new session.



The module bar is enabled by default in the first session after the Architect/Requirements Client with Office Integration installation.

To disable or enable the module:

Pull down the **View** menu and choose **Module Bar**.

Hiding or Showing the Module Bar

You can hide or show the module bar at any time during the current session.

To hide the module bar:

Click the left arrow at the top of the vertical split bar to the right of the module bar.

To show the module bar:

Click the right arrow at the top of the leftmost vertical split bar.

Resizing the Module Bar

You can change the width, but not the height, of the module bar.

To resize the module bar:

1. At the right of the module bar, point to the vertical split bar until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the split bar left or right, and then release the mouse button.

Address Bar

The **Address** bar shows the full path of the project or folder selected in the navigation tree. You cannot directly change the path in the **Address** bar. The path changes automatically with your navigation tree selection.



The **Address** bar does not show the full path of any object selected in the content table or in the notebook pane. That path is shown by the **Full Name** property for the object.

When versioning is enabled for a project, the **Address** bar also contains controls for choosing effectivity rules. An effectivity rule determines which version of a requirement or a building block is displayed in the main window. For more information, see chapter 10, [Working With Versions](#).



If you switch to another project that also has versions enabled and then switch back to the first project, the **Address** bar retains its previous size. If you switch to another project that does not have versions enabled and the **Effectivity** list is not displayed, the **Address** bar does not retain the previous size when you return to the original version-enabled project.

To control the display of the **Address** bar, you can do the following:

- Disable or enable the **Address** bar.
- Hide or show the **Address** bar.
- Resize the **Address** bar.

Disabling or Enabling the Address Bar

You can disable or enable the **Address** bar at any time. Your last setting in the current session is preserved when you start a new session.



- The **Address** bar is enabled by default in the first session after the Architect/Requirements Client with Office Integration installation.
- The controls for choosing effectivity rules are not accessible when the **Address** bar is disabled.

To disable or enable the Address bar:

Pull down the **View** menu and choose **Address Bar**.

Hiding or Showing the Address Bar

You can hide or show the **Address** bar at any time during the current session.

To hide the Address bar:

Click the up arrow on the left of the horizontal split bar below the **Address** bar.

You can simultaneously hide the **Address** bar, the navigation tree, the content table, and the notebook pane by clicking the right arrow at the top of the vertical split bar to the left of the **Address** bar.

To show the Address bar:

Click the down arrow on the left of the horizontal split bar above the navigation tree or the content table.

If the navigation tree or the content table is not displayed, click the down arrow on the left of the topmost horizontal split bar.

Resizing the Address Bar

You can change both the width and the height of the **Address** bar.

To change the width:

1. At the left of the **Address** bar, point to the vertical split bar until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the split bar left or right, and then release the mouse button.

To change the height:

1. At the bottom of the **Address** bar, point to the horizontal split bar until the pointer becomes a vertical double arrow.
2. Hold down the left mouse button, move the split bar up or down, and then release the mouse button.

Navigation Tree

The navigation tree displays all projects to which you have access. You use the navigation tree to open and work with those projects and their folders. Also, you gain access to your Architect/Requirements Recycle Bin through the navigation tree.

Selecting a project, a folder, or the Recycle Bin in the navigation tree displays that object's contents in the content table. Selecting the root node clears the content table. For more information, see [Content Table](#), later in this chapter.

Each project node or folder can have a unique view of property columns in the hierarchical content table. These views let you easily work with the properties. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.

To control the behavior and display of the navigation tree, you can do the following:

- Expand projects and folders.
- Disable or enable the navigation tree.
- Disable or enable the root node.
- Hide or show the navigation tree.
- Switch to the navigation tree.
- Resize the navigation tree.

Expanding a Project or a Folder

The navigation tree displays projects below the root node and folders below the project nodes. These folders may contain other folders, at lower levels in the navigation tree. You expand a project and its folders to view the hierarchy.

To expand a project or a folder:

Click the plus sign (+) to the left of the project or folder.

The plus sign becomes a minus sign (–), and the navigation tree displays the folders at the next lower level.



- To enable plus signs, pull down the **View** menu and choose **Root Node**.
- A plus sign is not displayed if the folder does not contain subfolders. The plus sign is removed if the last subfolder is moved or deleted from the folder.
- If you click a plus sign and the folder does not expand, it contains subfolders to which you do not have **Read** access. If you have questions about folder access, consult your project administrator.

You can also right-click the project or folder and choose **Expand** from the pop-up menu. You can collapse an expanded project or folder by clicking the minus sign, or by right-clicking the project or folder and choosing **Collapse** from the pop-up menu.

Disabling or Enabling the Navigation Tree

You can disable or enable the navigation tree at any time. Your last setting in the current session is preserved when you start a new session.



The navigation tree is enabled by default in the first session after the Architect/Requirements client installation.

To disable or enable the navigation tree, pull down the **View** menu and choose **Navigation Tree**.

Disabling or Enabling the Root Node

The root node occupies the highest level in the navigation tree. Directly below the root node, project nodes are displayed in alphabetical order. You can disable or enable the root node at any time. Your last setting in the current session is preserved when you start a new session.

With the root node enabled, the navigation tree displays plus signs (+) to the left of the projects, and you can click the plus signs to see the folders below the project nodes. With the root node disabled, the navigation tree displays projects without the root node and plus signs, and you have more space in the navigation tree.



The root node is enabled by default in the first session after the Architect/Requirements client installation.

To disable or enable the root node:

Pull down the **View** menu and choose **Root Node**.

Hiding or Showing the Navigation Tree

You can hide or show the navigation tree at any time during the current session.

To hide the navigation tree:

Click the left arrow at the top of the vertical split bar to the right of the tree.

To show the navigation tree:

Click the right arrow at the top of the rightmost vertical split bar.

Switching to the Navigation Tree

If you prefer to use the keyboard instead of the mouse, you can transfer the focus to the navigation tree for keyboard operations.

To switch to the navigation tree:

Pull down the **View** menu and choose the **Go To**→**Navigation Tree** options.

Resizing the Navigation Tree

You can change both the width and the height of the navigation tree.

To change the width:

1. At the right of the navigation tree, point to the vertical split bar until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the split bar left or right, and then release the mouse button.

To change the height:

1. At the top of the navigation tree, point to the horizontal split bar until the pointer becomes a vertical double arrow.
2. Hold down the left mouse button, move the split bar up or down, and then release the mouse button.

Content Table

The content table displays the contents of the object selected in the navigation tree:

- When you select a project node, the content table displays the folders at the project's primary level, directly below the project node in the navigation tree.
- When you select a folder, the content table displays the folders, requirements, building blocks, and groups at the top level of the selected folder.
- When you select your Architect/Requirements Recycle Bin, the content table displays the objects that you deleted from all projects to which you have access.

Depending on your selection in the navigation tree, the content table displays the objects in one of two aspects:

- For a project or a folder, the objects are displayed hierarchically, according to superior and subordinate relationships among the objects.
- For the Recycle Bin, the objects are displayed sequentially, regardless of their levels and relationships in hierarchies from which they were deleted.

Both content table aspects display information in rows and columns, with the following similarities:

- Each row displays one object.
- Each column displays a property name and values for the objects to which the property applies.
-

Special graphics, called *indicators*, symbolize property names and values in some columns.

However, the functions of the hierarchical content table differ from those of the Recycle Bin. The two aspects differ also in the columns that they display.

To control the behavior and display of the content table, you can do the following:

- View objects in a hierarchy.
- Disable or enable the content table.
- Hide or show the content table.
- Switch to the content table.
- Resize the content table.

Hierarchical Content Table

The content table displays the hierarchical view when you select a project or a folder in the navigation tree. For a project, the initial view shows the folders at the primary level, directly below the project node. For a folder, the initial view shows the folders, requirements, building blocks, and groups at the selected folder's top level.

Below either level, the **Name** column shows the hierarchy of superior and subordinate objects. This column is permanently set and contains the following:



A plus sign (+) for each object that has subordinates, or *members*, at the next lower level. You can click a plus sign to see the members directly below the object. For more information, see [Viewing Objects in a Hierarchy](#), later in this chapter.

The **Member Count** property shows the number of members for a selected object. You can view this property in the **Properties** tab or floating window. For more information, see [Properties Tab](#) and [Using Tabs in Floating Windows](#), later in this chapter.



If you click a plus sign and the folder does not expand, it may contain subfolders to which you do not have **Read** access. If you have questions about folder access, consult your project administrator.



An object type indicator in each row. The indicator shows whether that object is a folder, a requirement, a building block, a group, a document (the system-defined folder subtype), or a paragraph (the system-defined requirement subtype). The indicator may show the symbol for a user-defined subtype.

For a requirement, a building block, or a paragraph, the indicator may also show both the symbol for a variant and the symbol for an object that is under static control, or *frozen*. For more information, see chapter 10, [Working With Versions](#).

- An object name in each row, to the right of the object type indicator.

The other columns represent certain system-defined properties that are displayed by default. Each column heading contains a property name, under which the cells contain values for the objects to which the property applies. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

In some default columns, indicators symbolize the property names and values:



The **Attachment Count** property column displays an attachments indicator for each object to which notes or diagrams are attached. You can click an indicator to see the notes and diagrams in the **Attachments** tab or floating window. For more information, see [Attachments Tab](#), later in this chapter.



The **Trace Link Count** property column displays a links indicator for each object that is connected to other objects by a trace link. You can click an indicator to see the defining and complying objects in the **Links** tab or floating window. For more information, see [Links Tab](#), later in this chapter.

-

If the change management package is enabled for the project, the **Change Log** property column displays a links indicator for each object to which a change log is attached. You can click an indicator to see the change log in the **Attachments** tab or floating window. For more information, see [Attachments Tab](#), later in this chapter, and chapter 12, [Using the Change Management Package](#).

You can remove the default columns, and you can add columns for other properties. For example, you can add the **Version Count** property column to see a versions indicator for each object with versions or variants. When you click one of these indicators, that object's versions and variants are displayed in the **Versions** tab or floating window. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*, and [Versions Tab](#), later in this chapter.

In the hierarchical content table, you can also do the following:

- Add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), later in this chapter.
- Save the current view of column settings, and recall a saved view to display those column settings at any time. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.
- Export the current view to Microsoft Excel. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- Change editable property values for the objects. For more information, see [Editing the Properties of a Selected Object](#), [Editing Properties in Table View Cells](#), and [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

Figure 3-3 shows the hierarchical content table.

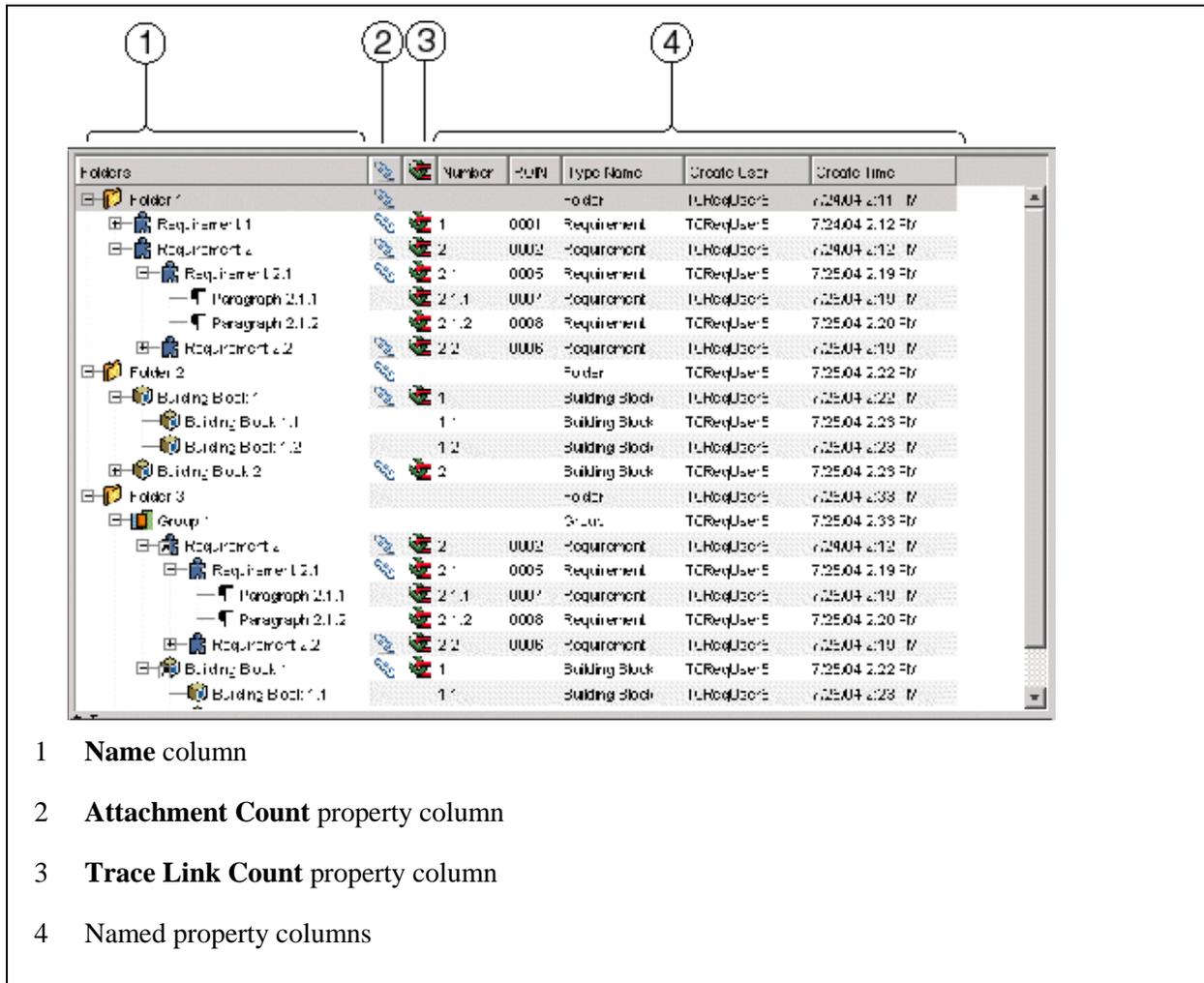


Figure 3-3. Hierarchical Content Table

Architect/Requirements Recycle Bin

When you select your Architect/Requirements Recycle Bin in the navigation tree, the content table displays the objects that you deleted from all projects to which you have access. This view displays the objects in the sequence of the current sort column, regardless of any relationships as parents, children, or siblings.

The columns in the Recycle Bin are permanently set and are displayed as follows:

- The leftmost column displays an object type indicator in each row, showing whether that object is a folder, requirement, building block, note, trace link, or group. The indicator may also show the symbol for a user-defined subtype and the symbol for a variant of a requirement or a building block.
- The attachments indicator column displays an indicator for each object to which one or more notes or diagrams are attached. You can click an indicator to see the notes and diagrams in the **Attachments** tab or floating window. For more information, see [Attachments Tab](#) and [Using Tabs in Floating Windows](#), later in this chapter; chapter 6, [Constructing System Views With Building Blocks and Diagrams](#); and chapter 8, [Recording Supplementary Information With Notes](#).
- The property columns display system-defined properties. Each column heading contains a property name, under which the cells contain values for the objects to which the property applies. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

You can restore objects to the locations from which they were deleted. You can empty your Recycle Bin to remove the objects permanently from the database. For more information, see [Restoring Objects](#) and [Emptying Your Architect/Requirements Recycle Bin](#) in chapter 4, *Maintaining a Project*.



Your Recycle Bin can be emptied automatically after a number of days specified by your Architect/Requirements enterprise administrator. Also, your enterprise administrator can empty your Recycle Bin manually, for example, to manage storage space in the database. If you have questions about emptying your Recycle Bin, consult your enterprise administrator.

You cannot edit properties in the Recycle Bin. However, you can remove the default columns, and you can add columns for other system-defined properties. Also, you can rearrange and resize columns and sort by any column, and you can export the current view to Microsoft Excel. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*, and [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.

Figure 3-4 shows the Architect/Requirements Recycle Bin.

	Name ▲	RCIN	Original Location	Date Deleted	Type Name
				7/25/04 6:03 PM	Link
				7/25/04 6:06 PM	Link
	Building Block 4 -> Building Block 4.1			7/25/04 5:57 PM	Trace Link
	Building Block 4 -> Building Block 4.2			7/25/04 5:57 PM	Trace Link
	Experimental			7/25/04 6:04 PM	Diagram
	Folder 4		WExample Project	7/25/04 6:08 PM	Folder
	Group 4			7/25/04 6:04 PM	Group
	Note Folder 4			7/25/04 6:04 PM	Note
	Requirement 3	0011	WExample Project\Folder 1	7/25/04 2:39 PM	Requirement
	Requirement 1 -> Requirement 1.1			7/25/04 5:59 PM	Trace Link
	Requirement 4 -> Requirement 4.2			7/25/04 5:59 PM	Trace Link

13 object(s), 1 selected

- 1 Object type indicator column
- 2 Attachments indicator column
- 3 Property columns

Figure 3-4. Architect/Requirements Recycle Bin

Disabling or Enabling the Content Table

You can disable or enable the content table at any time. Your last setting in the current session is preserved when you start a new session.



The content table is enabled by default in the first session after the Architect/Requirements client installation.

To disable or enable the content table:

Pull down the **View** menu and choose **Content Table**.

Hiding or Showing the Content Table

You can hide or show the content table at any time during the current session.

To hide the content table:

Click the up arrow on the left of the horizontal split bar below the content table.

If the split bar does not display arrows, pull down the **View** menu and choose **Notebook Pane**.

You can hide the content table and the notebook pane simultaneously by clicking the right arrow at the top of the vertical split bar to the left.

To show the content table:

Do one of the following:

- If only the content table is hidden, click the down arrow on the left of the horizontal split bar above the notebook pane.
- If both the content table and the notebook pane are hidden, click the left arrow at the top of the rightmost vertical split bar.

If the content table is still hidden, click the down arrow on the left of the horizontal split bar above the notebook pane. If the split bar does not display arrows, pull down the **View** menu and choose **Notebook Pane**.

Switching to the Content Table

If you prefer to use the keyboard instead of the mouse, you can transfer the focus to the content table for keyboard operations.

To switch to the content table:

Pull down the **View** menu and choose the **Go To**→**Content Table** options.

Resizing the Content Table

You can change both the width and the height of the content table.

To change the width:

1. At the left of the content table, point to the vertical split bar until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the split bar left or right, and then release the mouse button.

To change the height:

1. At the bottom of the content table, point to the horizontal split bar until the pointer becomes a vertical double arrow.
2. Hold down the left mouse button, move the split bar up or down, and then release the mouse button.

Notebook Pane

The notebook pane displays information for the object selected in the navigation tree or the content table. You work with that information through the following:

- The **Properties** tab displays the properties of the object selected in the navigation tree, the hierarchical content table, or your Architect/Requirements Recycle Bin. For more information, see [Properties Tab](#), later in this chapter.
- The **Attachments** tab displays notes, diagrams, spreadsheets, and change objects for the object selected in the navigation tree, content table, or Recycle Bin. For more information, see [Attachments Tab](#), later in this chapter.
- The **Links** tab displays each object that is connected by a trace link to the object selected in navigation tree or the hierarchical content table. For more information, see [Links Tab](#), later in this chapter.
- The **Connectivity** tab displays the ports and connections for the building block selected in the hierarchical content table. For more information, see [Connectivity Tab](#), later in this chapter.
- The **Preview** tab displays the content of the requirement selected in the hierarchical content table. For more information, see [Preview Tab](#), later in this chapter.
- The **Where Used** tab displays all groups, diagrams, and remote proxies that reference the object selected in the hierarchical content table. For more information, see [Where Used Tab](#), later in this chapter.
- The **Versions** tab displays all versions and variants of the object selected in the content table, or your Recycle Bin if versions are enabled for the project. For more information, see [Versions Tab](#), later in this chapter.

You can click the tabs to change from one to another, and work in the tab that is currently on top. In addition, you can open each tab in a separate floating window to display the selected object's information in multiple views. For more information, see [Using Tabs in Floating Windows](#), later in this chapter.

Floating tab windows also can display information for objects selected in the notebook pane:

- The **Properties** window displays the properties of the object selected in the **Attachments**, **Links**, **Connectivity**, **Where Used**, or **Versions** tab.
- The **Attachments** window displays notes, diagrams, spreadsheets, and change objects for the object selected in the **Links**, **Connectivity**, **Where Used**, or **Versions** tab.
- The **Links** window displays the trace links for the object selected in the **Connectivity**, **Where Used**, or **Versions** tab.
- The **Connectivity** tab displays the ports and connections for the building block selected in the **Links**, **Where Used**, or **Versions** tab.
- The **Preview** window displays the content of the note selected in the **Attachments** tab, or the content of the requirement selected in the **Links** tab.
- The **Where Used** window displays the groups, diagrams, and remote proxies for the object selected in the **Attachments**, **Links**, or **Connectivity** tab.
- The **Versions** window displays the versions and variants of the object selected in the **Links** or **Connectivity** tab.



Notebook pane tabs and windows show information for the most recently selected object.

To control the behavior and display of the notebook pane, you can do the following:

- Use tabs in floating windows.
- Disable or enable the notebook pane.
- Hide or show the notebook pane.
- Switch to the notebook pane.
- Resize the notebook pane.

Properties Tab

The **Properties** tab and floating window display all viewable properties that apply to the object selected in the navigation tree, the hierarchical content table, or your Architect/Requirements Recycle Bin. The **Properties** window displays the properties of the object selected in the **Attachments**, **Links**, **Connectivity**, **Where Used**, or **Versions** tab. For more information, see [Hierarchical Content Table](#) or [Architect/Requirements Recycle Bin](#), earlier in this chapter; and [Attachments Tab](#), [Links Tab](#), [Connectivity Tab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#), later in this chapter.

This view contains a table that displays the properties sequentially. Each row displays one property, and the columns display the property names and values. You cannot add, remove, or rearrange columns in this table. However, you can resize both columns and sort by either column. For more information, see [Resizing Columns](#) and [Setting the Sort Column](#), later in this chapter.

Also, you can export the current view to Microsoft Excel, and you can change editable property values directly in the **Value** column. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*; [Editing the Properties of a Selected Object](#) in chapter 9, *Working With Object Properties*; and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

The **Properties** tab and window also contain controls that filter the properties displayed in the table. You can use these controls to view properties according to certain categories. For more information, see [Filtering Property Tables](#) in chapter 9, *Working With Object Properties*.

Figure 3-5 shows the **Properties** tab in the notebook pane.

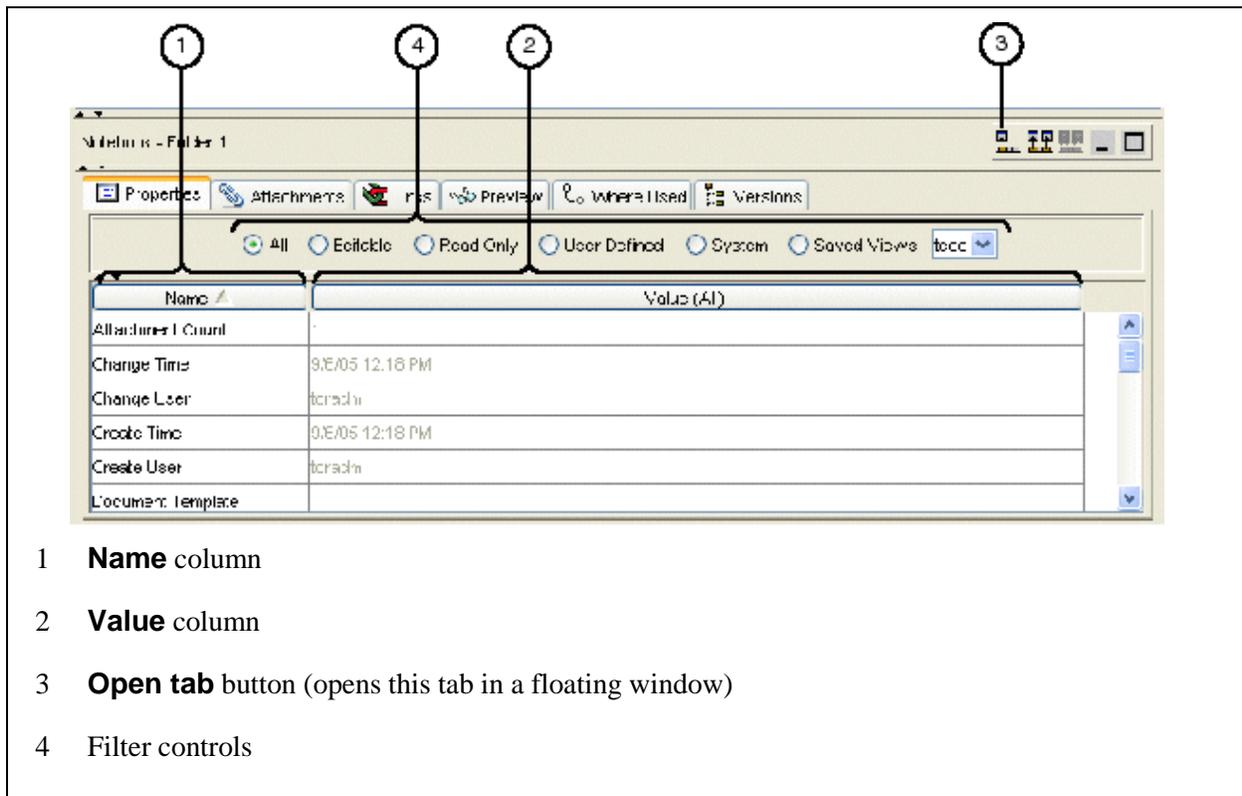


Figure 3-5. Properties Tab in Notebook Pane

Attachments Tab

The **Attachments** tab and floating window display the notes, diagrams, spreadsheets, change approval objects, and change logs attached to the object selected in the navigation tree, the hierarchical content table, or your Architect/Requirements Recycle Bin. The **Attachments** window displays the notes, diagrams, spreadsheets, change approval objects, and change logs for the object selected in the **Links**, **Connectivity**, **Where Used**, or **Versions** tab. For more information, see [Hierarchical Content Table](#) or [Architect/Requirements Recycle Bin](#), earlier in this chapter; and [Links Tab](#), [Connectivity Tab](#), [Where Used Tab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#), later in this chapter.

This view contains a table that displays the objects hierarchically. The leftmost column, **Attachment**, contains the following:

- A plus sign (+) for each object, including notes, to which at least one note is attached. You can click the plus sign to see the notes at the next lower level. In turn, plus signs may be displayed for lower level notes to which other notes are attached.
- An object type indicator in each row. The indicator shows whether that object is a note, diagram, spreadsheet, a change approval object, or change log. The indicator may also show the symbol for a user-defined subtype.
- An object name in each row, to the right of the object type indicator.

The other columns display the default system-defined properties of the objects. In these columns, a heading contains the property name, under which the cells contain the values. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

You can also do the following:

- Attach notes to objects in the **Attachments** tab and window, including other notes. For more information, see [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Edit and view note content. For more information, see [Editing a Note](#) or [Viewing Note Content](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Copy and move notes. For more information, see [Copying Objects](#) and [Moving Objects](#) in chapter 9, *Maintaining a Project*.
- Copy object URLs. For more information, see [Copying Object URLs](#) in chapter 9, *Maintaining a Project*.
- Edit and view diagrams. For more information, see [Editing a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Rename and delete objects. For more information, see [Renaming an Object](#) and [Deleting Objects](#) in chapter 4, *Maintaining a Project*.



When you delete a parent note, you also delete its child notes at all lower levels.

- Create reference links to notes, change logs, change approval objects, spreadsheets, and diagram images. For more information, see [Storing Property Values Through Reference Links](#) in chapter 9, *Working With Object Properties*, and [Creating a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Export objects to Microsoft Excel. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- View and edit the properties of the objects, in this view or in the **Properties** floating window. For more information, see [Properties Tab](#), earlier in this chapter, and chapter 9, [Working With Object Properties](#).

Figure 3-6 shows the **Attachments** tab in the notebook pane.

Attachment	Text	Create User	Create Time
Component 1		tcradm	3/26/2007 1:39 ...
Diagram Image Note		tcradm	3/26/2007 1:42 ...
Design Note		tcradm	3/26/2007 1:48 ...
Testing Comments	Preliminary design testing con...	tcradm	3/26/2007 1:54 ...
Note	Some testing comments on th...	tcradm	3/26/2007 1:51 ...
SpreadSheet		tcradm	3/26/2007 1:43 ...
Functions	Excel functions include the fol...	tcradm	3/26/2007 1:45 ...

- 1 **Attachment** column
- 2 Property columns
- 3 **Open tab** button (opens this tab in a floating window)

Figure 3-6. Attachments Tab in Notebook Pane

Links Tab

The **Links** tab and floating window consist of the **Trace** and **Relations** subtabs. Each subtab displays specific information for the object selected in the navigation tree or the hierarchical content table.

The **Trace** subtab displays each object that defines and complies with the selected object. Defining objects may reside within the same Architect/Requirements project or in a different project. Complying objects may reside within any project and in other Teamcenter products.

The **Relations** subtab displays each trace link and connection for the selected object. In addition, the subtab displays the starting or ending object for each trace link and connection.

When the **Links** window is open, each subtab can display information for the object selected in the **Connectivity**, **Where Used**, and **Versions** tab. For more information, see [Hierarchical Content Table](#), earlier in this chapter; and [Connectivity Tab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#), later in this chapter.

The following sections describe the **Trace** and **Relations** subtabs.

Trace Subtab

The **Trace** subtab consists of two panes, one for defining objects and one for complying objects. Each pane contains a table in which the leftmost column displays the linked objects in a hierarchy, according to their sequence in the defining or complying path:

- The **Defining Trace** column shows the defining objects that directly precede the selected object in the defining path. A plus sign (+) is shown for each defining object that complies with other objects, continuing the defining path upstream.
- The **Complying Trace** column shows the complying objects that directly follow the selected object in the complying path. A plus sign (+) is shown for each complying object that defines other objects, continuing the complying path downstream.

You can click a plus sign to see the next objects in a continuing path. For more information, see [Viewing Object Relationships](#) in chapter 7, *Showing Object Relationships With Trace Links*.

The hierarchical columns are permanently set, and each contains the following:

- An object type indicator in each row. The indicator shows whether that object is a folder, a requirement, a building block, a group, a document (the system-defined folder subtype), or a paragraph (the system-defined requirement subtype). The indicator may show the symbol for a user-defined subtype.

For a requirement, a building block, or a paragraph, the indicator may also show both the symbol for a variant and the symbol for an object that is under static control, or *frozen*. For more information, see chapter 10, [Working With Versions](#).
- An object name in each row, to the right of the object type indicator.

The other columns represent certain system-defined properties that are displayed by default. Each column heading contains a property name, under which the cells contain values for the objects to which the property applies. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

In the **Trace** subtab, you can also do the following:

- Rename and delete linked objects, and export the current view to Microsoft Excel. For more information, see [Renaming an Object](#), [Deleting Objects](#), and [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- Copy and move linked objects, through options on the **Edit** menu or by dropping selected objects on the destination. For more information, see [Copying Objects](#) and [Moving Objects](#) in chapter 4, *Maintaining a Project*.
- Open linked requirements to edit or view their content, and export linked requirements to Microsoft Word. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*, and [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Create and delete trace links, and navigate to linked objects. For more information, see [Creating Trace Links](#), [Linking to an Object in Another Teamcenter Product](#), [Deleting Trace Links for an Object](#), and [Navigating to a Linked Object](#) in chapter 7, *Showing Object Relationships With Trace Links*.
- Attach notes to defining and complying objects. For more information, see [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Edit the properties of linked objects, in this view or in the **Properties** floating window. For more information, see [Properties Tab](#), earlier in this chapter; and [Editing the Properties of a Selected Object](#), [Editing Properties in Table View Cells](#), and [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

Figure 3-7 shows the **Trace** subtab of the **Links** tab in the notebook pane.

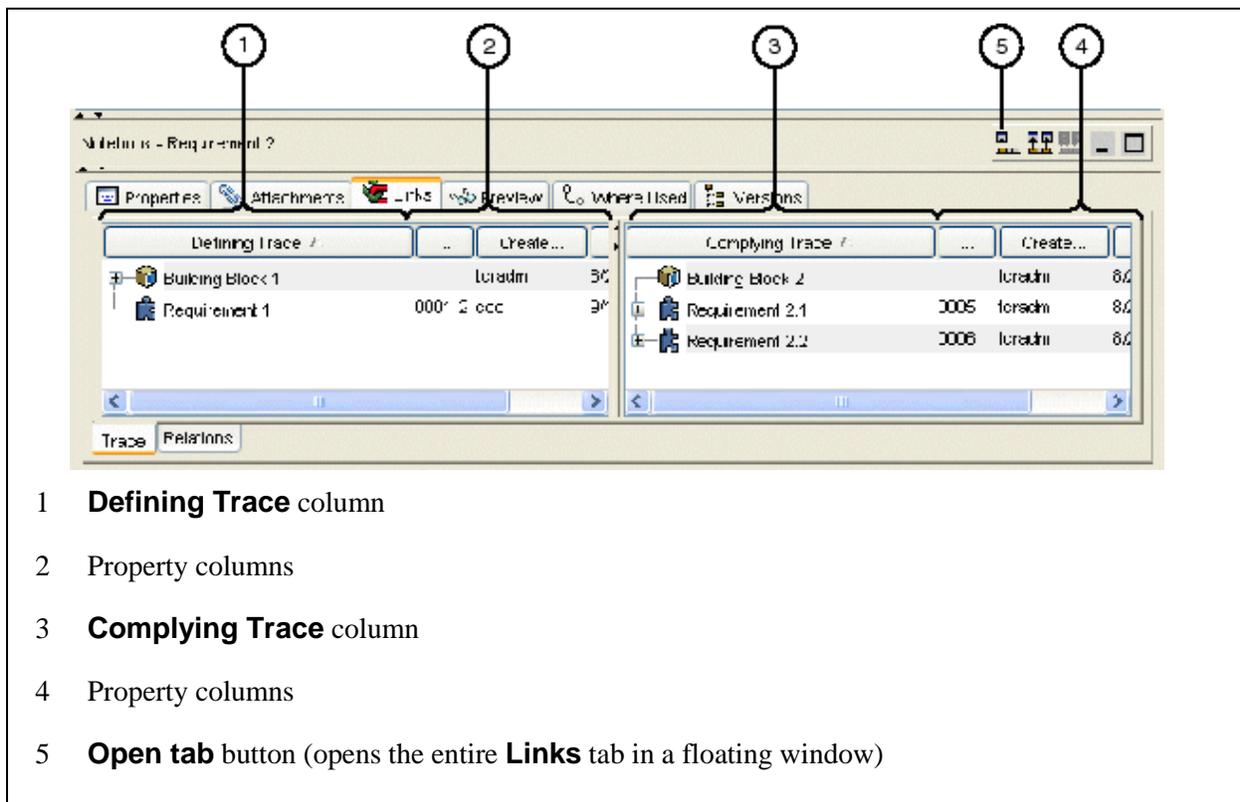


Figure 3-7. Trace Subtab of Links Tab in Notebook Pane

Relations Subtab

The **Relations** subtab displays trace links and connections as independent objects that you can view and modify. The subtab also displays the starting or ending object for each trace link and connection.

The **Relations** subtab consists of the **Start** pane and the **End** pane. Each pane contains two sequential tables, an *object table* at the left and a *link table* at the right:

- In the **Start** pane, the object table shows each object from which a trace link or a connection originates to the selected object. The actual trace link or connection is shown in the corresponding link table in the row that matches the starting object position in the object table.
- In the **End** pane, the object table shows each object at which a trace link or a connection originating from the selected object is completed. The actual trace link or connection is shown in the corresponding link table in the row that matches the ending object position in the object table.



In each pane, the rows in the object table and the link table remain synchronized at all times. For example, when you sort or scroll a link table, the corresponding object table is automatically sorted or scrolled to match the new view in the link table.

In each table in the **Relations** subtab, the leftmost column displays an object type indicator for each folder, document (the system-defined folder subtype), requirement, paragraph (the system-defined requirement subtype), building block, group, trace link, or connection. The indicator may show the symbol for a user-defined subtype.



For a requirement, a paragraph, or a building block, the indicator may also show both the symbol for a variant and the symbol for an object that is under static control, or *frozen*. For more information, see chapter 10, [Working With Versions](#).

The other columns display default system-defined properties for the objects in the table. In these columns, a heading contains the property name, under which the cells contain the values. You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#); and [Adding and Removing Columns, Rearranging Columns, Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

In the **Relations** subtab, you can also do the following:

- Rename and delete linked objects, trace links, and connections. For more information, see [Renaming an Object](#) and [Deleting Objects](#) in chapter 4, *Maintaining a Project*.
- Export the trace links, connections, or starting and ending objects in a given table to Microsoft Excel or Microsoft Word. For more information, see [Exporting Objects to Microsoft Office Excel](#) and [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.
- Open starting and ending requirements to edit or view their content. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Create and delete trace links and connections, and navigate to starting and ending objects. For more information, see [Creating Trace Links](#), [Deleting Trace Links for an Object](#), and [Navigating to a Linked Object](#) in chapter 7, *Showing Object Relationships With Trace Links*.
- Attach notes to trace links and connections. For more information, see [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.

- Edit the properties of trace links, connections, and their starting or ending objects. You can edit properties in this view or in the **Properties** floating window. For more information, see [Properties Tab](#), earlier in this chapter; and [Editing the Properties of a Selected Object](#), [Editing Properties in Table View Cells](#), and [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

Figure 3-8 shows the **Relations** subtab of the **Links** tab in the notebook pane.

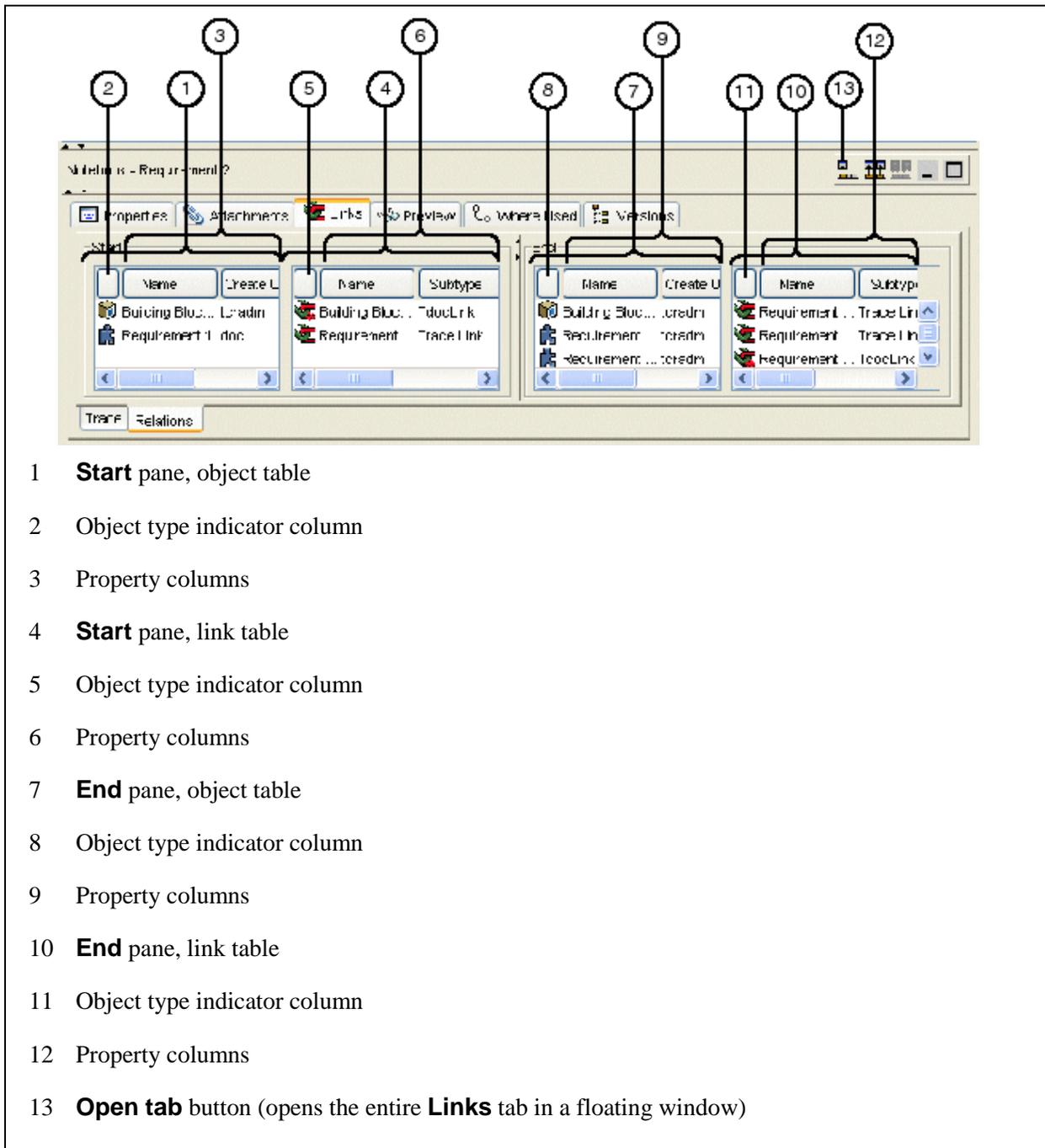


Figure 3-8. Relations Subtab of Links Tab in Notebook Pane

Connectivity Tab

The **Connectivity** tab and floating window display the ports and connections for the building block selected in the hierarchical content table. The **Connectivity** window displays the ports and connections for the building block selected in the **Links, Where Used**, or **Versions** tab.

This view contains a table that displays the objects hierarchically. The **Name** column contains the following:

- A plus sign (+) for each port that has at least one connection. You can click the port's plus sign to see the connections at the next lower level.
Below each connection, the port at the other end of the connection is displayed when you click the connection's plus sign. This hierarchy may extend to multiple levels below any given port.
- An object type indicator in each row. The indicator shows whether that object is a port or a connection. The indicator may also show the symbol for a user-defined subtype.
- An object name in each row, to the right of the object type indicator.

The other columns display the default system-defined properties of the ports and connections. In these columns, a heading contains the property name, under which the cells contain the values. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

You can also do the following:

- Attach notes to ports and connections by using the **Attachments** window. For more information, see [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Copy and move ports and connections. For more information, see [Copying Objects](#) and [Moving Objects](#) in chapter 9, *Maintaining a Project*.
- Copy the URLs of ports and connections. For more information, see [Copying Object URLs](#) in chapter 9, *Maintaining a Project*.
- Rename and delete ports and connections. For more information, see [Renaming an Object](#) and [Deleting Objects](#) in chapter 4, *Maintaining a Project*.

- Create reference links to ports and connections. For more information, see [Storing Property Values Through Reference Links](#) in chapter 9, *Working With Object Properties*, and [Creating a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Export ports and connections to Microsoft Excel. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- View and edit the properties of the ports and connections, in this view or in the **Properties** floating window. For more information, see [Properties Tab](#), earlier in this chapter, and chapter 9, *Working With Object Properties*.

Figure 3-9 shows the **Connectivity** tab in the notebook pane.

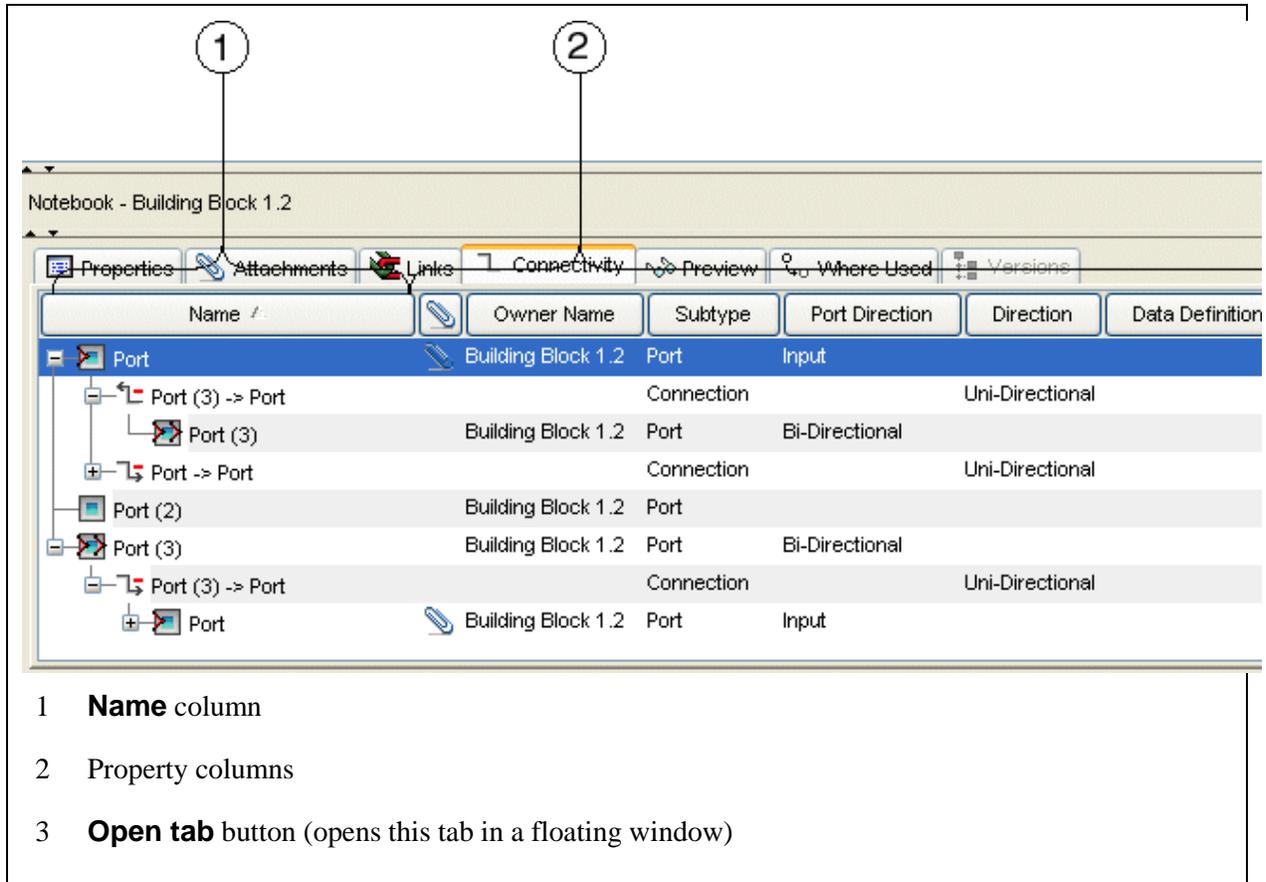


Figure 3-9. Connectivity Tab in Notebook Pane

Preview Tab

The **Preview** tab and floating window display the content of the requirement selected in the hierarchical content table. For more information, see [Hierarchical Content Table](#), earlier in this chapter.

The **Preview** window displays the content of the following:

- The note or diagram selected in the **Attachments** tab. For more information, see [Attachments Tab](#), earlier in this chapter.
- The requirement selected in the **Links** or **Versions** tab. For more information, see [Links Tab](#), earlier in this chapter, or [Versions Tab](#) and [Using Tabs in Floating Windows](#), later in this chapter.

You can see the entire content without opening the requirement, note, or diagram. You can print the content by right-clicking in content area and choosing **Print** from the pop-up menu. However, you cannot change the content from this view. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) in chapter 5, *Managing Requirements*; [Editing a Note](#) in chapter 8, *Recording Supplementary Information With Notes*; or [Editing a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.



A multiple-line tooltip that you display in the content table may overlap the **Preview** tab, disabling the tab temporarily. The tab is enabled when the tooltip is removed.

Figure 3-10 shows the **Preview** tab in the notebook pane.

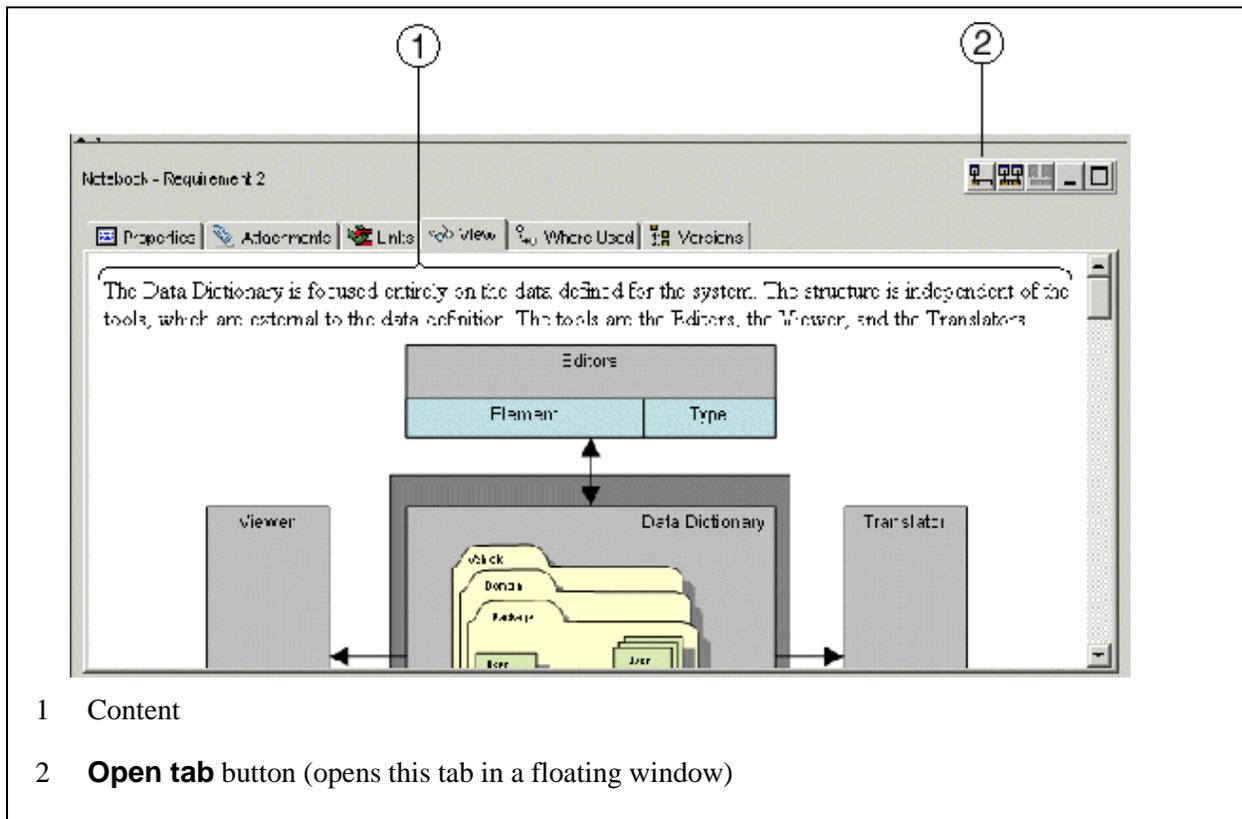


Figure 3-10. Preview Tab in Notebook Pane

Where Used Tab

The **Where Used** tab and floating window display the groups, diagrams, and remote proxies that reference the object selected in the hierarchical content table. The **Where Used** window displays such references for the object selected in the **Attachments**, **Links**, or **Connectivity** tab. For more information, see [Hierarchical Content Table](#), [Attachments Tab](#), [Links Tab](#), or [Connectivity Tab](#), [Where Used Tab](#), earlier in this chapter, or [Using Tabs in Floating Windows](#), later in this chapter.

This view contains a table that displays the groups, diagrams, and proxies sequentially. The leftmost column is permanently set and contains an object type indicator for each group, diagram, and proxy. The other columns display the default system-defined properties of those objects. In these columns, a heading contains the property name, under which the cells contain the values. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

For the selected object in the hierarchical content table or the **Attachments** or **Links** tab, you can remove the object's reference from one or more groups displayed in the **Where Used** tab. For more information, see [Removing Group Members](#) in chapter 4, *Maintaining a Project*.

You can also do the following:

- Rename and delete groups, diagrams, and proxies, and export the table to Microsoft Excel. For more information, see [Renaming an Object](#), [Deleting Objects](#), and [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- Open diagrams for editing. For more information, see [Editing a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- View and edit properties for the objects, in this view or in the **Properties** window. For more information, see [Properties Tab](#), earlier in this chapter, and chapter 9, *Working With Object Properties*.

Figure 3-11 shows the **Where Used** tab in the notebook pane.

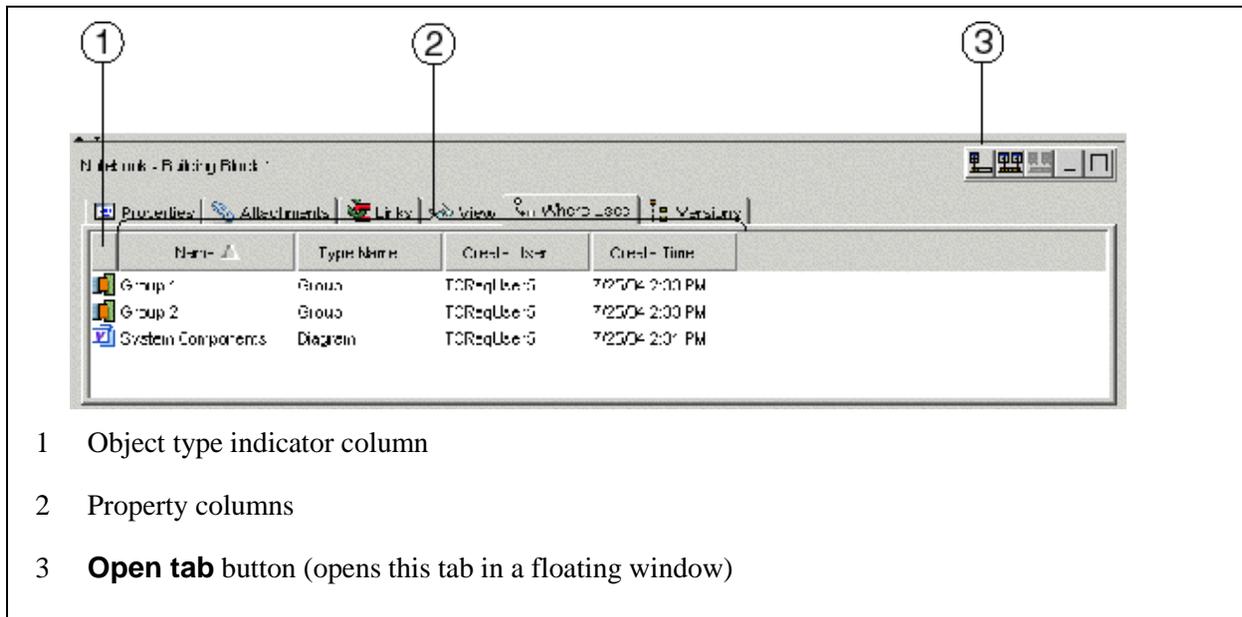


Figure 3-11. Where Used Tab in Notebook Pane

Versions Tab

The **Versions** tab and floating window display all versions and variants of the requirement or building block selected in the hierarchical content table, or your Architect/Requirements Recycle Bin if versions are enabled for the project. For more information, see [Enabling Versions for a Project](#) in chapter 10, *Working With Versions*.

The **Versions** window displays the versions and variants of the requirement or building block selected in the **Links** or **Connectivity** tab. For more information, see [Hierarchical Content Table](#), [Links Tab](#), and [Connectivity Tab](#), earlier in this chapter, and [Using Tabs in Floating Windows](#), later in this chapter.



The **Versions** tab is not available if versioning is disabled for the project.

This view contains a table that displays the versions and variants in a hierarchy, from the earliest to the latest. The hierarchical column, **Name**, is permanently set and contains the following:

- A plus sign (+) for each object that has later versions or variants. You can click a plus sign to see the later ones in separate rows below the earlier one. For more information, see [Viewing a Version Tree](#) in chapter 10, *Working With Versions*.
- An object type indicator in each row, showing whether the object is a requirement, a building block, or a paragraph (the system-defined requirement subtype). The indicator may also show the symbol for a user-defined subtype, the symbol for a variant, and the symbol for an object that is under static control, or *frozen*.
- An object name in each row, to the right of the object type indicator.

The other columns represent certain system-defined properties that are displayed by default. Each column heading contains a property name, under which the cells contain values for the objects to which the property applies. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can remove the default property columns, and you can add columns for more system-defined properties. Also, you can add and remove columns for user-defined properties, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.

You can also do the following:

- Rename versions and variants, and export the current view to Microsoft Excel. For more information, see [Renaming an Object](#) and [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.
- Open requirement versions and variants in Microsoft Word to edit or view their content. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) or [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Create defining and complying trace links, within the project, for versions and variants. For more information, see [Creating Trace Links](#) in chapter 7, *Showing Object Relationships With Trace Links*.

- Edit the properties of versions and variants, in this view or in the **Properties** floating window. For more information, see [Properties Tab](#), earlier in this chapter; and [Editing the Properties of a Selected Object](#) and [Editing Properties in Table View Cells](#) in chapter 9, *Working With Object Properties*.

Figure 3-12 shows the **Versions** tab in the notebook pane.

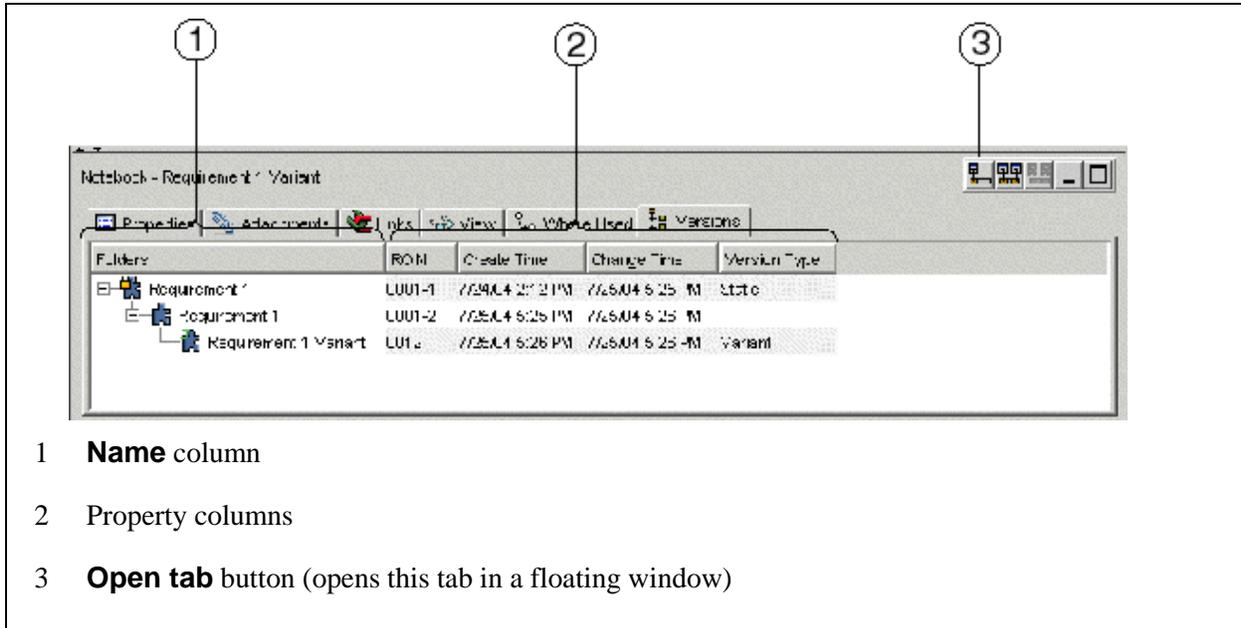


Figure 3-12. Versions Tab in Notebook Pane

Using Tabs in Floating Windows

For each tab in the notebook pane, you can temporarily transfer the information from the tab to a floating window. By opening a tab window, you can keep that information in view while you work on another tab directly in the notebook pane. Conversely, you can view a fixed tab in the notebook pane while working in a floating window.

Using the notebook pane's toolbar, you can open the tab that is currently on top, or you can open all tabs simultaneously, each tab in a separate window. Also with the toolbar, you can close all open tab windows in one action.

You close one window at a time through the standard Microsoft Windows functions. Also by standard functions, you can resize, minimize, and maximize the windows, and move them to any position on your screen.

In the same way as you work on the corresponding tab, you can use each floating window to work with information for the object selected in the content table. Moreover, tab windows extend your work to objects selected in the notebook pane:

- In the **Properties** window, you can work with the properties that apply to the object selected in the **Attachments**, **Links**, **Connectivity**, **Where Used**, or **Versions** tab. For more information, see [Properties Tab](#), earlier in this chapter.
- In the **Attachments** window, you can work with the notes, diagrams, spreadsheets, or change approval objects attached to the object selected in the **Links**, **Connectivity**, **Where Used**, or **Versions** tab. For more information, see [Attachments Tab](#), earlier in this chapter.
- In the **Links** window, you can work with the trace links and linked objects for the object selected in the **Connectivity**, **Where Used**, or **Versions** tab. For more information, see [Links Tab](#), earlier in this chapter.
- In the **Connectivity** window, you can work with the ports and connections for the building block selected in the **Links**, **Where Used**, or **Versions** tab. For more information, see [Connectivity Tab](#), earlier in this chapter.
- In the **Preview** window, you can see the content of the note or diagram selected in the **Attachments** tab and of the requirement selected in the **Links** tab. For more information, see [Preview Tab](#), earlier in this chapter.
- In the **Where Used** window, you can work with the groups, diagrams, and remote proxies for the object selected in the **Attachments**, **Links**, or **Connectivity** tab. For more information, see [Where Used Tab](#), earlier in this chapter.
- In the **Versions** window, you can work with the versions and variants of the requirement or building block selected in the **Links** tab. For more information, see [Versions Tab](#), earlier in this chapter.

Your column settings in the **Attachments**, **Links**, **Connectivity**, **Where Used**, and **Versions** tabs are maintained in the corresponding windows. Also, columns that you add and remove in the floating windows are maintained in the tabs when you close the windows. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.

To enhance navigation among the views, you can add columns in which indicators symbolize the names and values of certain system-defined properties:

-

The **Attachment Count** property column contains an attachments indicator for each object to which one or more notes, diagrams, spreadsheets, or change approval objects are attached. You can click an indicator to see that object's attachments in the **Attachments** tab.

-

The **Trace Link Count** property column contains a links indicator for each object to which one or more objects are connected by a trace link. You can click an indicator to see that object's trace links in the **Links** tab.

-

The **Version Count** property column contains a versions indicator for each object that has one or more versions or variants. You can click an indicator to see that object's versions and variants in the **Versions** tab.

To open floating tab windows:

On the notebook pane's toolbar, do one of the following:

- To open the top tab, click the **Open tab** button.

The new window opens as the active window. The next unopened tab in the display order becomes the top tab. For example, if you open the **Links** window, that tab is dimmed in the notebook pane, and the **Preview** tab is on top.

- To open all tabs simultaneously, click the **Open all** button.

The notebook pane reduces to minimum height, and a new window opens for each tab. The windows are cascaded, with the **Preview** window active.

To close floating tab windows:

Do one of the following:

- To close all windows, click the **Close all** button on the notebook pane's toolbar.

The windows return to their fixed tab positions. The top tab depends on which windows were open, in conjunction with the tab display order from left to right. For example, if only the **Properties** and **Links** windows were open, the **Links** tab, the rightmost of the two in the display order, becomes the top tab.

- To close a single window, click the **Close** box in the upper right corner.

The window returns to its fixed tab position and becomes the top tab.

Figure 3-13 shows the toolbar in the notebook pane.

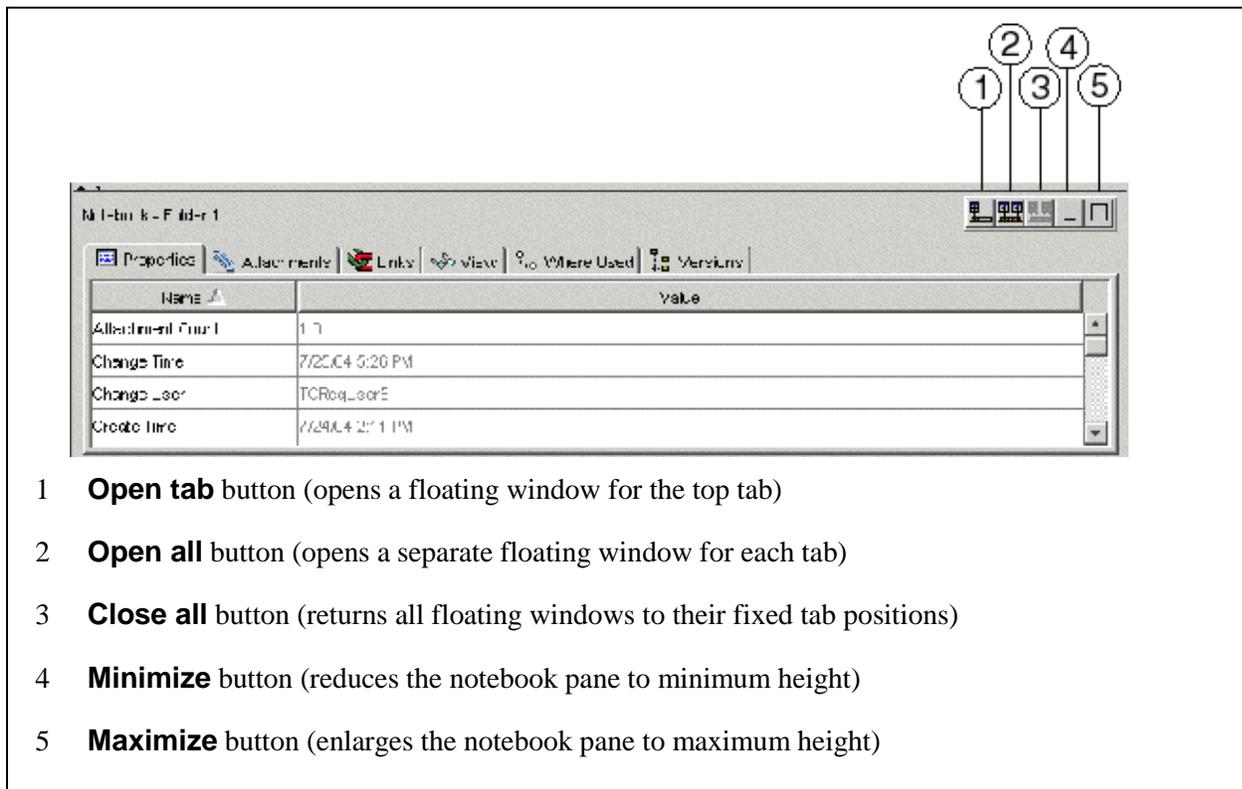


Figure 3-13. Toolbar in Notebook Pane

Disabling or Enabling the Notebook Pane

You can disable or enable the notebook pane at any time. Your last setting in the current session is preserved when you start a new session.



The notebook pane is enabled by default in the first session after the Architect/Requirements client installation.

To disable or enable the notebook pane:

Pull down the **View** menu and choose **Notebook Pane**.

Hiding or Showing the Notebook Pane

You can hide or show the notebook pane at any time during the current session.

To hide the notebook pane:

Click the down arrow on the left of the horizontal split bar above the notebook pane.

You can hide the notebook pane and the content table simultaneously by clicking the right arrow at the top of the vertical split bar to the left.



In the **Links** tab, you can hide a pane by clicking the left arrow or the right arrow on the vertical split bar between the panes.

To show the notebook pane:

Do one of the following:

- If only the notebook pane is hidden, click the up arrow on the left of the horizontal split bar below the content table.
- If both the notebook pane and the content table are hidden, click the left arrow at the top of the rightmost vertical split bar.

If the notebook pane is still hidden, click the up arrow on the left of the horizontal split bar below the content table.



In the **Links** tab, either pane may be hidden, with the tab displaying a vertical split bar to the left or the right of the pane that is shown. To show the **Defining Trace** pane, click the right arrow on the split bar. To show the **Complying Trace** pane, click the left arrow on the split bar.

Switching to the Notebook Pane

If you prefer to use the keyboard instead of the mouse, you can transfer the focus to the notebook pane for keyboard operations.

To switch to the notebook pane:

Pull down the **View** menu and choose the **Go To→Notebook Pane** options.

Resizing the Notebook Pane

You can manually change both the width and the height of the notebook pane. In addition, you can automatically reduce the notebook pane to its minimum height, enlarge it to its maximum height, and restore it to its default height.

When the notebook pane is reduced, the tabs are not visible, and the content table extends toward the bottom of the main window. When the notebook pane is enlarged, it extends toward the top of the main window, and the content table is not visible.

To manually change the width:

1. At the left of the notebook pane, point to the vertical split bar until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the split bar left or right, and then release the mouse button.



In the **Links** tab, you can change the width of both panes by moving the vertical split bar between the panes.

To manually change the height:

1. At the top of the notebook pane, point to the horizontal split bar until the pointer becomes a vertical double arrow.
2. Hold down the left mouse button, move the split bar up or down, and then release the mouse button.

To automatically change the height:

On the notebook pane's toolbar (figure [Toolbar in Notebook Pane](#)), do one of the following:

- To reduce the notebook pane to its minimum height, click the **Minimize** button.

At the top of the notebook pane, its heading and toolbar remain visible below the horizontal split bar. On the toolbar, the **Minimize** button becomes the **Restore** button, and the **Maximize** button becomes unavailable.



While the notebook pane is reduced, you can use the other buttons to open the tabs in separate windows. For more information, see [Using Tabs in Floating Windows](#), earlier in this chapter.

- To enlarge the notebook pane to its maximum height, click the **Maximize** button.

On the toolbar, the **Maximize** button becomes the **Restore** button, and the **Minimize** button becomes unavailable.

- To restore the notebook pane to its default height, click the **Restore** button.



You can see a description of each button by resting the pointer on the button to display a tooltip.

Viewing Objects in a Hierarchy

A project defines a hierarchy in which objects reside at levels determined by the individual object types. Only folders can occupy the primary level, directly below the project node. Objects of other types must reside in folders.

The following views can contain such hierarchies:

- The hierarchical content table.
- The **Attachments, Links, and Versions** tabs and floating windows.
- The **Search Results** dialog window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.

In any folder's hierarchy, objects can be nested from the top level to progressively subordinate levels. The leftmost column displays a plus sign (+) for each object that has subordinates, or *members*, at the next lower level.

You can view only these members, which may themselves have members that extend the hierarchy to even lower levels. Or, you can view the entire hierarchy below the object, displaying its members and all lower level objects simultaneously.

For a folder or a group, members can be folders, requirements, building blocks, groups, and the system-defined and user-defined subtypes of those object types. For a requirement or a building block, only objects of that type are allowed as members, including system-defined and user-defined subtypes based on that object type.

The members of a folder, a requirement, or a building block reside in the same location as their superior. Therefore, these hierarchies show relationships of child and parent objects. Such relationships do not apply to groups and their members, because group members reside elsewhere in the project. Although group members occupy the next lower level, child and parent relationships shown for members reflect hierarchies in the actual locations of the members. For more information, see [Using Groups to Maintain Objects](#) in chapter 4, *Maintaining a Project*.

For a selected object of any of those types, the **Member Count** property shows the number of members at the next lower level. You can view this property in the **Properties** tab or floating window. For more information, see [Properties Tab](#) and [Using Tabs in Floating Windows](#), later in this chapter.

To view only the members of an object:

In the leftmost column, click the plus sign (+) to the left of the object.

The members are displayed directly below the object. The object's plus sign becomes a minus sign (-), which you can click to hide the members.



For performance reasons, plus signs are shown for some objects that may or may not have members when:

- A member has been deleted from the object and not yet emptied from the recycle bin (this affects the content pane but not the navigation pane).
- The object has members that do not match the current effectivity.
- The current user does not have **Read** access to the object's members. If you have questions about folder access, consult your project administrator.

To view the entire hierarchy below an object:

Select the object, pull down the **View** menu, and choose **Expand All**. Or, right-click the object and choose **Expand All** from the pop-up menu.

The object's members and all lower level objects are displayed. A minus sign (-) is shown for each object that has members. You can click a minus sign to hide all objects below that level.

Filtering Objects in a Hierarchy

Filtering objects in Architect/Requirements allows you to filter the displayed rows of data in a hierarchical view down to a smaller subset so that only objects whose property values meet the filter criteria that you specified are displayed. If you've ever worked with filters in a spreadsheet like Microsoft Excel, then you are already familiar with the concept of what data filtering does. It filters the data according to user specified filter criteria so that only the rows with objects that meet the filter rules displays. Because there is a filter available for each property column in the hierarchical view, the Architect/Requirements menu command refers to this as **column filters**. There is no ability to filter on the tree column of hierarchical view because the tree structure is needed to give objects their context.

You can filter objects in the following views:

- The hierarchical content table.
- The **Search Results** dialog window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.

One of the most important concepts to understand about data filtering in Architect/Requirements is that it is static filtering, not dynamic filtering. In static filtering, if the data changes to something that does not meet the filter criteria, it is still displayed until a user requests that the filter rules are applied again. When using static filtering and the new data appears in the table, it is not compared to the current filtered criteria. The new information is allowed to display even if it does not meet the filter rules because a user action caused it to appear. If it didn't work this way, then it would lead to all kinds of user interface problems. For example, if you attempt to insert an object while filters were on and dynamic filtering was being used, then the moment that you requested to insert a new object, it would probably not display because it did not meet one of the filter rules. Because it is a new object, you will not have an opportunity to enter any of its property values, including changing its name from the default value. If dynamic filtering were used, you almost always have to clear the filters before inserting an object or making other changes to property values that didn't match the current criteria.

Filtering Objects Versus Filtering Properties

In Architect/Requirements, there are two types of filtering: object filtering and property filtering. Although they both reduce the number of rows showing in a view, each one is unique in its use and purpose. It is important to understand the difference between the two so that there is no confusion about which menu choice to choose.

The other type of filtering in Architect/Requirements is called **Property Filters**. It is only available in the property tab (fixed or floating). It controls the number of properties and their values showing in the Properties view according to the group that a user selects. For example, there are choices to show all properties or only those that are editable, read only, user defined, system, or properties associated with a saved view. For more information about filtering properties in the properties tab, see [Filtering Property Tables](#).

Showing and Hiding Filters

Because filters can either be visible or invisible, the **View** menu has an option to **Toggle Column Filters** from their current state to the opposite state. If the filters are not showing, using **Toggle Column Filters** turns them on and if they are off, using the command a second time displays them.



Hiding filters does not clear the filters or the data. The previous filter values are still present and can be seen again if you show the filters.

This menu command attempts to work for the currently selected hierarchical view. It does not work on a standard, flat table view such as the Properties tab. If the currently selected view does not allow filtering, a warning appears. Filtering is probably most useful in the **Content view** and the **Search Results View**.



The **Navigation Tree** is neither a hierarchical nor a flat table; it is a tree structure. It is used to navigate a project's folder structure and controls what shows up in the content view. As a matter of convenience, if you use the navigation tree to navigate what you want to appear in the Content area and then choose **Toggle Column Filters** while the **Navigation Tree** is the currently selected view, the application toggles the column filters for you in the content area.



The Architect/Requirements varies the font used in the content table to indicate the state of the filtered data. Italics and color variations are used for this purpose.

When a filter is applied and an object matches the criteria, the entire parent hierarchy for that object is displayed even if all the parent objects do not match the criteria. The parent objects, which are displayed to provide a context to the matching objects, have the font in maroon. The maroon color indicates that the row does not match the filter criteria. However, it indicates that the row is a parent of an object that does match the filter criteria.

When using the filters in Architect/Requirements, italics are used to indicate that the data that is showing may not match the filter criteria. This could occur if data is added after the data was last filtered. It may also occur if data is filtered on a column and then the column is removed without first removing the filter. Rows in italics are in an undetermined state, meaning it is unknown whether they match the filter criteria. When the content table is filtered and any rows are added to the table, those rows show up with an italicized font to indicate that they were added since the data was last filtered. Because the filtering in Architect/Requirements is static, not dynamic, the new data may or may not match the specified filter criteria. It must be filtered again in order for its correct state to be determined. Adding a row to already filtered data can be accomplished in many ways, including, but not limited to, creating a new object, pasting in an object, or undoing a delete.

Filter Toolbar

Filters appear as another small table on top of the table in the currently selected hierarchical view. The first column on the left of the filter table is the toolbar that is used to execute filter commands. It contains five icons:



Filters the data according to the currently selected filter choices.



Unfilters the data. This option keeps any filter values and operators that you selected, but shows the data as though it was unfiltered. This is very useful when you want to momentarily see an object that was filtered out of the view and then quickly return to the filtered state.



Clears all of the filters and restores the data to an unfiltered state.



Shows the row of filter operators.



Hides the filters.

Filter Choices

To the right of the filter toolbar there is a filter for each column in the hierarchical view. Each filter is an editable dropdown selection box that allows you select from a list of values or type in a value.

The first four entries in each filter are always the same. These constants are shown in all capital letters and are described below.

ALL (Default value) If this value is selected, all of the rows are considered to have met the filter criteria. In other words, no data is hidden as a result of this filter value; any value (including blanks) is accepted.

BLANKS Only shows rows whose data for that column property is empty. In other words, it does not contain any characters.



Blanks do not equate to spaces. If you type only spaces in a field, the field contain characters—it is not empty—and it will not match to **BLANKS**.

NON-BLANKS Only shows rows whose data for that column property is not empty. In other words, if any characters are in the cell for that row, the object matches that filter.

----- (Separator). A dashed line serves as a separator in the list between the fixed, system-defined values and the dynamic values that appear based on the data that is available for that column.

The rest of the values, after the fixed system-defined ones, are dynamically derived based on the data. For all property types except for multichoice properties, each unique value within that column is listed.

However, multichoice properties are an exception. There are many different combinations of choices that could show up. For example, if the choices are “a”, “b” and “c”, the value of “a” could appear by itself or as “a, b”, “a, c” or even “a, b, c”. Instead, each choice from the property definition is listed only once.



The values that appear in a multichoice and a single-choice filter dropdown box are derived differently:

- Single-choice properties behave just like any other property; only the unique values that are displayed in a column for the currently showing rows are shown in the box. This may prevent you from seeing every possible value because filtering on values that are not present in the currently displayed data always results in all rows being excluded; hence, no data appear.
- Multichoice properties are unique. Because you can choose none, one, or more than one choice for each object, a variety of combinations can exist. Therefore, you need is a single entry for each possible choice, not the combination of choices appearing in the table cells.



As soon as a filter choice is changed, the system filters the data immediately. This saves you from having to make a filter choice and then click the **Filter** icon.

Filter Operators

The filter operators are located above the filter choices. To minimize the amount of space used by filters, and to maximize the amount of space available for data to show, these operators are hidden by default even when filters are first displayed. Also, the most commonly used values for each filter type are selected by default. As long as you want to match on a data value equaling a chosen filter value, the default operator of **Contains** for a multichoice field and **Equals (=)** for all other fields work without your having to display the filter operators. However, if you want to use advanced filtering operators such as **Not Equals (!=)**, **Starts With**, **Ends With**, and **Contains**, you can display the filter operators. As described in [Filter Toolbar](#), you can show the filter operators by clicking the fourth icon in the toolbar that contains the ellipses.



There is an exception to the general rule that the filter operators have several choices such as **Starts With**, **Ends With**, etc. In Architect/Requirements, there are certain system count properties that always display in a hierarchical table column with their corresponding icon. For example, a paper clip icon indicates that there are attachments for this object. And, an interlocking chain link icon indicates that there are traceability links for an object. In these cases, the only operator value that makes sense is **Equals (=)**. Therefore, it is the only available operator. Options like **Contains**, **Starts With** and **Ends With** do not make sense with a binary choice of a cell either having and or not having an icon.

In the overview of this section, it was mentioned that Architect/Requirements uses static filtering. This means that every user action does not automatically cause the data to be filtered. This is true with the operators. Selecting an operator does not filter the data. It selects the operator that will be used whenever a filter value is changed or the **Filter** icon is selected. Since there are always really two parts to filtering (operator and filter value) that must be specified, only one of these (changing the filter value) immediately begins the process of filtering the data.

Filtering Data

To filter the data, show the filters and optionally the filter operators. Select a filter operator if you do not want to use the default operator and then select a filter value. Only those rows that match the specified filter criteria are displayed.

Filter value dropdown boxes are editable. This means that instead of just selecting a value, you can type in a value. This is especially helpful when using operators such as **Contains**, **Starts With**, and **Ends With**. For example, a column filter may list choices of **somewhere**, **awesome**, **somehow**, and many others, but you would like to filter on all rows where the value for that column starts with **some**. To do this, first select the **Starts With** operator. Then double-click the filter dropdown box, type the word **some**, and press the enter key.

Saved Views

In Architect/Requirements, named views can be saved in the database and used to control which properties appear in the content area as well as the **Search Results** window and **Properties** tab. Filters and their state are also saved with the view information; for example, whether filters are showing, operators are showing, and the filter and operator values are all saved with the named view. These settings are restored when you choose the **Named View**, resulting in the data being filtered according to the saved filter values. This filtering makes the data conform more to the exact state that a **Named View** was saved with. For more information, see *Customizing Views of Property Columns*.

Similarly, filter settings are also saved with local view settings. For example, if you are in a project where you have filters visible and you switch to a project where filters were not displayed when you last visited that project, the filters are hidden for that project. When you switch back to the project where filters were showing, the filters return. Filters are not automatically applied to the view content; the settings of previous filter values correspond to the rules of when columns settings are restored.

When there are more than 28 saved views (in addition to **System Settings** and **Default View**) for the current user in the selected project, the user can select a view by choosing the **Default View** or the **Select a Saved View** drop down options. The user can also right-click a folder and choose **Select View** to access the options.

If the user chooses the **Default View** option, then the default view column settings are applied. If the user chooses **Select a Saved View**, then a dialog window is displayed that allows the user to choose a view from a drop down box of the saved views. The user can choose a saved view and the click **OK** to apply the settings of the selected view or click **Cancel** to close the dialog window without applying any settings.

Printing a Selected Panel

You can print a selected panel, such as the navigation tree, the content pane, or the notebook pane, in the Architect/Requirements. You can choose **File**→**Print** option or click the **Print** button on the Architect/Requirements toolbar.



- For the Search module, the print command prints the body of that module as it appears on the screen.
- For all other panels, including Search Results, the entire content of the active panel is printed, which may be more than the currently visible portion. The print command prints everything that the active panel's scroll bars would allow you to reach.
- Each printed panel is scaled to fit one page in width. However, the print command prints as many pages as necessary to show all the content vertically.
- If the active panel is wide, the one-page horizontal scaling may make the content too small to read. Try reducing the number of columns in the panel, or reduce the column widths. Choosing landscape mode gives a wider effective paper width and improves the scaling marginally. There is no option to print across multiple pages horizontally.
- When you are printing for the first time in a session, a message window may appear to indicate a delay in opening the Print dialog window. The Print dialog window opens quickly when you try to print again in the same session.
- The print commands in Architect/Requirements are built on Java libraries that are relatively new. Some options in the Print dialog window may not work as expected for

all printers. They include options such as color versus monochrome, print resolution, and single versus duplex printing. These libraries are expected to improve over time.

Viewing Architect/Requirements System Properties

Architect/Requirements Customer Support may request information about your system to answer your questions or resolve problems. Obtain this information from the System Information dialog window, which displays properties for:

- The Architect/Requirements software.
- The required Java software.
- The computer on which the client is installed.

You can rearrange and resize the columns in the table, and you can sort the columns in ascending or descending order. In addition, you can do the following:

- Choose a group of related properties to display.
- Clear one or more rows to reduce the number of properties.
- Save the system information in a text file.
- Export the system information to Microsoft Excel.

To view Architect/Requirements system properties:

Pull down the **Tools** menu and choose **System Information** to display the System Information dialog window, and then click the **System Properties** tab.

- To view only Java properties, click **Show Java**.
- To view only those properties that are not Java properties, click **Show non-Java**.
- To view all properties, click **Show All**.
- To clear one row, select the row, and then click **Hide Node(s)**.
- To clear a group of adjacent rows:
 - . Select the first row in the group, hold down the shift key or the control key, and then select the last row.
 - . Click **Hide Node(s)**.
- To save the currently displayed properties in a text file:

. Click **Save As** to display the Save Properties dialog window.

The **Save in** field displays the currently selected drive or folder. You can select another drive or folder by clicking the button to the right.

The **File name** field displays the default name of the text file. You can enter another file name in this field.

- . Click **Save**.

The Save Properties dialog window closes, and Architect/Requirements saves the file with the specified name in the selected location. You can print your system information from this file, and you can send the file to E-mail recipients as an attachment.

- To export the current table to Microsoft Excel, click **Export to Excel**.

Excel opens a read-only file (**.html**), in which the worksheet's columns, rows, and cell contents match those in the System Information dialog window.

You can use all of Excel's viewing, printing, and navigation features to analyze your system information. From within this file, you can send the table to E-mail recipients in the body of a message. This file is temporary and is deleted from your computer when you exit Architect/Requirements.

You can create a permanent file by pulling down the Excel **File** menu and choosing **Save As**, and then assigning the file name, file type, and location. This file remains on your computer when you exit Architect/Requirements.

Working With the Client Log File

In the System Information dialog window, the **Client Log** tab displays the contents of the client log file. This text file (.txt) is stored on your computer, in the location shown in the **Log Path** field. Architect/Requirements maintains a backup log file in the location shown in the **Backup Path** field.



Use this tab to provide information to Architect/Requirements Customer Support in resolving problems with the client. System performance is diminished if you use this tab during normal operations.



The **Client Log** tab shows the log file contents as of the time when you displayed the System Information dialog window. This data is not updated dynamically because of performance considerations. However, the data is updated when you do certain actions in the tab.

- To update the log file with the latest data, click **Refresh**.
- To copy the log path or the backup path to the Windows clipboard, click the **Copy Path** button to the right of the corresponding field.



You can use the path to locate the file in other applications. In Microsoft Outlook, for example, you can attach the file to a message by pasting the path into the **File name** field and clicking **Insert**.

- To open the log file in Microsoft Notepad, click **Open in Notepad**.

Architect/Requirements refreshes the log file and copies it to a temporary file in the same directory as the log file. Then, the temporary file opens in Notepad.



You can do the following through standard Notepad features:

- o Find and replace text.
- o Edit the log file.
- o Print the log file.
- o Save the log file with a different name or in a different location.

The **temp_** prefix indicates the temporary file. This file remains on your computer and is overwritten each time you click **Open in Notepad**.



Clicking **Open in Notepad** while the file is already open generates multiple Notepad windows. Each window shows the latest data as of the time the window was generated.

You can save or copy the temporary file to other locations. You can also delete this file from the log file directory.

- To copy any portion of the log file to the Windows clipboard, select the text in the scrolling pane and press control-C.
- To add a line of text, enter the text in the **Append Text Line** field and click the **Append Text** button.

The data is refreshed and the text is added as the last line. A date and time stamp precedes the text.



The text you enter in the field remains there until you delete it. Therefore, you can reuse text elements for additional lines. For example, you may enter leading characters to make your entries unique. Later, you can open the log file in Notepad and search for the characters to find the lines you added.

- To add a blank line, place the cursor in the **Append Text Line** field and click the **Append Text** button.

The log file is refreshed and the blank line is added at the bottom.

- To clear the current data and start a new log:



Before you clear the current data, you can keep a copy by renaming the log file, opening it Notepad and saving it with another name, or copying it to another location. The log file location is shown in the **Log Path** field.

- Click **Empty Log**.

A confirmation message asks if you are sure you want to empty the client log.

- To continue, click **Yes**.

Architect/Requirements copies the current data to the backup log file. In the **Client Log** tab and in the log file, all previous data is removed and new data is added.



For better focus on your client problem, empty the log file before you run your test case. By starting a new log file that contains only relevant data, you help Customer Support isolate the problem.

Setting Client Debugging Options

In the System Information dialog window, the **Logging Options** tab provides options for writing certain debugging information to the client log. Under the **Logging Option** heading, each successive option incorporates the preceding option and writes additional information.



Use this tab to provide information to Architect/Requirements Customer Support in resolving problems with the client. System performance is diminished if you use this tab during normal operations.



The System Information dialog window is not a modal dialog window, and you can leave it open while debugging a problem. This allows you to view the log and set debugging options without repeatedly opening and closing the dialog window.

1. Select one of the following options:

Option	Description
Off	Deactivates debug logging and returns to normal logging. For each server call, standard information is written to the log file without extra debugging information.
On	For development purposes. Allows developers to put in special code that is run only if the basic debug flag is turned on. Has no effect in a normal production environment.
Response Time	Adds server call response time. For each server call, writes the last method name, the response time, the calling command, and the first parameter.
Short	Adds all parameters. For each server call, writes the method name, the response time, the calling command, and all parameters.
Trace	Adds a full trace of method names. For each server call, writes one trace line and the information for the Short option.
Long	Adds formatting to separate each call. For each server call, writes a blank line, a header line, the information for the Trace option, and a footer line.

2. To activate the selected option, do one of the following:

- Click **OK** to activate the option and close the dialog window.
- Click **Apply** to activate the option and leave the dialog window open.



On the first server call after debug logging is activated, a warning symbol is displayed in the status bar at the bottom of the client window. To the right of the symbol, a message shows which option is activated.

The symbol and message cannot be disabled while debug logging is activated. They are removed when debug logging is deactivated.

You can also do the following in the **Logging Options** tab:

- To add a line of text, enter the text in the **Append Text Line** field and click the **Append Text** button.

The data is refreshed and the text is added as the last line. A date and time stamp precedes the text.



The text you enter in the field remains there until you delete it. Therefore, you can reuse text elements for additional lines. For example, you may enter leading characters to make your entries unique. Later, you can open the log file in Notepad and search for the characters to find the lines you added.



To keep a constant line available, you can use this **Append Text Line** field in combination with the same field in the **Client Log** tab. For example, you can use one field to add unique characters, such as a separator line containing dashes, and use the other field to repeatedly change the text you want to add to the log. When you are finished entering different text lines, return to the other tab where the separator line remains in that field ready to be added after the text.

- To add a blank line, place the cursor in the **Append Text Line** field and click the **Append Text** button.
- To clear the current data and start a new log:



Before you clear the current data, you can keep a copy by renaming the log file, opening it Notepad and saving it with another name, or copying it to another location. The log file location is shown in the **Log Path** field in the **Client Log** tab.

- Click **Empty Log**.

A confirmation message asks if you are sure you want to empty the client log.

- To continue, click **Yes**.

Architect/Requirements copies the current data to the backup log file. In the **Client Log** tab and in the log file, all previous data is removed and new data is added.



For better focus on your client problem, empty the log file before you run your test case. By starting a new log file that contains only relevant data, you help Customer Support isolate the problem.

Changing Your Architect/Requirements Password

Your Architect/Requirements password can contain any combination of uppercase and lowercase letters, numerals, and symbols. There are no reserved characters, nor is there a maximum number of characters.



By default, there is no minimum number of characters, nor are any alphanumeric and numeric characters required. However, your Architect/Requirements system administrator may set restrictions for passwords at your site. If you have questions about changing your password, consult your Architect/Requirements system administrator.

To change your Architect/Requirements password:

1.

Pull down the **Tools** menu and choose **Change Password**.

The Change Password dialog window is displayed.

2. In the **Current Password** field, enter your current password.

3. In the **New Password** and **Verify New Password** fields, enter your new password.

4. Click **OK**.

The dialog window closes, and a confirmation message appears. Use the new password the next time you start Architect/Requirements.

Exiting Architect/Requirements

If you changed your Architect/Requirements password during the current session, your new password becomes effective when you exit Architect/Requirements.



Before you end the current session, close all temporary Microsoft Word or Excel files that you opened from Architect/Requirements. Otherwise, those files remain on your computer when you exit Architect/Requirements.

To exit Architect/Requirements:

1. Pull down the **File** menu and choose **Exit**.

A message appears, asking if you want to exit the application.

2. Click **Yes**.

The Architect/Requirements main window closes.

Procedure Notes

Step 1: You can also press control-Q or Alt-F4. Or, click the **Close** button in the upper right corner of the main window.

Chapter 4: Maintaining a Project

This chapter contains an overview of Architect/Requirements projects and their organization, including instructions for working with the objects in a project.

Overview of Projects

In Architect/Requirements, projects reside at the highest level of organization and are superior to all other objects. Each project defines a logical boundary for folders, requirements, building blocks, groups, notes, and trace links. These objects reside within a project at levels determined by the individual object types.

Directly below the project node, the primary level is reserved for folders, in which the other types of objects are organized into logical categories. Folders can be used, for example, to store requirements by source (such as customer or mil-spec), by related discipline (such as testability or reliability), or by technology (such as hardware or software).

As folders contain files and other folders in Microsoft Windows, folders contain requirements, building blocks, groups, and other folders in Architect/Requirements. A given folder can contain any number and combination of those objects. Consequently, requirements and building blocks can be stored in folders that are nested within other folders at progressively lower levels in the project.

A more document-centric method is to structure requirements and building blocks in a hierarchy of parent, child, and sibling relationships within a single folder. For each requirement and building block, the **Number** property value corresponds to that object's level in the hierarchy, similar to numbered paragraphs in Microsoft Word's outline view. This structured approach should be used if requirements and building blocks are to be exported to Word later in the process. However, folders containing those objects should not be nested within other folders.

Although both methods of organization can be used together, it is more effective to choose the method that is better suited to the particular requirements management process and to use that method exclusively. Consideration should be given also to the number of objects that exist at any level, at the top level in a folder, for example, or below parent objects in a hierarchy. Architect/Requirements imposes no limitations, but there is an impact on usability. If a folder contains several hundreds or thousands of objects, it is difficult for the user to locate the objects of immediate interest. There is also a performance consideration in populating a view with a large number of objects, which naturally takes longer than with a smaller number.

Objects can be copied from one location to another within a project and from one project to another. Trace links can be created between objects in different projects. However, objects can be moved only within the project where they reside.

Opening a Project

The navigation tree contains a node for each project to which you have access. Project nodes occupy the next level below the root node in the navigation tree. For more information, see [Navigation Tree](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Only folders can occupy the primary level in a project hierarchy, the level directly below the project node. Those folders are displayed in the hierarchical content table when you open a project. For more information, see [Hierarchical Content Table](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To open a project:

In the navigation tree, select the project node.

The content table displays the folders at the project's primary level.

You can display these folders in the navigation tree by double-clicking the project node, by clicking the plus sign (+) to the left, or by right-clicking the project node and choosing **Expand** from the pop-up menu.



If you click a plus sign and the folder does not expand, it may contain subfolders to which you do not have **Read** access. If you have questions about folder access, consult your project administrator.

Opening a Folder

A project can contain any number of folders, which can be nested to progressively subordinate levels. You can open a folder from the navigation tree or the hierarchical content table. For more information, see [Navigation Tree](#) and [Hierarchical Content Table](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Each folder selected in the navigation tree can have a unique view of property columns in the content table. These views let you easily work with the properties. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.

To open a folder from the navigation tree:

1.

Click the plus sign (+) to the left of the project node to display the folders at the project's primary level.

If necessary, click the plus signs for these folders to display lower level folders.



If you click a plus sign and the folder does not expand, it may contain subfolders to which you do not have **Read** access. If you have questions about folder access, consult your project administrator.

2. Select the folder to display the top level objects in the content table.

Procedure Notes

Step 1: You can also right-click the project node or folder and choose **Expand** from the pop-up menu. Or, double-click the project node or folder.

To open a folder from the hierarchical content table:

In the **Name** column, select the folder, and then do one of the following:

- To display only objects in the selected folder, pull down the **File** menu and choose **Open**, or right-click the folder and choose **Open** from the pop-up menu.

The content table displays the objects at the folder's top level. In the navigation tree, the folder is highlighted, and it is expanded if it contains lower level folders.

You can also display the top level objects by clicking the plus sign (+) to the left of the folder or by double-clicking the folder's object type indicator.

- To display objects in the selected folder and maintain the current view of other folders, do one of the following:
 - For all objects in the folder, pull down the **View** menu and choose **Expand All**, or right-click the folder and choose **Expand All** from the pop-up menu.
The content table displays all objects at each level of the folder hierarchy. Any other folders remain as previously displayed.
 - For only the top level objects in the folder, click the plus sign (+) to the left of the folder.
The content table displays the objects at the top level of the folder hierarchy. Any other folders remain as previously displayed.



If you click a plus sign and the folder does not expand, it may contain subfolders to which you do not have **Read** access. If you have questions about folder access, consult your project administrator.

Creating a Folder

You can create a folder at the primary level of a project, directly below the project node, or at any lower level as a child within an existing folder. Folders can also be created in live Excel. For more information, see [Creating Objects in Live Excel](#) in chapter 9, *Working With Object Properties*.

The new folder receives certain system-defined properties, including the **Subtype** property. You can assign the default subtype (**Folder**), a system-defined folder subtype (**Document**), or a user-defined subtype. The folder may also receive user-defined properties.

The subtype and all other editable properties can be changed after the folder's creation. In addition, you can copy the new folder to other locations, link it to other objects, and attach notes. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#); chapter 9, *Working With Object Properties*; [Copying Objects](#), later in this chapter; chapter 7, [Showing Object Relationships With Trace Links](#); and [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.



You must have **Modify** permission for the project or the parent folder in which you intend to create the folder.

To create a folder at the primary level of a project:

1. Select the project node in the navigation tree, or select an existing primary folder in the hierarchical content table.
2. Assign the subtype by doing one of the following:
 - For the **Folder** subtype, pull down the **File** menu and choose the **New→Folder** options, or click the **Create New Folder** button on the toolbar.
 - For the **Document** subtype, pull down the **File** menu and choose the **New→Document** options.
 - For a user-defined subtype, pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window. To see the user-defined subtypes, click the plus sign (+) to the left of **Folder**, and click any lower level plus signs. Select a subtype and click **OK**.

The navigation tree displays the new folder in alphabetical order of the default name. The content table displays the folder in the last position at the primary level, with the default name in an open text field.

3. Enter the folder name in the text field, and then press the enter key.



Using the **View** field, you can set the folder to display a unique set of columns in the content table. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.

Procedure Notes

Step 2: You can also right-click the project node and choose the options from the pop-up menu. Or, press control-L for the **Folder** subtype, control-D for the **Document** subtype, or control-U for a user-defined subtype.

Step 2: To reverse this action, you can pull down the **Edit** menu and choose **Undo New Folder**, **Undo New Document**, or **Undo Create Subtype**; click the **Undo** button on the toolbar; or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

To create a folder within a folder in the navigation tree:

1. Select the parent folder.

To see lower level folders, click the plus signs. Or, right-click a folder with a plus sign and choose **Expand** from the pop-up menu.

2. Assign the subtype to the child folder by pulling down the **File** menu and choosing one of the following:

- Choose the **New→Folder** or **New→Child** options to allow the child to inherit the parent subtype.

You can also right-click the parent and choose the **New→Folder** or **New→Child** options from the pop-up menu. Or, click the **Create New Folder** or **Create New Child** button on the toolbar. Or, press control-L or control-K.

- Choose the **New→Document** options to:
 - Assign the **Document** subtype when the parent subtype is **Folder** or a user-defined folder subtype.
 - Allow the child to inherit the parent subtype when the parent is **Document** or a user-defined document subtype.

You can also right-click the parent and choose the **New→Document** options from the pop-up menu. Or, press control-D.

- Choose the **New→Subtype** options to display the Select Subtype dialog window and assign a subtype to the new folder specifically:
 - Click the plus sign (+) to the left of **Folder** to display the **Document** subtype and user-defined folder subtypes.

Plus signs may be shown also for these subtypes. Click the plus signs to display additional subtypes at lower levels.
 - Select a subtype, and then click **OK** to close the dialog window.

To display the dialog window, you can also right-click the parent and choose the **New→Subtype** options from the pop-up menu. Or, press control-U.

In the content table, the new folder is displayed as the last object at the top level of the parent folder, with the default name in an open text field.

3. Enter the folder name in the text field, and then press the enter key.



Using the **View** field, you can set the folder to display a unique set of columns in the content table. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.

Procedure Notes

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Folder**, **Undo New Document**, **Undo New Subtype**, or **Undo New Child**. You can also click the **Undo** button on the toolbar or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

To create a folder within a folder in the hierarchical content table:

1. Do one of the following:

- Select the parent folder, pull down the **File** menu, and choose the **New→Child** options. Or, right-click the parent and choose the **New→Child** options from the pop-up menu. You can also click the **Create New Child** button on the toolbar, or press control-K.

The new folder inherits the subtype of the parent and is displayed as the last object at the next lower level, with the default name in an open text field.

- Within the parent folder, select an existing folder as a sibling of the new folder, and then pull down the **File** menu and choose one of the following:
 - Choose the **New→Folder** options to assign the subtype of the sibling to the new folder. Or, right-click the sibling and choose the **New→Folder** options from the pop-up menu. You can also click the **Create New Folder** button on the toolbar, or press control-L.
 - Choose the **New→Document** options to assign the **Document** subtype to the new folder when the sibling subtype is **Folder** or a user-defined folder subtype. When the sibling subtype is **Document** or a user-defined document subtype, that subtype is assigned to the new folder. You can also right-click the sibling and choose the **New→Document** options from the pop-up menu, or press control-D.
 -

Choose the **New→Subtype** options to display the Select Subtype dialog window and assign a subtype to the new folder specifically:

- Click the plus sign (+) to the left of **Folder** to display the **Document** subtype and user-defined folder subtypes.

You can click any additional plus signs to display more subtypes.

- Select a subtype, and then click **OK** to close the dialog window.

To display the dialog window, you can also right-click the sibling and choose the **New→Subtype** options from the pop-up menu. Or, press control-U.

The new folder is displayed as the last object at the level of the sibling, with the default name in an open text field.

2. Enter the folder name in the text field, and then press the enter key.



Using the **View** field, you can set the folder to display a unique set of columns in the content table. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.

Procedure Notes

Step 1: You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Folder**, **Undo New Document**, **Undo New Subtype**, or **Undo New Child**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Copying Objects

Folders, requirements, building blocks, groups, and notes can be duplicated as new objects in other locations. The destinations can be within the same Architect/Requirements project or in a different project, subject to the following conditions:

- Folders can be copied to the primary level and to the top levels of other folders. They cannot be copied to requirements, building blocks, notes, diagrams, or spreadsheets.
- Requirements can be copied to the top levels of folders and to other requirements as children. They cannot be copied to the project node or to building blocks, notes, diagrams, or spreadsheets. Each copied requirement receives a new **ROIN** property value.
- Building blocks can be copied to the top levels of folders and to other building blocks as children. They cannot be copied to the project node or to requirements, notes, diagrams, or spreadsheets.
- Groups can be copied to the top levels of folders. They cannot be copied to the project node or to requirements, building blocks, notes, diagrams, or spreadsheets.
- Notes can be copied to the project node, folders, requirements, building blocks, trace links, and to objects in the **Attachments** tab and window, including other notes.
- Diagrams can be copied to folders, requirements, and building blocks. They cannot be copied to the project node, to notes or spreadsheets, or to other diagrams.
- Spreadsheets can be copied to folders, requirements, and building blocks. They cannot be copied to the project node, to notes or diagrams, or to other spreadsheets.



Objects that are copied to groups are not duplicated as new objects. Instead, they are associated as group *members* through references to their present locations. For more information, see [Using Groups to Maintain Objects](#), later in this chapter.

When you copy certain object types from one project to another, the way you select the objects to copy from the originating project determines the behavior of the objects in the destination project:

- When the selection includes a shortcut and its referenced object, both objects are duplicated and the reference is within the destination project. When the selection does not include the referenced object, the copied shortcut in the destination refers to the object in the originating project. For more information, see [Working With Shortcuts](#), later in this chapter.
- When the selection includes a group and any of its members, each selected member is duplicated and becomes a member of the copied group in the destination project. When the selection does not include at least one member, the copied group is empty in the destination project.

- Trace links and connections between selected objects are preserved in the destination project. Trace links and connections to objects outside the selection are not copied with the selected objects.
- The **Version Type** and **Baseline** properties of copied versions are reset to their default values in the destination project. For versioned objects selected in the hierarchical content table, only the version specified in the effectivity field is copied to the destination project.
- Subtypes of selected objects are preserved in the destination project. A new subtype is automatically created in the destination if a copied subtype name does not exactly match an existing subtype name in the destination.
- Each selected spreadsheet is copied in its current state, for example, synchronized with the database through live Excel. However, the objects represented in the spreadsheet reside in the originating project.



Diagrams cannot be copied to other projects. If diagrams are included in the selection, they are ignored.

The hierarchical content table displays a plus sign (+) for each object that has one or more members at lower levels. In copying such objects, consider the following:

- When you copy a folder, you also copy all of the objects that it contains, including other folders. These objects may themselves have members, all of which are copied with the folder. The folder takes the last position at the next level below the destination object, maintaining its own hierarchy of objects. Its requirements and building blocks retain their **Number** property values.
- When you copy a requirement or a building block, you also copy all of its direct children and lower level descendants. The copied object takes the last position at the next level below the destination object, receiving a new **Number** property value for that level and position. The parent and child relationships are preserved, and each of these objects receives a new **Number** property value for the level and position below its parent.



- You must have **Modify** permission for each intended destination.
- To copy building blocks, TRAMs, and diagrams, and to copy a folder that contains those objects, you must have **Architect** privilege for the project.
- Objects cannot be copied from or to your Recycle Bin.

To copy objects:

1. Select each object that you want to copy.

In the navigation tree, you can select only one folder.

In the hierarchical content table or the **Attachments, Where Used**, or **Versions** tab or floating window, you can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

2. Pull down the **Edit** menu and choose **Copy**.

For a selection in the navigation tree, the content table, or the **Attachments**, **Links**, and **Versions** tabs or windows, you can also hold down the left mouse button on the selection, hold down the control key, and drop the objects on the destination. This action is subject to the conditions in step 3.

3. Set the destination by doing one of the following:

- To copy selected folders to the project's primary level, select the project node in the navigation tree.

The selection cannot include requirements, building blocks, groups, or notes.

- To copy the objects to a folder's top level, select the folder in the navigation tree, the hierarchical content table, or the **Links** tab or window.
- To copy the objects to a requirement or a building block, select the parent in the hierarchical content table or the **Links** or **Versions** tab or window.

For a requirement, the selection must consist solely of requirements. For a building block, the selection must consist solely of building blocks.

- To copy a selection that consists solely of notes:
 - If the destination is a folder, select it in the navigation tree, the hierarchical content table, or the **Links** tab or window.
 - If the destination is a requirement or a building block, select it in the hierarchical content table or the **Links** or **Versions** tab or window.

4. Pull down the **Edit** menu and choose **Paste**.

To duplicate these objects in additional destinations, repeat steps 3 and 4.

Procedure Notes

Step 2: You can also right-click the selection and choose **Copy** from the pop-up menu. Or, click the **Copy** button on the toolbar or press control-C.

Step 4: You can also right-click the destination and choose **Paste** from the pop-up menu. Or, click the **Paste** button on the toolbar or press control-V. To reverse this action, pull down the **Edit** menu and choose **Undo Copy**, click the **Undo** button on the toolbar, or press control-Z.

Copying Object URLs

Architect/Requirements assigns a Uniform Resource Locator (URL) to each folder, requirement, building block, group, note, and diagram. For one or more selected objects, including group members, object shortcuts, and linked objects, you can copy the URLs to the Clipboard. You can then insert the URLs in Windows programs that support hyperlinks, such as Microsoft Outlook and Microsoft Internet Explorer.

After insertion, URLs can be used to start a new Architect/Requirements session, and to navigate to objects in the main window. For example:

- In Outlook, you can paste the URLs into the body of a message and send it to recipients such as colleagues and customers. Recipients can start Architect/Requirements and navigate to an object by clicking its URL in the message.
- In Internet Explorer, you can paste a URL into the **Address** bar, and then click the **Go** button to navigate to the object in Architect/Requirements.

Also, E-mail recipients can copy a URL from a message body to the Internet Explorer **Address** bar, and then click the **Go** button to start Architect/Requirements and navigate to the object.



- The destination program must support hyperlinks.
- To support navigation, Architect/Requirements must be configured for object linking. If you have questions about navigating to objects, consult your Architect/Requirements system administrator.

To copy object URLs:

1. In the hierarchical content table, the **Attachments** tab or floating window, or the **Links** tab or window, select each object whose URL you want to copy.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

If the destination program is Internet Explorer, select only one object.

2. Pull down the **Edit** menu and choose **Copy URL**→**Include Full Name** or **URL Only**. Choose **Include Full Name** to copy the URL link that includes the **Full Name** property, or choose **URL Only** to copy only the URL link.

For example:

Choosing the **Include Full Name** option for an object named **Requirement1** copies the following:

\ATK TBx\TBX Test Data\Folder\Requirement1 at

http://pnsvin07v01:7001/tcr/controller/ObjLauncher?wolf_objectid=1.0.24535

Choosing the **URL Only** option copies only the URL link as shown here:

http://pnsvin07v01:7001/tcr/controller/ObjLauncher?wolf_objectid=1.0.24535

The URLs are placed in the Clipboard for insertion in Windows programs.

Procedure Notes

Step 2: You can also right-click the selection and choose **Copy URL** from the pop-up menu.

Moving Objects

Objects in a project can be moved from their current locations to other locations. Destinations must be within the project and are subject to the following conditions:

- Folders can be moved to the project node and to the top levels of other folders. They cannot be moved to requirements, building blocks, groups, or notes.
- Requirements can be moved to the top levels of folders and to other requirements as children. They cannot be moved to the project node or to building blocks, groups, or notes. Each moved requirement retains its **ROIN** property value.
- Building blocks can be moved to the top levels of folders and to other building blocks as children. They cannot be moved to the project node or to requirements, groups, or notes.
- Groups can be moved only to the top levels of folders. They cannot be moved to the project node or to requirements, building blocks, notes, or other groups.
- Notes can be moved to the project node, folders, requirements, building blocks, trace links, and to objects in the **Attachments** tab and window, including other notes.



An object cannot be moved in the following conditions:

- If its parent is frozen.
- If the object exists under multiple versions of the parent object and moving it results in the object getting displayed under different parents in the user interface.

This is also applicable for promote and demote operations.

For example, a requirement **A** has a child requirement **B**. The requirement **A** is frozen and a new version **A1** is created. Moving the child requirement **B** to a new owner is not allowed because moving **B** results in multiple owners for **B**. If you then create a new version of **B**, the new **B1** will have only one owner and you can move it.

The hierarchical content table displays a plus sign (+) for each object that has one or more members at lower levels. In moving such objects, consider the following:

- When you move a folder, you also move all of the objects that it contains, including other folders. These objects may themselves have members, all of which are moved with the folder. The folder takes the last position at the next level below the destination object, maintaining its own hierarchy of objects. Its requirements and building blocks retain their **Number** property values.
- When you move a requirement or a building block, you also move all of its direct children and lower level descendants. The moved object takes the last position at the next level below the destination object, receiving a new **Number** property value for that level and position. The parent and child relationships are preserved, and each of these objects receives a new **Number** property value for the level and position below its parent.



- You must have **Modify** permission for the objects and the destination.

- To move building blocks, TRAMs, and diagrams, and to move a folder that contains those objects, you must have **Architect** privilege for the project.

To move objects:

1. Select each object that you want to move.

In the navigation tree, you can select only one folder.

In the content table, your Architect/Requirements Recycle Bin, or the **Attachments**, **Links**, and **Versions** tabs or windows, you can select multiple nonadjacent and adjoining objects.

2. Pull down the **Edit** menu and choose **Cut**.

For a selection in the navigation tree, the content table, your Recycle Bin, or the **Attachments**, **Links**, and **Versions** tabs or windows, you can also hold down the left mouse button on the selection and drop the objects on the destination. This action is subject to the conditions in step 3.

3. Set the destination by doing one of the following:

- To move selected folders to the level below the project node, select the project node in the navigation tree.

If other object types are also selected, only the folders are moved.

- To move the objects to a folder's top level, select the folder in the navigation tree, the hierarchical content table, or the **Links** tab or window.

- To move the objects to a requirement or a building block, select the parent in the hierarchical content table or the **Links** or **Versions** tab or window.

If object types other than the parent's are also selected, only the objects of the parent's type are moved.

- To move a selection that consists solely of notes:

- If the destination is a folder, select it in the navigation tree, the hierarchical content table, or the **Links** tab or window.

- If the destination is a requirement or a building block, select it in the hierarchical content table or the **Links** or **Versions** tab or window.



In the content table, you can move a requirement or a building block within its level by selecting only that object, pulling down the **Edit** menu, and choosing **Move Up** or **Move Down**. Or, right-click the object and choose **Move Up** or **Move Down** from the pop-up menu. You can also select only that object and click the **Move Up** or **Move Down** button on the toolbar.

4. Pull down the **Edit** menu and choose **Paste**.



If you cut and paste between two separate instances of Architect/Requirements, the objects are copied to the destination. They are not moved from the source window.



In certain operations, such as moving objects from the content window to the navigation window, it is possible to see an **Object Not Found** error even if the operation is successful. Architect/Requirements uses a locking model that provides improved

performance but occasionally displays inconsistent data. This locking model does not compromise database integrity. If the operation is successful and no additional errors are reported, it is safe to ignore the error message.

Procedure Notes

Step 2: You can also right-click the selection and choose **Cut** from the pop-up menu. Or, click the **Cut** button on the toolbar or press control-X.

Step 4: You can also right-click the destination and choose **Paste** from the pop-up menu. Or, click the **Paste** button on the toolbar or press control-V. To reverse this action, pull down the **Edit** menu and choose **Undo Move**, click the **Undo** button on the toolbar, or press control-Z.

Moving frozen objects

If you freeze a parent object and its child objects, you cannot move the objects. However, if you need to move the child objects to a new parent object, you need to create a new version of the parent object and the child objects together. If you create a new version of the parent object and the child objects separately, the child objects ends up with two different parent objects. If you create a new version a child object with a frozen parent, the new version of the child object is also a child of the frozen parent. However, if you version the parent object and child objects together, the new version of the child object retains its link with the new version of the parent object only.



If the frozen parent object and child objects are multiselected and a new version created, a new version of the parent object is created first. Hence, the child object is movable.

For example, if you have the following requirement structure:

```
R1
    R1.1
R2
```

Where R1 and R1.1 are frozen.

Following are the steps to move R1.1 to R2.

1. Create a new version of R1 and R1.1
2. Move R1.1 to R2.

In the example given above, if only the child requirement R1.1 is frozen and the parent requirement R1 is not frozen, you need to create a new version of R1.1 and move it to R2.



You can also multiselect R1 and R1.1 and version them at the same time. If you version R1.1 before you create a new version of R1 and try to move R1.1, the move fails.

Renaming an Object

After you copy or move an object, you may want to assign a name that matches the object's new context within the project. Also, a new name may be appropriate to reflect a change in an object's purpose or content.

You can rename an object selected in the following views:

- The navigation tree and the hierarchical content table. For more information, see [Navigation Tree](#) and [Hierarchical Content Table](#) in chapter 3, *Using the Architect/Requirements Main Window*.
- The **Attachments**, **Links**, **Where Used**, and **Versions** tabs and floating windows. For more information, see [Attachments Tab](#), [Links Tab](#), [Where Used Tab](#), [Versions Tab](#), and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



- You must have **Modify** permission for the object.
- To rename a building block, a TRAM, or a diagram, you must have **Architect** privilege for the project.

To rename an object:

1. Select the object, and then pull down the **File** menu and choose **Rename**.
A text field opens around the object's current name.
2. Enter the new name, and then press the enter key.
The object is displayed with its new name.

Procedure Notes

Step 1: To see lower level objects in a hierarchical column, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click the object and choose **Expand All** from the pop-up menu.

Step 1: You can also right-click the object and choose **Rename** from the pop-up menu. Or, press the F2 key. You can cancel this action and close the text field by pressing the escape key.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Using Groups to Maintain Objects

A group consists of references to existing objects that reside elsewhere within the project. Each reference associates one object as a *member* of the group, and each referenced object remains in its present location.

By creating a group and adding members, you can define a collection of references to objects that reside in various locations. Then you can maintain the member objects directly from the group, without switching to other folders or views.

The member objects can be folders, requirements, building blocks, notes, diagrams, and other groups. A group can reference any number and any combination of these object types, as well as their system-defined and user-defined subtypes. Trace links and their subtypes cannot belong to groups.

Maintenance actions that change an object's location must be done by selecting the object where it resides, not in a group to which it belongs. Such actions include moving, promoting, demoting, and deleting an object.

Other actions can be done from a group as if the member objects were selected in their actual locations. Depending on the object type, you can do the following:

- Copy objects and their URLs, rename objects, create shortcuts to member objects, send object data by E-mail, and export the view of members to Microsoft Excel. For more information, see [Copying Objects](#), [Copying Object URLs](#), and [Renaming an Object](#), earlier in this chapter, and

[Working With Shortcuts](#), [Sending Object Data by E-Mail](#), and [Exporting Objects to Microsoft Office Excel](#), later in this chapter.

- Open requirements, paragraphs, and notes in Microsoft Word to edit or view their content, and export those objects to Word. For more information, see [Exporting Objects to Microsoft Office Word](#), later in this chapter; [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*; and [Editing a Note](#) and [Viewing Note Content](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Open diagrams in Microsoft Office Visio to edit or view their content. For more information, see [Editing a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Create, view, and delete trace links, and navigate to linked objects. For more information, see [Creating Trace Links](#), [Linking to an Object in Another Teamcenter Product](#), [Viewing Object Relationships](#), [Deleting Trace Links for an Object](#), and [Navigating to a Linked Object](#) in chapter 7, *Showing Object Relationships With Trace Links*.
- Change the values of editable properties. For more information, see [Properties Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*, and [Editing the Properties of a Selected Object](#), [Editing Properties in Table View Cells](#), or [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

After creating a new group, you specify its object references by adding members. For an existing group, you can add new members and remove current members at any time. For more information, see [Creating a Group](#), [Adding Group Members](#), and [Removing Group Members](#), later in this chapter.

Creating a Group

A group can be created only at the top level of a folder. The containing folder can reside directly below the project node or in another folder. The new group occupies the last position at the top level and cannot be promoted or demoted.

The new group receives certain system-defined properties, including the **Subtype** property. In this procedure, you can assign the default subtype, **Group**, or a user-defined subtype created by your project administrator. With either subtype, the group may also receive user-defined properties. The subtype can be changed after the group's creation. For more information, see chapter 9, [Working With Object Properties](#) and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

Before or after adding members, you can copy the group to other folders, link it to other objects, and attach notes. For more information, see [Adding Group Members](#), later in this chapter; [Copying Objects](#), earlier in this chapter; chapter 7, [Showing Object Relationships With Trace Links](#); and [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*.

Groups can also be created in live Excel. For more information, see [Creating Objects in Live Excel](#) in chapter 9, *Working With Object Properties*.



You must have **Modify** permission for the folder in which you intend to create the group.

To create a group:

1. In the navigation tree or the hierarchical content table, select the folder.

To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click the object and choose **Expand All** from the pop-up menu.

2. To assign the subtype, do one of the following:
 - For the **Group** subtype, pull down the **File** menu and choose the **New→Group** options, or click the **Create New Group** button on the toolbar.
 - For a user-defined subtype:
 - Pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window.
 - Click the plus sign to the left of the **Group** subtype, select a lower level subtype, and then click **OK**.
3. Enter the group name in the open text field, and then press the enter key.

Procedure Notes

Step 2: You can also right-click the folder and choose the options from the pop-up menu. Or, press control-G for the **Group** subtype or control-U for a user-defined subtype. You can also select the **Group** subtype in the Select Subtype dialog window. To reverse this action, pull down the **Edit** menu and choose **Undo New Group** or **Undo Create Subtype**, click the **Undo** button on the toolbar, or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Adding Group Members

For a new or existing group, you can add folders, requirements, building blocks, notes, diagrams, and other groups as members. These object types, and their system-defined and user-defined subtypes, can be added in any combination and in any number. You cannot add trace links or their subtypes.

Objects added as members are not moved to the group but remain in their present locations. Furthermore, parent and child relationships are retained in hierarchies where the objects actually reside. Therefore, a group is not the parent of its members, though they occupy the level below the group in the hierarchical content table.

Any lower levels below a member do not imply a hierarchy within the group itself. Below a folder, a requirement, or a building block, such lower levels show the member object's children, which do not belong to the group but are shown for additional context. Levels below a member group show objects that belong only to that group.

An object can belong to many groups concurrently and can be maintained from any one. Changes to a member affect the object in its actual location and are reflected in all other groups to which it belongs. Those actions can be done also for objects that do not belong to the group but are shown at levels below a member. For more information, see [Using Groups to Maintain Objects](#), earlier in this chapter.

You can add objects that you select in the navigation tree, the hierarchical content table, the **Attachments** tab or window, or the **Where Used** tab or window. You can remove members at any time. For more information, see [Navigation Tree](#), [Hierarchical Content Table](#), [Attachments Tab](#), [Where Used Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*, and [Removing Group Members](#), later in this chapter.

In the **Where Used** tab or window, you can see each group to which an object belongs. To see the number of members for a group, you can view its **Member Count** property in the **Properties** tab or

window. For more information, see [Properties Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.



You must have **Modify** permission for the group to which you intend to add members.

To add group members:

1. Select the objects that you want to add as members.

In the navigation tree, you can select only one folder.

In the hierarchical content table or the **Attachments** or **Where Used** tab or window, you can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.



If the selection is in the hierarchical content table and if the group is currently visible, you can drop the selection on the group to add the members and complete this procedure.

2. Pull down the **Edit** menu and choose **Copy**.
3. Select the group within its folder, pull down the **Edit** menu, and choose **Paste**.

The content table displays the members at the next level below the group.

Procedure Notes

Step 2: You can also right-click the selection and choose **Copy** from the pop-up menu. Or, click the **Copy** button on the toolbar or press control-C.

Step 3: You can also right-click the group and choose **Paste** from the pop-up menu. Or, click the **Paste** button on the toolbar or press control-V. To reverse this action, pull down the **Edit** menu and choose **Undo Copy**, click the **Undo** button on the toolbar, or press control-Z.

Removing Group Members

Objects belong to a group through references, with the member objects residing in locations other than the group's. When you remove members, you delete only the references that associate those objects with the group. The objects themselves are not deleted, nor are any other objects that are shown below the members for additional context. For more information, see [Adding Group Members](#), earlier in this chapter.

Because member objects remain in their present locations, any number of objects can belong to a particular group, and a given object can belong to any number of groups. Consequently, one or more members can be removed from a group at any time, and an object can be removed from one or more groups at the same time.

An object removed from one group remains a member of all others to which it belongs. It is removed from all groups when deleted from its actual location. You can view an existing object's groups in the **Where Used** tab or window. For more information, see [Where Used Tab](#) or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*, or [Deleting Objects](#), later in this chapter.

In the **Properties** tab or window, you can view other relevant information for removing group members. For example, the **Full Name** property shows a selected member's actual location, and the **Member Count**

property shows the number of members for a selected group. For more information, see [Properties Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Deleted member references are moved to your Architect/Requirements Recycle Bin and can be restored before it is emptied. For more information, see [Restoring Objects](#) or [Emptying Your Architect/Requirements Recycle Bin](#), later in this chapter.



You must have **Modify** permission for the group from which you intend to remove members.

To remove one or more members from a group:

1. In the navigation tree or the hierarchical content table, open the group's containing folder, and then click the group's plus sign (+) to display the members.

You can see the objects below all members by selecting the group, pulling down the **View** menu, and choosing **Expand All**, or by right-clicking the group and choosing **Expand All** from the pop-up menu.

2. In the content table, select each member that you want to remove.

You can select nonadjacent members by holding down the control key while you click the members. To select adjoining members, click the first member, hold down the shift key, and click the last member.

3. Pull down the **File** menu and choose **Delete**, and then click **Yes** when a confirmation message is displayed.

The selected member references are moved to your Recycle Bin.

Procedure Notes

Step 1: For more information, see [Navigation Tree](#) or [Hierarchical Content Table](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Step 3: You can also right-click the selection and choose **Delete** from the pop-up menu. Or, click the **Delete** button on the toolbar or press the delete key. To reverse this action, pull down the **Edit** menu and choose **Undo Delete**, click the **Undo** button on the toolbar, or press control-Z.

To remove an object from one or more groups:

1. Display the groups for the object by doing one of the following:
 - For an object in the hierarchical content table, select the object and click the **Where Used** tab. You can open the **Where Used** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
 - For an object in the **Attachments** or **Links** tab:
 - . With the **Where Used** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Where Used** window.
 - . Select the object in the **Attachments** or **Links** tab.

The **Where Used** tab or window displays the groups to which the object belongs.

2. In the tab or window, select each group from which you want to remove the object.

You can select nonadjacent groups by holding down the control key while you click the groups. To select adjoining groups, click the first group, hold down the shift key, and click the last group.
3. Pull down the **File** menu and choose **Delete**, and then click **Yes** when a confirmation message is displayed.

The object's reference in each selected group is moved to your Recycle Bin.

Procedure Notes

Step 3: You can also right-click the selection and choose **Delete** from the pop-up menu. Or, click the **Delete** button on the toolbar or press the delete key. To reverse this action, pull down the **Edit** menu and choose **Undo Delete**, click the **Undo** button on the toolbar, or press control-Z.

Working With Shortcuts

A shortcut is an object that represents another object by reference. A shortcut can represent a folder, a requirement, a building block, a data dictionary, or a data definition. The referenced object and the shortcut can reside in the same Architect/Requirements project. Also, a shortcut can be created in one project to reference an object in a different project.

Every shortcut must be created in a folder, including a shortcut to a folder. Shortcuts cannot be created at the primary level of a project, directly below the project node.

For the selected object, you can create a shortcut at the top level of the destination folder. Or, you can create the shortcut at a lower level, directly below another object of the selected object's type. You can select the object in the navigation tree, the hierarchical content table, or the **Links** tab or floating window.

For more information, see [Navigation Tree](#), [Hierarchical Content Table](#), [Links Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

The shortcut appears in the hierarchy with the same properties as the referenced object, including the **Number** property for a requirement or a building block. The shortcut also reflects the trace links and attachments for the referenced object. In addition, all lower level objects below the object appear below the shortcut, including all objects in a folder and all children of a requirement or a building block.

Without affecting the referenced object, you can copy, move, and delete a shortcut. For more information, see [Copying Objects](#) and [Moving Objects](#), earlier in this chapter, and [Deleting Objects](#), later in this chapter.

You can do the following on the shortcut as if you selected the referenced object itself:

- Rename the object and send data for the object by E-mail. For more information, see [Renaming an Object](#), earlier in this chapter, and [Sending Object Data by E-Mail](#), later in this chapter.
- Create a new object of the same type, as a sibling of the referenced object and in that object's location. For more information, see [Creating a Folder](#), earlier in this chapter; [Creating a Requirement Object](#) in chapter 5, *Managing Requirements*; or [Creating a Building Block](#), [Creating a Data Dictionary](#), or [Creating Data Definitions](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Open requirements and paragraphs in Microsoft Word to edit or view their content, and export those objects to Word. For more information, see [Exporting Objects to Microsoft Office Word](#), later in this chapter, and [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Create and edit diagrams in Microsoft Office Visio for building blocks. For more information, see [Live Visio Diagrams](#) and [Editing a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.
- Create, view, and delete trace links, and navigate to linked objects. For more information, see [Creating Trace Links](#), [Linking to an Object in Another Teamcenter Product](#), [Viewing Object Relationships](#), [Deleting Trace Links for an Object](#), and [Navigating to a Linked Object](#) in chapter 7, *Showing Object Relationships With Trace Links*.

- Change the values of editable properties, if the shortcut and the referenced object reside in the same project. You cannot edit properties for a referenced object in a different project. For more information, see [Properties Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*, and [Editing the Properties of a Selected Object, Editing Properties in Table View Cells](#), or [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

Shortcuts cannot be created for groups, notes, trace links, or diagrams. However, those object types can be added to groups as members. For more information, see [Adding Group Members](#), earlier in this chapter.

Creating a Shortcut



- You must have **Modify** permission for the intended destination.
 - To create a shortcut to a building block, a data dictionary, a data definition, a TRAM, or a diagram, you must have **Architect** privilege for the project.
1. Select the master object, pull down the **Edit** menu, and choose **Copy**.
 2. In the navigation tree, the content table, or the **Links** tab or window, select the destination object for the shortcut.
 3. Do one of the following:
 - In the navigation tree or the **Links** tab or window, pull down the **Edit** menu and choose **Paste Shortcut**.
 - In the content table, pull down the **Edit** menu and then do one of the following:
 - o To display the master object's children below the new shortcut, choose the **Paste Shortcut**→**Show Children** options.
 - o To keep the master object's children undisplayed, choose the **Paste Shortcut**→**Hide Children** options.



The current display state of the children is effective when the shortcut is replaced and when it is exported to Microsoft Office Word, to Microsoft Office Excel, or to an XML data file. You can change the display. For more information, see [Hiding and Showing the Children of a Master Object](#), [Replacing Shortcuts With Copies of the Master Objects](#), [Exporting Objects to Microsoft Office Word](#), [Exporting Objects to Microsoft Office Excel](#), and [Exporting Objects to an Architect/Requirements XML File](#), later in this chapter.

Procedure Notes

Step 1: You can also right-click the master object and choose **Copy** from the pop-up menu. Or, click the **Copy** button on the toolbar or press control-C.

Step 3: You can also right-click the destination object and choose **Paste Shortcut** from the pop-up menu. Or, press control-S.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Navigating From a Shortcut to the Master Object

1. Select the shortcut.
2. Pull down the **View** menu and choose the **Go To→Go To Object** options.

The master object is highlighted in the content table, with the **Address** bar showing the project where the object resides.

Procedure Notes

Step 2: You can also right-click the shortcut and choose the **Go To→Go To Object** options from the pop-up menu.

Hiding and Showing the Children of a Master Object



The children below a shortcut are also shortcut objects. The actual children reside with the master object.

1. In the content table, select the shortcut.
2. Do one of the following:
 - In the **Properties** tab or window or the Edit Properties dialog window, do the following:
 - . Double-click the **Shortcut Options** property value.
 - . In the Multi-Choice dialog window:
 - To display the master object's children below the shortcut, choose **ShowChildren**.
 - To hide the master object's children, choose **HideChildren**.
 - Pull down the **Edit** menu, and then do one of the following:
 - . To display the master object's children below the shortcut, choose the **Paste Shortcut→Show Children** options.
 - . To hide the master object's children, choose the **Paste Shortcut→Hide Children** options.



The current display state of the children is effective when the shortcut is replaced and when it is exported to Microsoft Office Word, to Microsoft Office Excel, or to an XML data file. For more information, see [Replacing Shortcuts With Copies of the Master Objects](#), [Exporting Objects to Microsoft Office Word](#), [Exporting Objects to Microsoft Office Excel](#), and [Exporting Objects to an Architect/Requirements XML File](#), later in this chapter.

Replacing Shortcuts With Copies of the Master Objects



Before you replace a shortcut, set the shortcut to hide or show the master object's children as you prefer. The master object copy retains the shortcut's last setting. For more information, see [Hiding and Showing the Children of a Master Object](#), earlier in this chapter.



- You must have **Architect** privilege to replace a shortcut to a building block, a data dictionary, a data definition, a TRAM, or a diagram.
 - To replace a shortcut to other object types, you must have **Read and Write** access privilege for the project that contains the master object.
 - The master objects can reside in this project or in other projects.
1. In the content table at the top level of the containing folder, select the shortcuts.



You cannot replace shortcuts at lower levels or children of any shortcuts.

You can select one shortcut or a group of adjacent or adjoining shortcuts.

2. Pull down the **Edit** menu and choose **Uncouple Shortcut**.

The shortcuts are deleted and copies of the master objects are created in the same location.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Uncouple Shortcut**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Sending Object Data by E-Mail

Directly from the Architect/Requirements client, you can send E-mail containing data for the objects in any project to which you have access. For each object that you select, Architect/Requirements extracts data from the database to an E-mail message, independent of the E-mail program on your computer.

You can send the message to any number of recipients. For internal recipients, the users and user groups in the project, you can address the message automatically from a list maintained by your project administrator. As in the Microsoft Outlook Address Book, for example, you can select users individually, and you can select user groups for distribution to many users simultaneously. For external recipients, those without access to the project, you enter E-mail addresses manually.



Also from the Architect/Requirements client, you can send E-mail that does not contain data for selected objects. For example, you may want to communicate with external vendors who do not have project access. Or, you may want to notify an internal user group of a project meeting.

You can send data for folders, requirements, building blocks, groups, notes, and diagrams. These object types and their subtypes can be selected in any combination and in any number. In addition, you can select group members, object shortcuts, and linked objects. For more information, see [Using Groups to Maintain Objects](#) and [Working With Shortcuts](#), earlier in this chapter, and [Viewing Object Relationships](#) in chapter 7, *Showing Object Relationships With Trace Links*.

Architect/Requirements can generate the data in the MHTML format, the HTML format, or the plain text format, according to your choice. Or, you can send the URLs of the selected objects as hyperlinks, which internal recipients can click to start Architect/Requirements and to navigate to the objects.

You can select the objects in the hierarchical content table, the **Attachments** tab or floating window, the **Links** tab or window, the **Where Used** tab or window, or the **Versions** tab or window. For more information, see [Hierarchical Content Table](#), [Attachments Tab](#), [Links Tab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



Data is sent only for the selected objects. The message does not include data for unselected objects at levels below a selected folder, requirement, building block, group, or shortcut.

To send object data by E-mail:

1. Select each object for which you want to send data.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

- 2.

Pull down the **Tools** menu and choose **Send Email**.

The Send Email dialog window is displayed, with the name of each selected object in the **Object(s)** field.



In a tab window, right-click the selection and choose **Send Email** from the pop-up menu. The **Tools** menu is not available in tab windows.

- 3.

Address the recipients by doing one or both of the following:

- For project users and user groups:
 - . Click the **To** button to display the Select User dialog window.
 - . Check the check box for each recipient user and user group, and then click **OK** to close this dialog window and add the names in the **To** field.
- For external recipients, enter the E-mail addresses directly in the **To** field.

In the remaining fields, you can do any or all of the following:

- In the **Subject** field, enter a general description of the message.
- In the **Send As** field, select one of the following data formats:
 - . Select **Web Page including graphics (MHTML)** to send the data in an **.mhtml** file, as an attachment containing all data specified by the document template named in the **Format Using** field. In that field, you can select another document template to specify different data.

The document template also determines the style sheet that is applied to the data. For more information, see [Exporting Objects to Microsoft Office Word](#), later in this chapter. If you have questions about document templates, consult your project administrator.

- . Select **Plain Text** to send the data as unformatted text in the message body.
 - . Select **URL** to send each object's URL as a hyperlink in the message body.
 - . Select **Web Page without graphics (HTML) as attached file** to send the data in an **.html** file, as an attachment containing only text and tables.
 - . Select **Web Page without graphics (HTML) embedded in message** to send the data as HTML content in the message body, with text and tables only.
- In the **Message** field, enter the text of the message body.
4. Click **OK** to close the Send Email dialog window and send the message.

A confirmation message is displayed, stating that the message has been sent.

Procedure Notes

Step 1: To see lower level objects in a hierarchical view, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click the object and choose **Expand All** from the pop-up menu.

Step 3: You can also enter E-mail addresses for project users and user groups directly in the **To** field.

Exporting Objects

The following sections describe procedures for exporting objects to AP233 STP and XML files, to Architect/Requirements XML data file, and to Microsoft Excel.

Exporting Objects to an AP233 STP File

From any project where you have Read and Write access, you can export objects to an AP233 STP file generated by Architect/Requirements. The file is saved in the drive or folder that you specify. This data can be imported from the file to a new or existing project in the Systems Engineering and Requirements Management module, where Architect/Requirements uses the data to reproduce the objects. In addition, the file can be imported to update property values for existing objects. For more information, see [Importing Objects From an AP233 STP File](#), later in this chapter, and [Updating Properties From an AP 233 STP File](#) in chapter 9, *Working With Object Properties*.



The AP233 export feature is for exchanging requirements data with other products that support the AP233 standard. This feature is not suitable for archiving data or moving data from one Architect/Requirements project or installation to another. To archive or move data, export the objects using the Architect/Requirements XML format. For more information, see [Exporting Objects to an Architect/Requirements XML File](#), later in this chapter.

The export file carries the **.stp** file name extension. The data is exported in the format defined by AP233, the STEP² application protocol for systems engineering data representation. In Architect/Requirements, the data represents folders and requirements. You can export all such objects in a selected project or a folder. The objects are exported with complete data for the following:

- All applicable user-defined properties, and the following system-defined properties:
 - **Name**
 - **Text**
 - **ROIN**
 - **Number**
 - **Source Paragraph**
 - **Source Filename**
 - **Baseline**
 - **Version Type**
 - **Security Profile**
 - **Subtype**

- Parent, child, and sibling relationships among the objects.
- Notes attached to the objects.
- Trace links for requirements, and notes.

If a starting or ending object is not present in the exported set of objects, then the trace link is neither exported from nor imported back to Architect/Requirements.

- Versions and variants of requirements.

² Standard for the Exchange of Product Model Data (ISO 10303).



Exporting building blocks, groups, diagrams, graphics, and OLE objects to **.stp** file is not supported. However, these objects can be exported to XML data files.

To export objects to an AP233 STP file:

1. Do one of the following:
 - For all folders and requirements in the project, select the project node in the navigation tree.
 - For all folders and requirements in a folder, select the folder in the navigation tree or in the hierarchical content table.

2. Pull down the **File** menu and choose the **Export**→**AP233 (.stp)** options.

Architect/Requirements displays the Save dialog window.

3. In the **File name** field, enter the name of the export file.

The **Save in** field displays the current drive or folder. You can save the export file in this drive or folder, or you can use this field to change the file location.

4. Click **Save**, or press the enter key.

The dialog window closes, and a message states that the export is in progress.



Although the export file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the export file.

Procedure Notes

Step 1: To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the project node or folder and choose the **Export**→**AP233 (.stp)** options from the pop-up menu.

Step 3: Architect/Requirements automatically adds the **.stp** file name extension to the file name that you enter. You can also select an existing **.stp** file in the list to fill in the **File name** field automatically and overwrite that file.

Exporting Objects to an AP233 XML File

From any project where you have Read and Write access, you can export objects to an AP233 XML file generated by Architect/Requirements. The file is saved in the drive or folder that you specify. This data can be imported from the file to a new or existing project in the Systems Engineering and Requirements Management module, where Architect/Requirements uses the data to reproduce the objects. In addition, the file can be imported to update property values for existing objects.



When you export objects from a folder, and create new objects and relations, importing from the exported AP233 XML file to the same folder does not delete the objects or relations created after the export. The import operation only updates the existing objects and creates new objects if the importing objects do not exist in the selected folder.

When a set of exported objects in a folder contains both a requirement and its versioned object, the newly created version of the requirement is not updated upon importing the set of objects in to the same folder. However, a new requirement object is created.



The AP233 export feature is useful for exchanging requirements data with other products that support the AP233 Part 28 standard. This feature is not suitable for archiving data or moving data from one Architect/Requirements project or installation to another. To archive or move data, export the objects using the Architect/Requirements XML format. For more information, see [Exporting Objects to an Architect/Requirements XML File](#), later in this chapter.

In Architect/Requirements, the data represents folders and requirements. You can export all such objects in a selected project or a folder. The objects are exported with complete data for the following:

- All applicable user-defined properties, and the following system-defined properties:
 - **Name**
 - **Text**
 - **ROIN**
 - **Number**
 - **Subtype**

- Building blocks.
- Parent, child, and sibling relationships among the objects.
- Notes attached to requirements, building blocks, folders, and ports.
- Trace links for requirements and notes.

If a starting or ending object is not present in the exported set of objects, then the trace link is neither exported from nor imported back to Architect/Requirements.

- Graphics and OLE objects in requirements.

Some of the Architect/Requirements concepts are not defined in the AP233 standard. Therefore, the export functionality is limited as specified below:

- Groups, diagrams, spreadsheets, change logs, Matlab files, and trace links involving building blocks are not exported to or imported from AP233 XML files.
- Notes on trace links and connections are not exported.

- Proxy links (such as wolf links and advanced links between Architect/Requirements objects and Enterprise Knowledge Management objects) that are created for Architect/Requirements objects are not exported.
- Only the currently effective version of the objects are exported.
- The **Document Options** property of the requirement objects is not exported.

Exporting objects that have **Document Options** set and importing them back may have undesired behavior. For example, consider an object that has **Document Options** set. If it is exported and imported at the same location, the Number property may be changed incorrectly. If the same object is imported in a new folder, then the object is created without having **Document Option** set.

- Shortcut objects are exported as objects that are referenced by the respective shortcuts, and not as shortcuts.

When you export a shortcut and import it back in the same or a different folder, a new object is created. The new object's type and property values are the same as that of the object referenced by the exported shortcut.

- Objects in the Administration module and information related to baselines are not exported.
- The **Security profile** property of the objects is not exported.

To export objects to an AP233 XML file:

1. Do one of the following:
 - For all folders and requirements in the project, select the project node in the navigation tree.
 - For all folders and requirements in a folder, select the folder in the navigation tree or in the hierarchical content table.

2. Pull down the **File** menu and choose the **Export**→**AP233 (.xml)** option.

Architect/Requirements displays the Save dialog window.

3. In the **File name** field, enter the name of the export file.

The **Save in** field displays the current drive or folder. You can save the export file in this drive or folder, or you can use this field to change the file location.

4. Click **Save**, or press the enter key.

The dialog window closes, and a message states that the export is in progress.



Although the export file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the export file.

Procedure Notes

Step 1: To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the project node or folder and choose the **Export**→**AP233 (.xml)** options from the pop-up menu.

Step 3: Architect/Requirements automatically adds the **.xml** file name extension to the file name that you enter. You can also select an existing **.xml** file in the list to fill in the **File name** field automatically and overwrite that file.

Exporting Objects to an Architect/Requirements XML File

From any project where you have Read and Write access, you can export objects to an XML data file in a format defined specifically for Architect/Requirements. The content of this file is more complete than the AP233 XML file and is well suited for archiving Architect/Requirements projects or moving Architect/Requirements content from one project or installation to another.

The file is saved in the drive or folder that you specify. The data can be imported to a project node or folder in the Systems Engineering and Requirements Management module, where Architect/Requirements uses the data to reproduce the objects. Thus, objects can be shared among different projects, in the same Architect/Requirements installation and in installations at other sites. For more information, see [Importing Objects From an Architect/Requirements XML File](#), later in this chapter.

Data is exported for all of the objects in the folder selected in the navigation tree or the hierarchical content table. The data in an Architect/Requirements XML file may represent folders, requirements, building blocks, notes, diagrams, and versions and variants of requirements and building blocks. The objects are exported with complete data for system defined and user defined properties, and for parent, child, and sibling relationships among the objects.

Data is included for trace links between objects within the folder. Inter-folder and inter-project trace links are not exported.



- When the data is imported to a different project, some data may be ignored if the target project schema is not compatible with the originating project schema.
- Versions and variants are preserved while exporting or importing a project. However, when exporting at the folder level, only the currently effective versions are exported. Hence the version tree is not re-established during import. Similarly, the variants are marked as variants at import, but they are not associated with any version tree or master object, as that version may or may not be part of the export.

To export objects to Architect/Requirements XML data file:

1. Select the folder in the navigation tree or in the hierarchical content table.
2. Pull down the **File** menu and choose the **Export**→**XML** options.

Architect/Requirements displays the Save dialog window.

3. In the **File name** field, enter the name of the export file.

The **Save in** field displays the current drive or folder. You can save the export file in this drive or folder, or you can use this field to change the file location.

4. Click **Save**, or press the enter key.

The dialog window closes, and a message indicates that the export is in progress. When the export is complete, a confirmation message is displayed.



Although the export file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the export file.

Procedure Notes

Step 1: To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the folder and choose the **Export**→**XML** options from the pop-up menu.

Step 3: Architect/Requirements automatically adds the **.xml** file name extension to the file name that you enter. You can also select an existing **.xml** file in the list to fill in the **File name** field automatically and overwrite that file.

Exporting Objects to Microsoft Office Excel

You can export data to a Microsoft Office Excel workbook for objects in the following Architect/Requirements client views:

- The hierarchical content table and your Recycle Bin. For more information, see [Hierarchical Content Table](#) and [Architect/Requirements Recycle Bin](#) in chapter 3, *Using the Architect/Requirements Main Window*.
- The **Properties**, **Attachments**, **Links**, **Where Used**, and **Versions** tabs and floating windows. For more information, see [Properties Tab](#), [Attachments Tab](#), [Where Used Tab](#), [Links Tab](#), [Versions Tab](#), and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

From any of those client views, you specify the objects that you want to export. You can export only the currently selected objects, or you can export all displayed objects whether or not they are selected.

To specify the properties to export, you can use one of the following:

- The properties that are currently displayed in the client view.
- An Excel template, containing property columns, tags for property values, and other information. For more information about Excel templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about Excel templates, consult your project administrator.
- A saved view of property columns. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*. If you have questions about views, consult your project administrator.

In addition, you can choose the state in which you want to create the workbook. You can export the objects and properties to a workbook that is not connected to the Architect/Requirements database. Or, you can create a new Excel Live workbook, through which you can interactively edit the properties of exported objects and see your changes applied in the Architect/Requirements client. You can import the object data from either workbook to create new objects automatically. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*, and [Importing Objects From Microsoft Office Excel](#), later in this chapter.

Architect/Requirements extracts the data to a read-only Excel workbook. Although this file is temporary, you can use all of Excel's viewing, printing, and navigation features. From within this file, you can send the data to E-mail recipients, such as colleagues, customers, and suppliers. In both the E-mail message and the file itself, you can click the object type indicators to navigate to those objects in the client.



If you sort the data in Excel, the object type indicators and the objects may be mismatched. This condition occurs in Excel when a cell contains a graphic that is taller than the row height. For a row height that accommodates the indicators, increase the size of Excel's standard font.

The effective size depends on the standard font that you use. With Arial, for example, a size of 11 points allows the indicators to be sorted with the correct objects. To increase the size, Click Excel's **Office Button**, choose **Excel Options**, and select the new size on the **Popular**→**When creating new workbooks** section.

If you want to make this change, do so before you export. Otherwise, you must exit and restart Excel for the change to take effect.

The temporary file is deleted from your computer when you exit Architect/Requirements. While this file is open, however, you can create a permanent version that is independent of Architect/Requirements.

You can store the permanent file for reference, as a record of the view at a certain time, for example, or as a comparison with the corresponding view in another project or folder. Because you assign the file name, file type, and location, you can easily retrieve the exported view. You can click an object type indicator in the file to start Architect/Requirements and automatically navigate to the object in the client.

The view in the content table is set by your selection in the navigation tree:

- With a project node selected, the content table shows the folders at the project's primary level.
- With a folder selected, the content table shows the folders, requirements, building blocks, and groups at the folder's top level.
- With the Recycle Bin selected, the content table shows the objects that you deleted from all projects to which you have access.

In the notebook pane, you set the views by selecting an object in the navigation tree or the content table and clicking the **Properties**, **Attachments**, **Links**, **Where Used**, or **Versions** tab. You can then open tab windows to set views for objects selected in the notebook pane.

While exporting the content to Excel using a view, Architect/Requirements reads the locale-specific styles for date from a CSS (for example, **DateStyle_en_US.css**) file. There should be a **DateStyle_languageCode_countryCode.css** file present for each locale. If the date styles for the client locale are not found, then the user is prompted to confirm before exporting the file using English (US) date styles.

Before exporting a table, consider the information that you want to see in Excel. For example, you can add columns to export other properties, and you can remove displayed columns to exclude those properties from the file. To adjust for more or fewer properties, you may want to rearrange or resize columns, or sort the information by a different column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.



Microsoft Office Excel 2013 or Excel 2016 must be installed on your computer.

To export objects to Microsoft Office Excel:

1. To activate the view, do one of the following:
 - For the hierarchical content table, pull down the **View** menu and choose the **Go To**→**Content Table** options, or click inside the table.

- For the **Properties** tab, pull down the **View** menu and choose the **Go To→Notebook Pane** options, or click inside the table.
- For the Recycle Bin, the **Attachments** tab, the **Links** tab, the **Where Used** tab, or the **Versions** tab, click inside the table.

You can open the floating window for an activated tab by clicking the **Open tab** button on the notebook pane's toolbar.

2.

Pull down the **File** menu and choose the **Export→Excel Spreadsheet** options to display the Export To Excel dialog window.

You can also right-click inside the view and choose the **Export→Excel Spreadsheet** options from the pop-up menu.



In a tab window, right-click inside the table and choose the **Export→Excel Spreadsheet** options from the pop-up menu. The **File** menu is not available in tab windows.

3. Under **Object Selection**, do one of the following:

- To export only the selected objects, click **Export Selected Objects**.
- To export all objects in the view, click **Export All Objects in View**.

4. Under **Formatting**, do one of the following:

- To export the properties in the currently displayed columns, click **Use Current View Columns**.
- To export the properties specified in an Excel template or in a saved view of property columns:
 - Click **Use Excel Template or View**.

Below this button, the list becomes available. The list shows the Excel templates and the public views for the project, and includes the private and pending views for your user name.

- In the list, select the Excel template or the view.



- The Excel template may contain multiple worksheets, and each worksheet may specify different properties and rules. Therefore, data may be separated on worksheets in the export file.
- If the Excel template or the view contains properties that do not apply to an object type specified for export, the inapplicable values are indicated in the related cells.

5. Under **Output**, do one of the following:

- To create a workbook that is not connected to the database, click **Static Snapshot**.
- To create an interactive live Excel workbook that is connected to the database, click **Excel Live**.

6. Click **OK** to close the dialog window.

Excel opens a read-only file containing the specified rows and columns. This file is temporary and is deleted from your computer when you exit Architect/Requirements.

You can create a permanent file by pulling down Excel's **File** menu and choosing **Save As**. In the Save As dialog window, assign the file name, file type, and location outside the Architect/Requirements database, for example, on a local drive. This file remains on your computer when you exit Architect/Requirements.



- If you save a live Excel file, Siemens PLM Software recommends that you assign it the **.xlsm** file type. Only the **.xlsm** file type supports the full capability of live Excel.
- If you save a live Excel file as the **.mhtml** file type, it becomes static and loses the live capability permanently.

Exporting Objects to Microsoft Office Word

Through the Architect/Requirements interface with Microsoft Office Word, you can capture current information from the database for analysis and distribution. For each object that you specify, Architect/Requirements extracts data to a Word document, using a *document template* as a reference. In turn, the document template controls the export process through the following:



An *object template*, which determines the data that is exported. Tags in the object template represent certain object properties, whose values are extracted from the database to the export document for each specified object. By default, one of two object templates is assigned to each object type, and to the system-defined subtypes, **Document** for folders and **Paragraph** for requirements:

- o For requirements, paragraphs, notes, and diagrams, the default object template contains the **Name** property, in the template's heading, and the **HTML** property, in a paragraph below the heading. With this template, each object's name is extracted to a heading in the export document, and the full content, including tables and graphics, immediately follows in the body.

The **HTML** property is not displayed in the Systems Engineering and Requirements Management module because the value can exceed the effective size for table cells. However, you can see an object's full content in the **Preview** tab or floating window. For more information, see [Preview Tab](#) or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

- o For other object types, the default object template contains only the **Name** property in the heading. With this template, each object's name is extracted to a heading in the export document.
- A *style sheet*, which determines the Word formatting styles that are applied to the data in the export document.

The document template may be customized to export additional data and apply different styles. In the default object template, for example, your project administrator may add the **ROIN** property to export the Requirement Object Identification Number (ROIN) of each requirement and paragraph. Your project administrator may create style sheets to apply custom Word styles to the exported data for particular

object types. Furthermore, a unique object template may be assigned to each type definition, to export different data for that object type. For more information about templates and style sheets, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about templates or style sheets, consult your project administrator.

By the way in which you specify the objects, you set the context of the data in the export document. The context can range from broad to narrow. For example:

- To specify all requirements in a folder, select only the containing folder. The document includes data for all direct children and lower level descendants of each parent requirement, but not for the folder itself. In this context, you see the data according to the organization of requirements within the entire folder.

For each requirement at the top level, the data from the object template heading is formatted in Word's Heading 1 style. For each child and descendant, that data is formatted in a Word heading style that matches the object's level in the hierarchy. Headings appear in ascending order of the **Number** property values.



If the selected folder also contains groups or other folders, data is not exported for those objects. Data is not exported for requirements in folders within the selected folder.

- To specify one requirement, its direct children, and all lower level descendants, select only the parent requirement. In this context, you see the data according to the relationships in the entire hierarchy below the parent.

For the parent, the data from the object template heading is formatted in Word's Heading 1 style. For the direct children, that data is formatted in Word's Heading 2 style, with data for lower level descendants in heading styles that match those levels. Headings appear in ascending order of the **Number** property values.

- To specify all members of a group, select only that group. In this context, you see the data for all of the group's members, the objects shown at the level directly below the group, but not for the group itself.

For each member, the data from the object template heading is formatted in Word's Heading 1 style. Headings appear in the order in which the members are found in the database.



Data is not exported for objects that are shown at lower levels below the members. For more information, see [Adding Group Members](#) in chapter 4, *Maintaining a Project*.

- To specify one or more objects, of mixed types or of the same type, select the individual objects while holding down the control or shift key. In this context, you see data only for the selected objects, regardless of any hierarchical relationships to other objects.

For each object, the data from the object template heading is formatted in Word's Heading 1 style. Headings appear in the order in which the objects are displayed on the screen, regardless of the order in which the objects are selected.



When multiple objects are specified, data is not exported for unselected objects below a selected folder, requirement, building block, or group.



When objects with embedded OLE content are exported to Microsoft Office Word, none of the OLE content is present in the Word document. The OLE objects are exported with a preview

image representing the OLE content. Objects retain their OLE content when opened for edit individually.

The views where you select the objects depend on the objects that you want to specify:

- For all requirements in a folder, select the containing folder in the navigation tree, the hierarchical content table, or the **Links** tab or window.
- For individual objects, select them in the hierarchical content table, the **Links** tab or window, or the **Versions** tab or window. In each view, you can select one object or multiple nonadjacent and adjoining objects. In the hierarchical content table, you can select objects in different folders. In the **Links** tab or window, you can select objects in either pane but not in both panes at the same time.



A dash before a level in a **Number** value, for example, **1.-1**, indicates that the requirement is an unnumbered part of the parent paragraph at the next higher level. The requirement's **Document Options** property value is **Disable Numbering**. Such a requirement is not assigned a heading or an outline level in the export document.

The document template must now reside on the folder prior to export.

To add the document template to the folder:

1. In the **Notebook** pane, select **Properties**.
2. Scroll down to find **Document Template** in the list of properties.
3. Double-click the empty values cell next to **Document Template**.
The **Document Template** dialog box is displayed.
4. Select the document template that you want as the default template for the folder.
5. Click **OK**.

The following are guidelines for exporting Microsoft Word 2013 or Microsoft Word 2016 documents:

- Document templates must be used for successfully exporting documents from Microsoft Word. Users must create a custom document template, a custom style sheet, custom text object templates, and object templates. Assign this document template to a folder. To maintain consistency, users must import the styles into each associated component of the document template. Siemens PLM Software recommends using the Microsoft Word Document organizer to transfer the styles.
- The style sheet attached to the document template is applied when the document is exported and should not be changed at that time. Changing the style sheet at the time of export yields inconsistent results. For consistency in using the default styles, especially for the list styles such as bullets and numbers, you must allow the consistent formatting of the exported document. Custom styles work fine but you must create the styles properly using Microsoft Word.
- When exporting a document, the style sheet must be explicitly associated with the document template. If it is not, the style sheet is not applied successfully on exported document.
- You can also export macros when exporting objects to Word. However, you must add the macros to the Architect/Requirements Style Sheet that is used if you want the macros to be available after export. After the export, the macros can be triggered using VBA events.

- You cannot include the Diagram Images directly when exporting to Word. You must first symbolically reference it into a requirement or note. To include the diagram image in a Word export of the building block structure, create a note on the building block and reference Diagram Image in that note. For information on including notes in Word exports see *Object Templates* in the *Systems Architect/Requirements Management Project Administrator's Manual*.

To export objects to Microsoft Office Word:



Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

1.

Specify the objects as described above, and then pull down the **File** menu and choose the **Export**→**Word Document** options.

The Select Document Template dialog window is displayed. In the **Select a Document Template** field, you can change the selection for this document.

2. Do one of the following:

- To use the selected document template for all objects, click **OK** to close the Select Document Template dialog window and generate the document.
- To change the stylesheet for the document or the object template for one or more types, click **More** to display the Document Template dialog window.

. Do one or both of the following:

- o To change the Word styles that are applied to all data in the document, select the style sheet in the **Select Stylesheet** field.
- o To change the object template for a type in the **SubTypes** column:

a.

Double-click the cell in the **OverrideObjectTemplate** column to display the **Single-Choice** dialog window.

- a. Select the check box for the object template that you want to use, and then click **OK** to close this dialog window.

Repeat these steps for each additional type whose object template you want to change.



If you want to apply a numbered or bulleted list style, create an appropriate list style and apply the list style. You must also add the list style to the stylesheet.

There are out of the box Microsoft Word list styles can be added to the stylesheet such as List Bullet.

. Click **Export** to close the Document Template dialog window and generate the document.

A Word file opens (**.mhtml**), with data for the exported objects. You can view and print the document and send it by E-mail and fax.



This file is temporary and is deleted from your computer when you exit Architect/Requirements.

You can create a permanent document by clicking Word's **Office Button** and choosing **Save As→Word Document**. In the **Save As** dialog window, assign the file name, file type, and location. This document can serve as a record for comparison with future changes. Or, the document can be imported to create new requirements in the same project or another project, and content can be edited independently of Architect/Requirements.

Procedure Notes

Step 1: You can also right-click the selection and choose the **Export→Word Document** options from the pop-up menu.

OLE objects embedded in requirement content are not exported to Microsoft Office Word. This limitation includes the first OLE object encountered. However, the first object was exported in earlier Architect/Requirements versions.

The graphics for the OLE objects are extracted to the Word export document. However, double-clicking the graphic does not launch the OLE application.

This change was made because the embedded OLE objects sometimes became associated with the wrong graphic, which caused the graphics to behave incorrectly when double-clicked. Also, incorrect graphics appeared when the document was printed. Only the first OLE object could be exported, which led to inconsistent behavior.

To workaround or avoid:

You can use Tool Command Language (Tcl) to include the first OLE object in requirement content that is exported to Word. For example:

- To specify which document template to use in the export, enter:

```
set documentTemplate ""
```

Empty double quotation marks ("") specify the default document template. Or, enter the name of another document template in the quotation marks.

- To export the selected objects to a Word file on the server, including embedded OLE objects, enter:

```
set includeOle true
```

```
set file [exportDocument $selected MS_WORD $documentTemplate true $includeOle]
```

- To download the file to the client and open it in Word, enter:

```
createAction FileDownload [list $file MS_WORD]
```

Exporting Objects for importing into Teamcenter

Architect/Requirements allows you to export objects for importing into Teamcenter. The objects are exported in XML format.



You must have access to the Architect/Requirements server to access the exported file.

You can export the following objects:

- Requirements and paragraphs including their rich text
- Folders
- Document Folders

These map to requirement specifications in Teamcenter.

- Trace links
- Notes including rich text
- Groups



When exporting a portion of a project, references to project objects outside the exported region are not resolved. Objects referenced by a trace link, shortcut, or group membership are not exported. You must export at the project level to ensure that all intraproject relationships are preserved.

To export objects for Teamcenter:

1. Select the project or objects that you want to export.
2. Pull down the **File** menu and choose **Export**→**TC XML**.

Architect/Requirements displays a message confirming the export. It also displays the location of the exported file.

The exported objects are saved on the Architect/Requirements server in XML format. The naming convention that Architect/Requirements follows for the exported file is **TcSE_to_TC_Export-nnnnnnn.xml** where **nnnnnnn** is a number that increments with each export.

The schema for the XML file is written to **TcSE_to_TC_Schema-nnnnnnn.xsd** file. The schema includes user defined subtypes and properties. This file is used for mapping the Architect/Requirements type and property names to the corresponding Teamcenter names.

A folder named **TcSE_to_TC_Export-nnnnnnn.xml** is also created. It contains the **docm** files of the exported requirements and paragraphs.

You must process the exported XML file to convert it to TC XML format. The TC XML file can then be imported into Teamcenter.

The **TcSE_to_TC_Text-nnnnnnn.xml** file is used for importing rich text into Teamcenter.

For information about importing to Teamcenter, see the topic *Migrate from standalone Systems Engineering and Requirements Management* in Appendix A of the *Getting Started with Systems Engineering Guide* Teamcenter manual set.

For information on controlling what is exported, see *Customizing exports for migration to Teamcenter* in the *Systems Architect/Requirements Management API Reference*.

Importing Objects

The following sections describe procedures for importing objects from AP233 STP and XML files, from Architect/Requirements XML data file, and from Microsoft Excel.



If you encounter an out of memory error when importing a large document, you may need to increase the maximum memory available for the Architect/Requirements client.

For information on increasing the available memory, see [Appendix D - Changing the maximum memory available for rich client](#).

Importing Objects From an AP233 STP File

To create objects automatically in the Systems Engineering module, you can import object data from files that carry the **.stp** file name extension. This data is in the format defined by AP233, the STEP³ application protocol for systems engineering data representation. In Architect/Requirements, the data represents folders and requirements that were previously exported to the file. Architect/Requirements uses the data in the file to reproduce the objects. You can also use the import file to update property values for existing objects. For more information, see [Exporting Objects to an AP233 STP File](#), earlier in this chapter, and [Updating Properties From an AP 233 STP File](#) in chapter 9, *Working With Object Properties*.



This function is for exchanging data with other products that support the AP233 standard. This procedure is not for archiving data or moving data between Architect/Requirements projects or installations. To archive or move data, import the objects using the Architect/Requirements XML format. For more information, see [Importing Objects From an Architect/Requirements XML File](#), later in this chapter.

Points to Note

- Before importing, ensure that you know the name and location of the **.stp** file containing the object data that you want.
- Although the import file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the import file.
- You must have **Modify** permission for the project or folder where you intend to import the objects.
- Importing building blocks, groups, diagrams, graphics, and OLE objects from an **.stp** file is not supported. However, these objects can be imported from XML data files.
- Date properties with blank values receive the value **TBD** when imported.

For a new or existing project, you can import the objects to the primary level, directly below the project node, or to the top level of any folder in the project. The imported objects are complete with the following:

- Applicable user-defined properties, and the following system-defined properties:
 - **Name**

³ Standard for the Exchange of Product Model Data (ISO 10303).

- o **Text**
 - o **ROIN**
 - o **Number**
 - o **Source Paragraph**
 - o **Source Filename**
 - o **Baseline**
 - o **Version Type**
 - o **Security Profile**
 - o **Subtype**
- Parent, child, and sibling relationships among imported objects.
 - Notes attached to the objects.
 - Trace links for requirements and notes.

If a starting or ending object is not present in the exported set of objects, then the trace link is neither exported from nor imported back to Architect/Requirements.

- Versions and variants of requirements.

Variants that are exported will be imported as new objects in Architect/Requirements.

To import objects from an AP233 STP file:

1. Do one of the following:
 - For a project's primary level, select the project node in the navigation tree.
 - For a folder's top level, select the folder in the navigation tree or in the hierarchical content table.



Siemens PLM Software recommends that you import the objects to an empty folder.

2. Pull down the **File** menu and choose the **Import**→**AP233 (.stp)** options.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder. The **Look in** field displays the current drive or folder. If the import file is not in the list, you can use this field to change the drive or folder.

- 3.

Select the import file in the list, and then click **Open**.

The Import AP233 dialog window is displayed.

4. Select **Create New Objects**, and then click **OK**. You can also select **Update Existing Objects** import property values from an AP 233 **.stp** file that contains records for objects in the database. For more information see chapter 9, [Updating Properties From an AP 233 STP File](#).



This action clears the queue for the **Undo** option and cannot be reversed.

The Import AP233 dialog window closes, and a message is displayed to indicate that the import is in progress. When the import is complete, a confirmation message is displayed.

Procedure Notes

Step 1: To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the import folder or the project node and choose the options from the pop-up menu.

Importing Objects From an AP233 XML File

When you import from an AP233 XML file, the import process updates objects if they already exist in Architect/Requirements. Otherwise, new objects are created. Architect/Requirements uses the data in the AP233 XML file to reproduce the objects.



When you export objects from a folder, and create new objects and relations, importing from the exported AP233 XML file to the same folder does not delete the objects or relations created after the export. The import operation only updates the existing objects and creates new objects if the importing objects do not exist in the selected folder.

When a set of exported objects in a folder contains both a requirement and its versioned object, the newly created version of the requirement is not updated upon importing the set of objects in to the same folder. However, a new requirement object is created.



The import operation from an AP233 XML file is for exchanging data with other products that support the AP233 Part 28 standard. The import operation is not for archiving data or moving data between Architect/Requirements projects or installations. To archive or move data, import the objects using the Architect/Requirements XML format. For more information, see [Importing Objects From an Architect/Requirements XML File](#), later in this chapter.

Points to Note

- Before importing, ensure that you know the name and location of the **.xml** file containing the object data that you want.
- Although the import file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the import file.
- You must have **Modify** permission for the project or folder where you intend to import the objects.
- The Architect/Requirements concepts of groups, diagrams, spreadsheets, change logs, Matlab files, and trace links involving building blocks are not defined in the AP233 standard. Therefore, those objects are not exported to or imported from AP233 XML files.

For a new or an existing project, you can import the objects to the primary level, directly below the project node, or to the top level of any folder in the project. The imported objects are complete with the following:

- Applicable user-defined properties, and the following system-defined properties:
 - **Name**
 - **Text**
 - **ROIN**
 - **Number**
 - **Subtype**

- Building blocks.
- Parent and child relationships among imported objects.
- Notes attached to requirements, building blocks, folders, and ports.
- Trace links for requirements and notes.

Behavior After Importing Objects

- When the subtype property of an object is changed after the object is exported to an AP233 XML file, importing it into the same folder ignores the changed subtype property of the object.

For example, export a folder that contains a requirement R1 with R12 and R13 as child objects to an AP233 XML file. Change the subtype property of R13 to **Paragraph**. Select the same folder and import it from the exported AP233 XML file. R13 is shown as a paragraph, rather than as a requirement.

- If Architect/Requirements does not find the exact subtype for an object, the object is created as its parent type.
- If a starting or ending object is not present in the exported set of objects, then the trace link is neither exported from nor imported back to Architect/Requirements.
- Date properties with blank values receive the default value when imported.
- Shortcuts of objects are imported as new objects.

When you export a shortcut and import it back in the same or a different folder, a new object is created. The new object's type and property values are the same as that of the object referenced by the exported shortcut.

- Variants are imported as normal objects.

To import objects from an AP233 XML file:

1. Do one of the following:
 - For a project's primary level, select the project node in the navigation tree.
 - For a folder's top level, select the folder in the navigation tree or in the hierarchical content table.

Siemens PLM Software recommends that you import the objects to an empty folder.

2. Pull down the **File** menu and choose the **Import→AP233 (.xml)** option.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder. The **Look in** field displays the current drive or folder. If the import file is not in the list, you can use this field to change the drive or the folder.

3. Select the import file in the list, and then click **Open**.

Procedure Notes

Step 1: To see lower-level objects in the table of contents, click the plus sign. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the import folder or the project node and choose the options from the pop-up menu.

Importing Objects From an Architect/Requirements XML File

From any project where you have Read and Write access, you can export objects to and import from an XML data file in a format defined specifically for Architect/Requirements. The content of this file is more complete than the AP233 XML file and is well suited for archiving Architect/Requirements projects or moving Architect/Requirements content from one project or installation to another.

To create or update objects automatically in the Systems Engineering and Requirements Management module, you can import object data from Architect/Requirements XML files. The data represents all objects in a folder that was previously exported using the Architect/Requirements XML format.

For a new or existing project, you can import the objects to the primary level, directly below the project node, or to the top level of any existing folder. The imported objects are complete with the following:

- Properties and values, both system-defined and user-defined.
 -  Date properties with blank values receive the value **TBD** when imported.

- Parent, child, and sibling relationships among imported objects.
- Trace links between objects represented in the file.
- Versions and variants of requirements and building blocks.



- You must have **Modify** permission for the project or folder where you intend to import the objects.
- To import building blocks, TRAMs, or diagrams to an existing project, you must have **Architect** privilege for the project.
- Imported diagrams are not connected directly to the Architect/Requirements database as are diagrams created through live Visio.
- If the import file contains data for properties that do not exist in the project, that data is ignored.
- If the import file contains data for trace links that cross project boundaries, that data is ignored.
- If the import file contains data for an object subtype that does not exist in the project, an object of the built-in base type is created.



- Before importing, ensure that you know the name and location of the Architect/Requirements XML file containing the object data that you want.
- A file that was exported from the Administration module cannot be imported to the Architect/Requirements module.
- The file being imported must have been exported from either the same Architect/Requirements version or the prior major version. Exporting from a newer version and importing into an older version is not supported because it may not be possible to map new data into an older database.

- Although the import file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the import file.
- Versions and variants are preserved while exporting or importing a project. However, when exporting at the folder level, only the currently effective versions are exported. Hence the version tree is not re-established during import. Similarly, the variants are marked as variants at import, but they are not associated with any version tree or master object, as that version may or may not be part of the export.

To import objects from Architect/Requirements XML data file:

1. Do one of the following:

- For a project's primary level, select the project node in the navigation tree.
- For a folder's top level, select the folder in the navigation tree or in the hierarchical content table.



To import building blocks and their subtypes to the selected folder, you must have **Architect** privilege for the project. Otherwise, building blocks and subtypes in the import file are ignored.

2. Pull down the **File** menu and choose the **Import**→**XML** options.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder.

3. Select the import file in the list, or enter the name in the **File name** field.

The **Look in** field displays the current drive or folder. If the import file is not in the list, you can use this field to change the drive or folder.

4. Click **Open**, or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

The dialog window closes, and a message is displayed to indicate that the import is in progress. When the import is complete, a confirmation message is displayed.

Procedure Notes

Step 1: To see lower level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the import folder or the project node and choose the **Import**→**XML** options from the pop-up menu.

If object names or text properties include certain control characters when exported to XML, then the file is not imported correctly. This problem occurs only in special cases when you try to copy a file from separate applications into the Architect/Requirements edit fields for object name and text properties. It does not occur when entering text with the keyboard.

These control characters are mapped to the HTML **&#nn;** notation in the XML exported file. One specific case is the Vertical Tab character, which appears as ****.

To work around or avoid:

Load the XML file into any XML or plain text editor, and replace the offending **&#nn;** patterns with a space. You may also remove them, depending on the control character and the desired outcome. You must replace or remove all the five characters, beginning with the ampersand and ending with the semicolon.

Importing Objects From Microsoft Office Excel

To create objects automatically in the Systems Engineering and Requirements Management module, you can import object data from any Microsoft Excel workbook that meets the following conditions:

- The import file can have any of the Excel file types including: **.xls**, **.xlsm**, **.xlsx**.
- Each column represents a single object property in Architect/Requirements, and each row below the column headings represents a single object.
- The column headings in the first row exactly match the names of object properties in Architect/Requirements. For any heading that does not match a property name, the import wizard has the ability to create new object properties or there is the option to ignore them.
- Objects that define a relationship between two other objects cannot be created by the **Excel Import Wizard**. This includes trace links and connections. The reason is difficulty in accurately identifying the two objects to be related.

For example, the conditions may be met by a file containing previously exported data. For more information, see [Exporting Objects to Microsoft Office Excel](#), earlier in this chapter.

Only the first worksheet in the file is imported. Cell formatting is ignored. In the selected import folder, one object is created for each populated row in the file.

Properties of imported objects are subject to the following conditions:

- Editable property values are applied to the new objects as entered in the worksheet.
- Read-only properties take system-defined values automatically, regardless of the values entered in the worksheet.
- Default values are assigned to editable text and numeric properties whose format is incorrect in the worksheet.
- Date properties with blank values receive the value **TBD** when imported.
- Markup (for example, boldface and italics), fonts, bullets, and other formatting are not preserved when importing text properties or the full text of requirements or notes. The imported text is plain.
- The **Number** property is applied to new requirements and building blocks. However, the rows representing these objects in the worksheet must be in correct logical order with no gaps in the numbering sequence.
- The **Type Name** and **Subtype** properties are assigned to each new object.



- You must have **Modify** permission for the folder to which you intend to import the objects.

- To import building blocks, TRAMs, or diagrams, you must have **Architect** privilege for the project.



If you want to perform imports that need to create new properties or new choices on existing properties, you must have the Project Administrator or Power User privilege. If you want a Power User to be able to import Excel spreadsheets with additional columns or changed information, the Power User must be assigned permission to modify definitions and type definitions. Power Users can make additions or changes to spreadsheets within Architect/Requirements and they can export and import such spreadsheets.

- If an error occurs during the import, all objects imported before the error are retained.
- Microsoft Office Excel 2013 or Excel 2016 must be installed on your computer.

To import objects from Microsoft Office Excel:

1.

Select the folder under which you want to import objects from Excel. To do this, pull down the **File** menu, and choose the **Import**→**Excel Spreadsheet** options. Or, right-click the folder and choose the **Import**→**Excel Spreadsheet** options from the pop-up menu.

Architect/Requirements displays the **Open** dialog window, which lists existing folders and files in the current drive or folder. If the import file is not in the list, you can use the **Look in** field to change the drive or folder.

2. Select the import file in the list, or enter the name in the **File name** field.
3. Click **Open** or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

4. Excel opens the file, places it in the background and the **TcSE Excel Import Wizard** window opens. There are four separate folder tabs for this wizard.

- **Start** - This area has a message that states “Welcome to the Excel Import Wizard for Teamcenter Systems Engineering. This wizard will allow you to import an Excel spreadsheet to a TcSE database.” Click **Next** to continue working with the wizard.
- **Default Type** - This is the area where you select how you want the information to import into Architect/Requirements from Excel. Your choices for the default type/subtype are: Requirement, Building Block, Data Definition, Data Dictionary, Document, Folder, Paragraph, TRAM and subtype of these objects. Once you select how to handle the new properties, select **Next**; the last tab is displayed.
- **Select Fields** - This area lists all of the column headings in the Excel spreadsheet that you have chosen to import and fields to select how the information should be handled during the import. The fields are listed as follows:
 - Excel Column: Lists all of the column headings in the spreadsheet.
 - TcSE Property: Allows you to indicate whether it is a new property to be created during the import. You also have the option to omit the property on import if you

choose not to add it to the Architect/Requirements database; however, this causes the column to be omitted upon import.

- o Type: Allows you to choose how the information is to be formatted within the project. You can choose; **Text**, **Date**, **Multi-Choice**, **Numeric**, or **Single Choice**.



The **Excel Import Wizard** associates existing properties (properties that already exist in the Architect/Requirements database) with new types (as happens with the new properties during the import process). For new properties, the **Excel Import** decides whether the property should be associated with types, based on which cells have a value for that property. For existing properties, if the property is already associated with the type, the values are imported. If the property is not already associated with the type, it associates with the type and the values are imported.

- **Choice Definitions** - This area shows only the choice properties that are either new or already existing in Architect/Requirements. For this area to work, you must have either one new or existing property to be imported as a **Single Choice** or **Multiple Choice** property; otherwise, this window is skipped. If there are new properties, the import wizard automatically detects them from the Excel sheet and displays them under the new choices column. If they are existing Architect/Requirements choice properties, they are shown in the **Existing** choices column. New choice properties are displayed in the **New Choices** column. This new column gives you the ability to keep the new choice or use the **Add New Choices** column and add additional choices. Select **Next** to continue.



The default type is associated with objects when the **Type Name** and **Subtype** columns either have no values, or have invalid values. This includes when they are non-existent in the Excel sheet. All the rows that do not have type or subtype information are imported as the default type selected in the **Default Type** window.



For performance reasons, the existing choices for dynamic picklist properties are not displayed.

The **Summary Window** displays all the new properties and the types they are associated with — including any new choices added to the Architect/Requirements database. Once you have finished reviewing the **Summary Window**, a **File Import** window displays the progress of the import. When complete, your file is in Architect/Requirements in the format you have chosen.

ASCII control characters are not allowed in Architect/Requirements text property values. If a control character is present in an imported Excel cell, the import fails. The error message displayed allows you to locate the problem cell. You can manually resolve the issue from the following information provided in the error message:

- The object number being processed (corresponds to the Excel row)
- The object name
- The property (column) name
- The property (cell) value

Deleting Objects

Objects that you delete from a project are moved to your Architect/Requirements Recycle Bin. These objects remain in the database, and you can restore them until you empty your Recycle Bin. For more information, see [Restoring Objects](#) and [Emptying Your Architect/Requirements Recycle Bin](#), later in this chapter.

You can delete objects from the navigation tree, the hierarchical content table, and the **Attachments** and **Links** tabs and windows. For more information, see [Navigation Tree](#), [Hierarchical Content Table](#), [Attachments Tab](#), [Links Tab](#), and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

When you delete a folder, you also delete all of the objects that it contains. When you delete a parent requirement, a parent building block, or a parent note, you also delete all of its children. You can delete children selectively, without deleting the parent.

Trace links for a deleted object are not moved to the Recycle Bin. They are disabled, not deleted, and are re-enabled if the object is restored. Individual links can be deleted without deleting the object. For more information, see [Deleting Trace Links for an Object](#) in chapter 7, *Showing Object Relationships With Trace Links*.



- To delete a folder, you must have **Full** permission for the folder and for all objects in the folder. If the folder contains building blocks, TRAMs, or diagrams, you must also have **Architect** privilege for the project.
- To delete a building block, a TRAM, or a diagram, you must have **Architect** privilege for the project. For a building block or a TRAM, you must also have **Full** permission for that object and for all of its descendants.
- To delete a requirement, you must have **Full** permission for that object and for all of its descendants.

To delete objects:

1. Select the objects that you want to delete.

In the navigation tree, you can select only one folder. In the hierarchical content table, the **Attachments** tab and window, and the **Links** tab and window, you can select one object or group of nonadjacent or adjoining objects.

To see lower level objects in a hierarchical column, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click the object and choose **Expand All** from the pop-up menu.

2. Pull down the **Edit** menu and choose **Delete**.

A message asks you to confirm this action.

3. Click **Yes** to move the objects to your Recycle Bin.

Procedure Notes

Step 2: You can also right-click the selection and choose **Delete** from the pop-up menu. Or, click the **Delete** button on the toolbar or press the delete key.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Delete**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Shortcut Objects

- You cannot delete a shortcut object if its parent object is frozen.
- The delete operation does not delete the shortcut from all versions of the parent object. The shortcut is deleted only from the current version of the parent.

Deleting a Child Object

- Deleting a child object, which is also a child of the previous version of the parent, removes the child object from the current version of the parent. The child object is not deleted, and therefore it does not appear in the Recycle Bin. This behavior enables the previous version of the parent to display the child in the currently frozen effectivity.
- Deletion of a child object is not allowed when its parent is frozen. However, this restriction is not applicable if the previous version of the child also has the same parent as the current version child. In this case, deletion is allowed, which in effect removes the current version of the child from the parent.

Restoring Objects

Before you empty your Architect/Requirements Recycle Bin, you can restore objects to the paths from which they were deleted, shown by the **Original Location** property for each object. If a path no longer exists, you can recreate it and move the objects to the new path, or you can move the objects to an existing path.

Each object is restored also to its former state. For example, a folder is restored with all of the objects that it contained when it was deleted. These objects are not displayed in the Recycle Bin and cannot be restored individually. The **Member Count** property shows the number of top level objects, which may themselves have lower level members that are contained in and restored with the displayed folder.

Furthermore, a parent requirement or building block is restored with all of its descendants. These objects are not displayed in the Recycle Bin and cannot be restored individually. The **Member Count** property shows the number of direct children, which may themselves have children that descend from and are restored with the displayed parent.

All notes and trace links for a deleted object are restored with the object. The **Attachment Count** and **Trace Link Count** properties show the number of undisplayed notes and trace links that are restored with the displayed object.



You must have **Modify** permission for the objects and for the intended destinations.

To restore objects:

1. In the navigation tree, select the Recycle Bin.
The content table displays the objects that you deleted from all projects.
2. In the content table, select the objects, right-click the selection, and then choose **Restore From Recycle Bin** from the pop-up menu.
If the former path exists, the objects are returned to that path.



Descendants of a parent whose path has been changed are returned to the parent in the modified path. Descendants of a parent that no longer exists are restored at the next higher level than that from which they were deleted.

If the former path no longer exists, a warning message is displayed. To continue, recreate the path and move the objects to the new path, or move them to an existing path. For more information, see [Creating a Folder](#) or [Moving Objects](#), earlier in this chapter.

Procedure Notes

Step 2: You can select one object or a group of nonadjacent or adjoining objects.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Restore Trash**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Emptying Your Architect/Requirements Recycle Bin

Objects that you deleted from all projects are moved to your Architect/Requirements Recycle Bin. The content table displays these objects sequentially when you select your Recycle Bin in the navigation tree. Although they are removed from their original locations, objects in the Recycle Bin remain in the database and can be restored. For more information, see [Restoring Objects](#), earlier in this chapter.

When you empty your Recycle Bin, Architect/Requirements permanently removes the objects from the database.



Objects cannot be restored after you empty your Recycle Bin.



Your Recycle Bin can be emptied automatically after a number of days specified by your Architect/Requirements enterprise administrator. Also, your enterprise administrator can empty your Recycle Bin manually, for example, to manage storage space in the database. If you have questions about emptying your Recycle Bin, consult your enterprise administrator.

Before emptying your Recycle Bin, you may want to view certain information for verification. For example, you can view the content of a requirement or note by selecting the object in the content table, pulling down the **File** menu, and choosing **Open Read-Only**. You can view a selected object's properties by pulling down the **File** menu and choosing **Properties** to display the Edit Properties dialog window, or by clicking the **Properties** tab.

Also for example, you can click the **Attachments** tab to display the notes for a selected object, or click the **Preview** tab to display the content of a selected requirement. In addition, you can open tabs in floating windows, and extend your verification to objects selected in the notebook pane.



Deleting a child object, which is also a child of the previous version of the parent, removed the child object from the current version of the parent. The child object is not deleted, and therefore it does not appear in the Recycle Bin. This behavior enables the previous version of the parent to display the child in the currently frozen effectivity.

Deletion of a child object is not allowed when its parent is frozen. However, this restriction is not applicable if the previous version of the child also has the same parent as the current version child. In this case, deletion is allowed, which in effect removes the current version of the child from the parent.

To empty your Architect/Requirements Recycle Bin:

1. In the navigation tree, select the Recycle Bin.
The content table displays the objects that you deleted from all projects.
2. Right-click the Recycle Bin and choose **Empty Recycle Bin** from the pop-up menu.
Architect/Requirements displays a message asking you to confirm this action.
3. Click **Yes**.



This action clears the queue for the **Undo** option and cannot be reversed.

Architect/Requirements removes the objects from the database.

Chapter 5: Managing Requirements

This chapter provides an overview of requirements and their management and contains instructions for creating requirements, editing and viewing requirement content, importing requirements from Microsoft Office Word, and promoting and demoting requirements within a folder hierarchy.

Overview of Requirements

The most important aspect of a requirement is its content. In Architect/Requirements, content is created and edited through Microsoft Office Word. Requirement content can consist of familiar elements such as text paragraphs and lists, hyperlinks, tables and graphics, and equations and other special characters. Paragraph formatting can be applied through Word styles and direct formatting options. Character formatting, such as boldface and italics, can be applied as well.

Every requirement resides in a folder. Within each folder, multiple levels of organization allow flexibility in the structure of requirements. For example, all requirements can occupy the folder's highest level. Or, requirements can be structured in a hierarchy, in which the levels convey relationships among the requirements in that folder.

Within a hierarchy, a requirement might occupy the highest level, as a sibling of all other requirements at level 1. Or, a requirement might occupy a lower level, as a child of the requirement at the next higher level. In turn, a child requirement might also be a parent of children at even lower levels.

Working with hierarchical requirements in Architect/Requirements is similar to working with numbered paragraphs in Word's outline view. Each requirement has a paragraph number that corresponds to the requirement's level in the hierarchy. The current level and paragraph number can be changed by promoting or demoting the requirement, or by manually changing the requirement's **Number** property.

To create requirements and content simultaneously, documents can be imported from Word. For analysis or distribution, requirements can be exported to Word, thus capturing both their content and their organization within a folder.

Though a traditional requirements specification can include hundreds or thousands of requirements, it is typically managed as a single document. In Architect/Requirements, however, each requirement is managed as a separate object, with specific properties and access control.

Creating a Requirement Object

Every requirement object must be created in a folder, which can reside directly below the project node or in another folder. Within a folder, requirements can be organized in a hierarchy, with multiple levels of parents, children, and siblings.

You can create a requirement at any level of a folder hierarchy:

- At the top level, in the last position following all other top level objects.
- As a sibling of an existing requirement that you select, directly following the selected requirement within the same level.
- As a child at the next level below an existing requirement that you select, following all other children at that level.

The new requirement object can be demoted to a lower level. It can be promoted if it does not already occupy the top level. For more information, see [Organizing Requirements in a Hierarchy](#), later in this chapter.

After creating a requirement, you can enter rich text content through the Architect/Requirements interface with Microsoft Office Word. Or, you can enter plain text content in Architect/Requirements property tables. You can import Word documents to create requirements and content simultaneously. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#), [Editing Requirement Content in Text Property Cells](#), and [Creating Requirements and Content by Import From Microsoft Office Word](#), later in this chapter.

Security profiles can be automatically applied when an object is created. Applying security profile when an object is created prevents users from creating objects of a certain type or with a certain owner. If a new object inherits a security profile from its owner or gets the Instance Security Profile from its type definition, that security profile takes effect when the object is created. If the created users do not have at least write access in the profile, they are not allowed to create the object. If Creator is specified in the profile's Full Control or Modify And Read Access list, any user is able to create the object.



Requirements can be created also in live Excel. For more information, see [Creating Objects in Live Excel](#) in chapter 9, *Working With Object Properties*.

Requirement Subtype Assignment and Inheritance

The new requirement receives certain system-defined properties, including the **Subtype** property.

- For a new top level or a new sibling requirement, you can assign the default subtype, **Requirement**, a system-defined requirement subtype, **Paragraph**, or a user-defined subtype created by your project administrator.
- When you create a sibling of an existing requirement, you can allow it to inherit the subtype of the existing sibling.
- When you create a child of an existing requirement, the child inherits the subtype of the parent.

The new requirement may also receive user-defined properties. The subtype and all other editable properties can be changed after the requirement's creation. For more information, see chapter [Working With Object Properties](#) and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

Creating a Requirement at the Top Level of a Folder



You must have **Modify** permission for the folder in which you intend to create the requirement.

1. Select the folder in the navigation tree or in the content table.
2. Assign the subtype to the new requirement by doing one of the following:
 - For the **Requirement** subtype, pull down the **File** menu and choose the **New→Requirement** options, or click the **Create New Requirement** button on the toolbar.
 - For the **Paragraph** subtype, pull down the **File** menu and choose the **New→Paragraph** options, or click the **Create New Paragraph** button on the toolbar.
 - For a user-defined subtype, pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window. To see the user-defined subtypes, click the plus sign (+) to the left of **Requirement**, and click any lower level plus signs. Select a subtype and click **OK**.

The content table displays the new requirement as the last object at the top level, with the default name in an open text field.

3. Enter the requirement name, and then press the enter key.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Procedure Notes

Step 2: You can also right-click the folder and choose the options from the pop-up menu. Or, press control-R for the **Requirement** subtype, control-H for the **Paragraph** subtype, or control-U for a user-defined subtype.

Step 2: To reverse this action, you can pull down the **Edit** menu and choose **Undo New Requirement**, **Undo New Paragraph**, or **Undo Create Subtype**. You can also click the **Undo** button on the toolbar, or press control-Z.

Creating a Sibling of an Existing Requirement



You must have **Modify** permission for the folder in which you intend to create the requirement.

1. In the content table, select the requirement that you want the new requirement to follow immediately as a sibling.
2. Assign the subtype to the new requirement by doing one of the following:
 - For the same subtype as the selected sibling, pull down the **File** menu and choose the **New→Requirement** options, or click the **Create New Requirement** button on the toolbar.
 - For the **Paragraph** subtype, pull down the **File** menu and choose the **New→Paragraph** options, or click the **Create New Paragraph** button on the toolbar.



For a user-defined subtype, pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window. To see the user-defined subtypes, click the plus sign (+) to the left of **Requirement**, and click any lower level plus signs. Select a subtype and click **OK**.

The content table displays the new requirement as the last object at the selected level, with the default name in an open text field.

3. Enter the requirement name, and then press the enter key.

To place the requirement in position, pull down the **View** menu and choose **Refresh**. Or, right-click the folder in the navigation tree and choose **Refresh** from the pop-up menu. You can also click the **Refresh** button on the toolbar.



If the **Number** property column is displayed, the values change according to the new relationships in the hierarchy.

Procedure Notes

Step 2: You can also right-click the folder or sibling and choose the options from the pop-up menu. Or, press control-R for the **Requirement** subtype, control-H for the **Paragraph** subtype, or control-U for a user-defined subtype.

Step 2: To reverse this action, you can pull down the **Edit** menu and choose **Undo New Requirement**, **Undo New Paragraph**, or **Undo Create Subtype**. You can also click the **Undo** button on the toolbar; or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating a Child of an Existing Requirement



You must have **Modify** permission for the folder in which you intend to create the requirement.

1. In the hierarchical content table, select the parent requirement.
2. Pull down the **File** menu and choose the **New→Child** options, or click the **Create New Child** button on the toolbar.

The content table displays the new requirement as the last child below the parent, with the default name in an open text field.

3. Enter the requirement name, and then press the enter key.

Procedure Notes

Step 2: You can also right-click the parent and choose the **New→Child** options from the pop-up menu. Or, press control-I. You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Child**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating Requirements and Content by Import From Microsoft Office Word

Project-related documents, such as requests for proposal, process definitions, and specifications, often contain information that you want to reuse in requirements. By importing a Microsoft Office Word document, you extract its content and create new requirements simultaneously. You can import any file type associated with Word.

Architect/Requirements generates a separate requirement for each paragraph that is formatted in a Word style with a numbered outline level. You can import requirements by outline levels only, assigning the same subtype to all imported requirements. For further granularity, you can specify keywords that occur in the document.



If the document does not begin with a Word outline level style, a note named **Preamble** is generated for the import folder. This note receives all content from the beginning to the first outline level, or the entire content if there is no outline level. The note is in the **Attachments** tab and window.

Outline levels and keywords in the import document produce a hierarchy of requirements in the content table. The plus signs in the **Name** column and the values in the **Number** column show these parent, sibling, and child relationships.

Document Parsing by Outline Levels Only

Architect/Requirements uses Word's numbered outline levels to parse the import document into multiple requirements. For each paragraph that is formatted in an outline level style, level 1 through level 9, a separate requirement is created in the folder that you select.

- The paragraph text becomes the requirement name, for example, the text of a paragraph formatted in a Word heading style
- Any markups or graphics in the paragraph titles are not retained because the titles are imported as plain text. These imported titles become the value of the **Name** property. For example, a comment inserted in the title is not retained.
- All material under the paragraph becomes the requirement content. The paragraph itself does not appear in the content.
- The next outline level style starts another new requirement.

Document Parsing by Outline Levels and Keywords

Within an outline level in the import document, further granularity of requirements can be achieved through keywords. Starting with the first Word outline level style, the parser searches for the keyword or keywords that you specify.



When parsing a Word document into Architect/Requirements using keywords such as **shall**, **will**, special characters such as **&**, **\$**, **#**, ***** are not allowed.

- If no keyword is found before the next outline level, the requirement starts with the preceding outline level and contains all material under that paragraph. The paragraph text is the requirement

name, and the paragraph itself does not appear in the content. The **Paragraph** subtype is assigned by default.

- If only one keyword is found before the next outline level, the requirement starts after the nearest outline level preceding the keyword:
 - The text of the immediately preceding outline level paragraph becomes the requirement name.
 - All material under the outline level paragraph becomes the requirement content. The paragraph itself does not appear in the content.
 - The requirement has the subtype that you assign.
- If two or more keywords are found before the next outline level, the requirements are separated as follows:
 - A paragraph with no content is created. The text of the preceding heading level is used as the name of the paragraph.
 - A requirement or paragraph, depending on the selection, is created based on the sentence breaks. The text preceding the sentence containing the second keyword becomes the first requirement. The **Document Options** property value for the requirement is set to **Disable Numbering**.
 - The subsequent requirements contains the sentence containing the next instance of the keyword and the sentences following it till the next instance of the keyword is found.

These requirements also become the next child of the empty paragraph. Following is an example of the document text containing multiple keywords and the paragraph and requirements created after parsing it.

The headlamp assembly

The headlamp assembly is a composite part. The headlamp assembly *shall* include the lamp, the reflector, the socket, and the lens. It can also include multiple lamps in the same unit. The socket *shall* hold the lamp and connect to the battery. The reflector will position the lamp in the correct position. The lens *shall* focus and diffuse the light such that it provides a good view of the road while driving at night.

 The headlamp assembly	2.1.2	0320	Requirement
 The headlamp assembly	2.1.2.-1	0321	Requirement
 The headlamp assembly	2.1.2.-2	0322	Requirement
 The headlamp assembly	2.1.2.-3	0323	Requirement

Figure 5-1. Parsing a document based on multiple keywords

In the example in figure 5-1, the document is parsed and a paragraph with no content is created. It is identified as **2.1.2**. The following requirements are created:

Requirement number	Requirement text
2.1.2.-1	The headlamp assembly is a composite part. The headlamp assembly shall include the lamp, the reflector, the socket, and the lens. It can also include multiple lamps in the same unit.
2.1.2.-2	The socket shall hold the lamp and connect to the battery. The reflector will position the lamp in the correct position.
2.1.2.-3	The lens shall focus and diffuse the light such that it provides a good view of the road while driving at night.



If multiple keywords are found in the same sentence, they are treated as a single entry and only one requirement is created with the complete sentence.

The name of each requirement is the text of the immediately preceding outline-level paragraph. All requirements have the same subtype, which you assign.

Tables, irrespective of whether they contain the keyword, form a part of the requirement containing the preceding paragraph.

At the next outline level, the parser repeats the processing described above.



The **TcSEPreprocessor.dot** file is required for parsing by keywords. This file is installed in the Architect/Requirements client installation folder. A progress bar indicates the progress of the parsing process. Although the process is optimized for speed, it may take longer for large documents, and particularly for documents with large tables or several tables.

Style Sheet of the Import Folder

The import folder has its own style sheet, which is controlled by the document template that is applied to the folder. The styles in the Word document do not change the folder's style sheet, nor do they affect the imported requirements.

The folder's **Document Template** property shows the current document template. This value is blank if the default document template for the project is applied. You can view or edit that property in the **Properties** tab or floating window. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.

To import requirements from Microsoft Office Word:



- You must have **Modify** permission for the intended import folder.
 - Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.
1. In the navigation tree or the content table, select the import folder, and then pull down the **File** menu and choose the **Import→Word Document** options.

A new Word session starts with the **Open** dialog window.

2.

Select the import document, and then click **Open**.

Word closes and Architect/Requirements displays the Document Import dialog window.

3. Under **Without keyword parsing, import all text as**, select the subtype for the requirements.

For further requirement granularity, you can specify one or more keywords:

- a. Under **Identify Requirement by**, click the **Keyword (eg: shall, will,...)** button.
- b. Enter the keyword or keywords in the text field, with a comma between two keywords.



To enable case sensitivity and wildcard characters, click the **Match Case and Allow Wildcards** button.

A requirement is created for each occurrence of a keyword. The selected subtype is assigned to each requirement.



For each outline level under which a keyword is not found, a requirement is created with the **Paragraph** subtype assigned by default.

4. Click **OK** to start the import.



This action clears the queue for the **Undo** option and cannot be reversed.

A message states that the document is being imported. You can close the message and perform other actions while the import is in progress. When the import is complete, a confirmation message is displayed.



If the **Number** value for an imported requirement contains a dash before a level, for example, **1.-1**, the requirement is an unnumbered part of the parent paragraph at the next higher level. The requirement's **Document Options** property value is **Disable Numbering**. When such a requirement is subsequently exported to Word, it is not assigned a heading or an outline level in the export document.

Procedure Notes

Step 1: You can also right-click the import folder and choose the **Import→Word Document** options from the pop-up menu.

Step 3: You can select the default subtype (**Requirement**), a system-defined requirement subtype (**Paragraph**), or a user-defined subtype. If you have questions about subtypes, consult your project administrator.

Entering and Changing Requirement Content in Microsoft Office Word

When you open a requirement, Architect/Requirements generates a temporary Microsoft Office Word file, which you use to edit the content in the database.



Microsoft Word must be installed on your computer.

For information about version of Microsoft Word, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.



In Word, graphical shapes can be added to a document. These shapes can optionally be placed on a drawing canvas. When a requirement or note containing a canvas is edited with different versions of Word, the canvas may get corrupted. In such cases, the shapes are not displayed correctly.

To avoid this problem, do not use Word drawing canvases in Architect/Requirements requirement and note content. Add the shapes to the Word document directly.

Content Elements

As in a typical Word file, you can enter and change content elements such as:

- Headings, body paragraphs, lists, and hyperlinks.
- Comments and footnotes.
- Tables and graphics.
- Equations, symbols, and other special characters.

You can copy, move, and delete selected portions by the standard Word functions.

Content Formatting

You can apply manual formatting to content in the database through Word's **Format** menu and formatting toolbar.

The Word styles that you can apply are governed by the style sheet for the folder that contains the requirement. That style sheet is specified by the document template that is associated with the containing folder.

The document template also specifies the information that Architect/Requirements extracts from the database when you export data for objects in the folder to Word. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.

The current document template is shown by the folder's **Document Template** property, an editable system-defined property. If the property's value is blank, the default document template for the project is currently applied to the folder.



Before you open the requirement for editing, you can view the containing folder's **Document Template** property in the **Properties** tab or floating window. You can also change that property value to specify different styles for the Word file. For more information, see [Properties Tab](#) or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*, or chapter 9, [Working With Object Properties](#). If you have questions about document templates or style sheets, consult your project administrator.

Changes to styles in the temporary file do not affect the content in the database. Although you can modify and create styles, and apply them with visible results in Word, such changes are not retained when you close the file. Nor is it effective to modify Word template settings or to attach other templates.

Formatting Lost in Microsoft Word

Your document is likely to lose the Bullets and Numbering formatting when a requirement is exported from Architect/Requirements to Microsoft Word. The problem occurs when the exported document does not contain the styles that are used in the document body. Siemens PLM Software recommends applying style-based formatting on the text to prevent losing the formatting.



Use the **List Number** style or **List Bullets** style for applying numbers or bullets to text instead of directly formatting the text using **Bullets** or **Numbers**.

Page Setup

Page setup modifications have no effect in the database. Changes to margins, paper, layout, or headers and footers are not retained beyond the temporary Word session. The next time you open that requirement, Architect/Requirements applies the initial settings.

Editing Requirement Content in Microsoft Office Word



You must have **Modify** permission for the requirement.

1.

Select the requirement in the content table, the **Links** tab or window, or the **Versions** tab or window.

2. Pull down the **File** menu and choose **Open**.

The requirement opens in an **.mhtml** Word file, in which you enter and change content as described above. This file is deleted from your computer when you exit Architect/Requirements.

3. To save the content in the database, pull down Word's **File** menu and choose **Save**, or click the **Save** button on Word's toolbar.



When the content is saved in the database, editing is disabled in the requirement's **Text** property cell. However, this cell can be reset for direct editing. For more information, see [Enabling Text Property Editing for a Requirement](#) and [Editing Requirement Content in Text Property Cells](#), later in this chapter.



If you choose **Save As**, the content is saved in the database but Word's Save As dialog window is not displayed. Therefore, the file name, file type, and location cannot be changed.

After you close this file, you can create a permanent file with the same content by exporting the requirement to Word. Or, you can open the requirement as a read-only file and choose **Save As** to display Word's Save As dialog window. For more information, see [Viewing Requirement Content](#), later in this chapter.

Procedure Notes

Step 2: You can also right-click the requirement and choose **Open** from the pop-up menu. Or, double-click the requirement's object type indicator.

Using OLE Objects in Word

Architect/Requirements supports Word text content that contains OLE embedded objects. Excel spreadsheets and Visio diagrams can be included in requirements and notes. Content with OLE objects can be created using Word document import or by inserting objects while editing a requirement or note.

When an embedded OLE object is used, the object content is stored in the database and is available to all. The OLE content is available when editing or viewing a single requirement or note. However, the OLE content is not included in document exports. Architect/Requirements does not merge OLE content from different sources into a single exported document. Preview images for the OLE objects are included in the export, however, they cannot be opened as OLE objects.

Microsoft Word's **Link to File** option is not a viable alternative to embedded OLE objects. When a file link is used to reference a file on a local disk drive, then the linked file will not be accessible when accessing the text on another machine. There are also issues even if the referenced file is on a shared drive accessible via the same path name on all user machines. Since Architect/Requirements objects are edited individually, Word generates duplicate identifiers for the file references. Those duplicates become a problem when multiple requirements or notes are later exported together as one Word document.

Referencing Information in Requirement Text Edited in Microsoft Word

There are two ways to reference information in requirement text edited in Microsoft Word.

- By referencing Architect/Requirements object text or properties, described in the [Storing Property Values Through Reference Links](#) section in this manual.
- Using the traditional Microsoft Word caption and bookmark features.

Microsoft Word's captions and bookmarks works correctly in documents where the caption or the definition of a bookmark is in the same document where it is referenced. Requirement text is a large document that is split into many small documents. Microsoft Word does not support references across different documents but you can use some special techniques to create the reference.

The difference between captions and bookmarks are that the caption name specific items such as tables and graphics in the document. A caption is a combination of text and fields. Cross references are created by making a link to the hidden bookmark on the caption. Hidden bookmarks are depicted by underscores preceding them.

The cross reference and caption issue is overcome by using user-defined bookmarks rather than the default hidden bookmarks in a document. The visible bookmarks are then referenced by Architect/Requirements. Following is an example of cross referencing.

Project **A** has folder **One**. The folder contains 3 requirement objects, **REQ-A**, **REQ-B**, and **REQ-C**. You need to create a cross reference in **REQ-C** that references a bookmark in **REQ-A**.

To create a cross reference in a document that references a bookmark in another document:

1. Open **REQ-A** in Microsoft Word and select the object or text being referenced.
2. Select **Insert** → **Bookmark** and enter a name for the bookmark.



Do not begin the bookmark name with an underscore (**_**) as bookmarks beginning with underscores are hidden.

3. Navigate to the end of the open requirement.

4. Select **Insert** → **Cross-Reference**.
5. Select **Reference Type: Bookmark** and **Insert Reference to: Bookmark Text**.
6. Select the required bookmark, click **Insert**, and then click **Close**.
7. Select the inserted cross-reference and select **Home** → **Cut**.
8. Open the document **REQ-C** in Microsoft Word.
9. Navigate to the location in the document where you want the cross-reference and paste the reference using **Home** → **Paste**.
10. Save both **REQ-A** and **REQ-C** and close them.



Opening **REQ-C** displays a reference error. This is due to the bookmark not being defined in the same document. Ignore the error. The error disappears when the complete document is later exported to Microsoft Word and the reference resolves correctly.

11. Right click the folder and generate the final document. The final document has the bookmark and the cross-reference.

Editing Requirement Content in Text Property Cells

Plain text content can be edited directly in the cells of the **Text** property. A column for this property can be added in the hierarchical content table, the **Trace** subtab of the **Links** tab and window, the **Versions** tab and window, and the Search Results dialog window. The **Text** property is displayed by default in the **Properties** tab and window and the Edit Properties dialog window. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.

You can use any of these client views to edit the **Text** property for one requirement. For two or more requirements, you can use only the Edit Properties dialog window.



You must have **Modify** permission for each requirement that you intend to edit.



In the content table, you can set the number of lines in the cells. Pull down the **View** menu and choose **Lines Per Row** and the number. Or, click the **#** button on the toolbar and enter the number in the **Lines per row** field.

To edit a Text property cell for one requirement:

In the content table, the **Trace** subtab or window, the **Versions** tab or window, or the Search Results dialog window:

1. With the **Text** property added, double-click the cell that you want to edit.



A shaded background and blue text indicate that cell editing is disabled. The requirement's **Text Format** property value is **MHTML**, for content saved in Microsoft Office Word. Cell editing can be enabled by changing the **Text Format** value to **Text**. For more information, see [Enabling Text Property Editing for a Requirement](#), later in this chapter.

A text field opens in the cell. Scroll bars are provided to the right if the text length exceeds the width of the **Text** column.

2. Enter your changes, and then press the enter key or click outside the field.

In the **Properties** tab or window or the Edit Properties dialog window:



To see more text in the **Text** column, change the width before you begin. Scroll bars are not provided in the cell.

1. Double-click the **Text** property value to open a text field in the cell.



Dimmed text indicates that cell editing is disabled. The requirement's **Text Format** property value is **MHTML**, for content saved in Microsoft Word. Cell editing can be enabled by changing the **Text Format** value to **Text**. For more information, see [Enabling Text Property Editing for a Requirement](#), later in this chapter.

2. Enter your changes, and then press the enter key or click outside the field.

To display the **Properties** tab, select the requirement and click the tab. You can open the **Properties** window by clicking the **Open tab** button on the notebook pane toolbar. To display the Edit Properties dialog window, select the requirement, pull down the **File** menu, and choose **Properties**.

To edit Text property cells for two or more requirements:



The **Text** property cannot be displayed if any one selected requirement has a **Text Format** property value of **MHTML**, for content saved in Microsoft Word. Before you start this procedure, ensure that the **Text Format** value is **Text** for each requirement that you intend to edit. You can set this value if necessary. For more information, see [Enabling Text Property Editing for a Requirement](#), later in this chapter.

1. Select the requirements.

You can select nonadjacent and adjoining requirements of the same subtype or of mixed subtypes.

2. Pull down the **File** menu and choose **Properties**.

The Edit Properties dialog window is displayed.

3. In the **Value** column, double-click the **Text** property cell.

A text field opens in the cell.

4. Enter your changes, and then press the enter key or click outside the field.

5. Click **OK** to close the dialog window.

Procedure Notes

Step 1: You can select the requirements in the content table, the **Links** tab or window, the **Versions** tab or window, or the Search Results dialog window.

Step 2: You can also right-click the selection and choose **Properties** from the pop-up menu.

Enabling Text Property Editing for a Requirement

When editing is disabled in a **Text** property cell, the requirement has a **Text Format** property value of **MHTML**. This value indicates content saved in Microsoft Office Word.

Disabled cell editing is indicated also by certain visual cues in the **Text** property cell. Depending on the client view where the property is displayed, the cues are:

- A shaded background and blue text for disabled cells in the hierarchical content table, the **Trace** subtab of the **Links** tab and window, the **Versions** tab and window, and the Search Results dialog window.
- Dimmed text for disabled cells in the **Properties** tab or window and the Edit Properties dialog window.



Although you can edit disabled **Text** property cells in a live Excel file, edits are not saved in the Architect/Requirements database. An error message is displayed when you click outside the cell. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

For a requirement with such a cue, you can enable direct editing in the **Text** property cell by changing the **Text Format** property value to **Text**. For more information, see [Editing Requirement Content in Text Property Cells](#), earlier in this chapter.



The next time the cell is edited, current headings, body text, lists, and table text are converted to plain text, and all text is concatenated into one paragraph. All formatting is removed, including graphics, table grids, and other content elements that may be saved in Word. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#), earlier in this chapter.

To enable Text property editing for a requirement:

1. Select the requirement, and then do one of the following:
 - To use the **Properties** tab, click the tab. You can open this tab in a floating window by clicking the **Open tab** button on the notebook pane toolbar.
 - To use the Edit Properties dialog window, pull down the **File** menu and choose **Properties**. Or, right-click the requirement and choose **Properties** from the pop-up menu.
2. In the **Value** column, double-click the **Text Format** property value.
3. In the Single-Choice dialog window, check the **Text** check box and click **OK**.



This action changes the text format to plain text and removes all other content elements, effective the next time the cell is edited.

A warning message states that saving as plain text causes all formatting, pictures, and objects in the text to be lost on the next text edit.

4. To continue, click **OK** to close the Single-Choice dialog window.
5. Click **Close** to apply the change and close the Edit Properties dialog window.

Organizing Requirements in a Hierarchy

Requirements in a folder hierarchy can be organized in multiple levels of parents, children, and siblings. The content table shows this hierarchy in the **Name** column, with a plus sign for each requirement that has one or more children. In the **Number** property column, the values indicate each requirement's level in the hierarchy and its sequence within that level, according to the numbered outline style.

You can promote and demote requirements to higher and lower levels, thereby changing their relationships and **Number** property values. For example, you may want to promote or demote requirements after completing certain actions that affect the hierarchy. Such actions include creating and importing requirements, and copying, moving, deleting, and restoring objects. For more information, see [Creating a Requirement Object](#) and [Creating Requirements and Content by Import From Microsoft Office Word](#), earlier in this chapter, and chapter 4, [Maintaining a Project](#).

Positions of Promoted Requirements

A promoted requirement moves to the next higher level, as a sibling of all other requirements at the new level. In addition, all of the requirement's children move up by one level, including direct children and all lower level descendants. For example:

- A requirement previously occupying level 2, with number *n.2*, moves to level 1, with number *n*.
- Children previously occupying level 3, with numbers *n.2.1* and *n.2.2*, move to level 2, with numbers *n.1* and *n.2*.

Requirements cannot be promoted above the folder's top level, shown in the **Number** column by the highest level in the numbered outline style. A top level number has no following decimal point and consists of one or more digits, for example, *n* and *nn*.

Positions of Demoted Requirements

A demoted requirement moves to the next lower level, as a child of the immediately preceding sibling at the previous level. In addition, all of the requirement's children move down by one level, including direct children and all lower level descendants. For example:

- A requirement previously occupying level 2, with number *n.2*, moves to level 3, with number *n.1.1*.
- Children previously occupying level 3, with numbers *n.2.1* and *n.2.2*, move to level 4, with numbers *n.1.1.1* and *n.1.1.2*.

A requirement cannot be demoted if:

- Its **Number** value precedes that of any other requirement at the top level.
- It is a parent requirement's first direct child at any level, for example, numbers *n.1*, *n.n.1*, and *n.n.n.1*.
- Its **Number** value directly follows that of a building block within the current level.

Number Property Values of Promoted and Demoted Requirements

For both promoted and demoted requirements, **Number** property values are changed automatically for successive requirements at the new level and all lower levels. Also, values are changed for requirements that followed the promoted or demoted requirement at the previous level. Because requirements and building blocks can occupy the same level, numbers are changed also for building blocks that follow the promoted or demoted requirement within the new level.



When you change the **Number** property of a requirement from a positive value to a negative value (or vice versa), the requirements are reorganized. However, the **Document Options**→**Disable Numbering** property value does not change with any change in the **Number** value. The behavior of these requirements in export is based on the corresponding **Disable Numbering** values.

To change the **Number** property from a positive value to a negative value (or vice versa), Siemens PLM Software recommends that you use the **Document Options**→**Disable Numbering** property.

You can edit a requirement's **Number** property to promote or demote the requirement manually. For more information, see chapter 9, [Working With Object Properties](#).

Promoting and Demoting Requirements



- To promote a requirement to the folder's top level, you must have **Modify** permission for the requirement and for the folder. For any lower level, you must have **Modify** permission for the requirement and for the intended parent requirement.
 - To demote a requirement, you must have **Modify** permission for the requirement and for the intended parent requirement.
1. In the hierarchical content table, select each requirement that you want to promote or demote. You can select nonadjacent and adjoining requirements.
 2. Pull down the **Edit** menu and choose **Promote** or **Demote**, or click the **Promote** or **Demote** button on the toolbar.

The content table displays each selected requirement at its new level. In the **Name** column, plus signs or minus signs are displayed or removed for affected parent, sibling, and child relationships. In the **Number** column, values are changed according to the new relationships.

Procedure Notes

Step 2: You can also right-click the selection and choose **Promote** or **Demote** from the pop-up menu. In addition, you can point to the selection, hold down the left mouse button, and drop the selection within the new level.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Promote** or **Undo Demote**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Viewing Requirement Content

There are two ways to view the content of a requirement:

- In a read-only Microsoft Word file, you can view and print the content, and you can send it to E-mail and fax recipients. You cannot change the content in the database, and therefore you do not interfere with someone else's work.



Microsoft Office must be installed on your computer.

- In the **Preview** tab or window, you can view and print the content without opening the requirement in Word. You cannot change the content in the database. For more information, see [Preview Tab](#) or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

You can use Word to open requirements selected in the hierarchical content table, the **Links** tab or window, and the **Versions** tab or window. For requirements selected in the tabs and in your Architect/Requirements Recycle Bin, you can view content in the **Preview** tab or window. However, requirements in your Recycle Bin cannot be opened in Word. For more information, see [Hierarchical Content Table](#), [Architect/Requirements Recycle Bin](#), [Links Tab](#), and [Versions Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To view requirement content in Microsoft Word:

Select the requirement, pull down the **File** menu, and choose **Open Read-Only**. Or, right-click the requirement and choose **Open Read-Only** from the pop-up menu.

The requirement opens in an **.mhtml** Word file. This temporary file is deleted when you exit Architect/Requirements.

To view requirement content in the Preview tab or window:

Do one of the following:

- For a requirement in the hierarchical content table or the Recycle Bin, select the requirement and click the **Preview** tab. You can open the **Preview** window for this requirement by clicking the **Open tab** button on the notebook pane's toolbar.
- For a requirement in the **Links** or **Versions** tab:
 - . With the **Preview** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Preview** window.
 - . Select the requirement in the **Links** or **Versions** tab to display the content in the **Preview** window.

Comparing the Content of Different Requirements

Architect/Requirements works with the **Compare and Merge Documents** option in Microsoft Office Word to give you features for comparing requirement content. The comparison can be between two selected requirements, or between the requirements in two selected folders. The requirements or folders can reside in the same project or in different projects.

When you compare two requirements, you can choose to view only the content of a selected requirement. Or, you can choose to include the content of all direct children and lower-level descendants; this deep content is exported to a single Word document.

A comparison of two folders shows you the content of all requirements in each requirement hierarchy. The deep content of each folder's parent and child requirements is exported to a single Word document.

Setting the Comparison Mode

The default comparison mode uses Word's Legal blackline option. In this mode, the first requirement is opened in Word and the second requirement is merged into the first document. Then, the results are shown in a new, third Word document.



In the Legal blackline mode, only the first requirement's content can be edited. The second requirement is not opened in Word.

You can also set a comparison mode that uses Word's side by side option. In this mode, each document is opened in a separate Word window. By choosing **Compare Side by Side with** from Word's **Window** menu, you can tile the documents vertically for comparison. A common scroll bar synchronizes the content.



In the side by side mode, the content of each requirement can be edited if the document is generated from only one level of the hierarchy. Content from a deep comparison cannot be edited.

- To switch between modes, pull down the **Tools** menu and choose the **Compare**→**Compare Side by Side** options.

The side by side mode is in effect when a checkmark appears beside these options.

The Legal blackline mode is in effect when no checkmark appears.



When the side by side comparison is turned on, the difference in the documents is not visible.



The effective mode persists until you switch to the other mode or you exit Architect/Requirements. The Legal blackline mode is effective automatically when you start a new Architect/Requirements session.

For more information, see Microsoft Office Word Help.

Comparing Two Requirements

1. In the content table or the **Versions** tab or window, select the first requirement.
2. Pull down the **Tools** menu and choose one of the following options:
 - Choose **Compare**→**Start Compare 1st Level** to compare only the selected requirement.
 - Choose **Compare**→**Start Compare Deep** to compare the selected requirement and all of its direct children and lower-level descendants.
3. In the content table or the **Versions** tab or window, select the requirement that you want to compare with the first requirement.

You can select a requirement in the same project or in a different project.

4. Pull down the **Tools** menu and choose one of the following options:
 - Choose **Compare**→**End Compare 1st Level** to compare only the selected requirement.
 - Choose **Compare**→**End Compare Deep** to compare the selected requirement and all of its direct children and lower level descendants.

The Word export process begins with the Select Document Template dialog window.

- To use the selected document template for all objects, click **OK**.
- To change the stylesheet for the document or the object template for one or more types, click **More**.

The Document Template dialog window is displayed.

- Do one or both of the following:
 - To change the Word styles that are applied to all data in the document, select the style sheet in the **Select Stylesheet** field.
 - To change the object template for a type in the **SubTypes** column:
 - Double-click the cell in the **OverrideObjectTemplate** column to display the **Single-Choice** dialog window.
 - Check the check box for the object template that you want to use, and then click **OK** to close this dialog window.

Repeat these steps for each additional type whose object template you want to change.

- Click **Export**.

The requirements open in the selected Word comparison mode. Use Word's comparison features to view the content. For more information, see the Microsoft Office Word Help.

Comparing the Requirements in Two Folders

1. In the content table, select the first folder.
2. Pull down the **Tools** menu and choose **Compare**→**Start Compare Deep**.

This option exports to Word the entire requirement hierarchy in the folder.

3. In the content table, select the folder containing the requirements that you want to compare with those in the first folder.

You can select a folder in the same project or in a different project.

4.

Pull down the **Tools** menu and choose **Compare**→**End Compare Deep**.

The Select Document Template dialog window is displayed. Here, you associate the template for exporting the first folder's requirements to Word.

- To use the selected document template for all objects, click **OK**.
- To change the stylesheet for the document or the object template for one or more types, click **More**.

The Document Template dialog window is displayed.

. Do one or both of the following:

- o To change the Word styles that are applied to all data in the document, select the style sheet in the **Select Stylesheet** field.
- o To change the object template for a type in the **SubTypes** column:
 - a.

Double-click the cell in the **OverrideObjectTemplate** column to display the **Single-Choice** dialog window.

- a. Check the check box for the object template that you want to use, and then click **OK** to close this dialog window.

Repeat these steps for each additional type whose object template you want to change.

. Click **Export**.

The Select Document Template dialog window is redisplayed. Here, you associate the template for exporting the second folder's requirements to Word.

- o Repeat the steps listed for the first folder above.

The exported requirements open in the selected Word comparison mode. Use Word's comparison features to view the content. For more information, see the Microsoft Office Word Help.

Chapter 6: Constructing System Views With Building Blocks and Diagrams

This chapter discusses methods of constructing views for system decomposition. Instructions are provided for using building blocks and live Visio, the Architect/Requirements interface with Microsoft Office Visio. **Architect** privilege is required for most of these procedures.

Building Blocks

With building blocks, you can construct hierarchies that decompose systems and illustrate relationships among system elements. A building block can represent any element in a hierarchy, such as a function or a component of a product, a task in a work breakdown structure, or a job function in an organizational chart.

Building blocks reside in folders, and a folder can contain any number of building blocks. Below each top level building block, subordinate building blocks can be organized in multiple levels of parents, children, and siblings.

For example, to construct a view of an entire system, create a single building block for the system at the top level. At the next lower level, create child building blocks for the major subsystems. Below those, continue to decompose the system into finer levels of detail.

In such a hierarchy, building blocks can be created at specific levels, promoted to higher levels, demoted to lower levels, and moved up or down within a level. For each building block, the **Number** property value shows the level in the hierarchy and the position within the level. As the structure changes, values are renumbered automatically for affected building blocks.

Trace links can be created to and from building blocks, to show defining and complying relationships with other objects. Such relationships may exist between building blocks that reside within the same hierarchy. Also, building blocks may have trace links to objects that reside elsewhere in the same Architect/Requirements project, in a different project, or in a different Teamcenter product.

A system-defined building block subtype, **TRAM**, can be used for *transitional mapping*, a method of interrelating system views for comparison and analysis. Through trace links, a TRAM can be associated with building block hierarchies to create a flow of information among source views and destination views, with the TRAM as the focal point.

Any number of source views can be linked to a TRAM, with the peak building blocks as defining objects. From a TRAM, peak building blocks in any number of destination views can be linked as complying objects. When the TRAM is selected, these defining and complying paths can be followed to evaluate relationships among the systems. For more information, see [Creating Trace Links](#) and [Viewing Objects in a Defining or Complying Path](#) in chapter 7, *Showing Object Relationships With Trace Links*.

Creating a Building Block

You can create a building block at any level of a folder hierarchy:

- At the top level, in the last position following all other top level objects.
- As a sibling of an existing building block that you select, directly following the selected building block within the same level.
- As a child at the next level below an existing building block that you select, following all other children at that level.

After its creation, the new building block can be demoted to a lower level, and it can be promoted if it does not already occupy the top level. For more information, see [Organizing Building Blocks in a Hierarchy](#), later in this chapter.

The new building block receives certain system-defined properties, including the **Subtype** property. When you create a child of an existing building block, the child inherits the subtype of the parent. When you create a sibling of an existing building block, you can allow it to inherit the subtype of the existing sibling, or you can assign the subtype as you do when you create a building block at the top level. For a new top level or a new sibling building block, you can assign the default subtype (**Building Block**), a system-defined building block subtype (**TRAM**), or a user-defined subtype created by your project administrator.

The new building block may also receive user-defined properties. The subtype and other editable properties can be changed after the building block's creation. For more information, see chapter 9, [Working With Object Properties](#) and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

After you create the building block, you can attach diagrams through live Visio, the Architect/Requirements interface with Microsoft Office Visio. For more information, see [Creating a Diagram](#), later in this chapter.

Before or after you attach the diagram, you can copy the new building block to other locations, link it to other objects, attach notes, and create versions and variants. For more information, see [Copying Objects](#) in chapter 4, *Maintaining a Project*; [Creating Trace Links](#) and [Linking to an Object in Another Teamcenter Product](#) in chapter 7, *Showing Object Relationships With Trace Links*; [Attaching a Note to an Object](#) in chapter 8, *Recording Supplementary Information With Notes*; and [Creating Versions](#) or [Creating Variants](#) in chapter 10, *Working With Versions*.

Building blocks can also be created in the live Excel interface. For more information, see [Creating Objects in Live Excel](#) in chapter 9, *Working With Object Properties*.



- You must have **Architect** privilege for the project.
- You must have **Modify** permission for the folder in which you intend to create the building block.

To create a building block at the top level of a folder:

1. Select the folder in the navigation tree or in the hierarchical content table.
2. Assign the subtype to the new building block by doing one of the following:
 - For the **Building Block** subtype, pull down the **File** menu and choose the **New→Building Block** options, or click the **Create New Building Block** button on the toolbar.
 - For the **TRAM** subtype, pull down the **File** menu and choose the **New→TRAM** options.
 - For a user-defined subtype, pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window. To see the user-defined subtypes, click the plus sign (+) to the left of **Building Block**, and click any lower level plus signs. Select a subtype and click **OK**.

The content table displays the new building block as the last object at the top level, with the default name in an open text field.

3. Enter the building block name, and then press the enter key.

Procedure Notes

Step 2: You can also right-click the folder or sibling and choose the options from the pop-up menu. Or, press control-B for the **Building Block** subtype or control-U for a user-defined subtype. To reverse this action, you can pull down the **Edit** menu and choose **Undo New Building Block**, **Undo New TRAM**, or **Undo Create Subtype**; click the **Undo** button on the toolbar; or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

To create a sibling of an existing building block:

1. In the hierarchical content table, select the building block that you want the new building block to follow immediately as a sibling.
2. Assign the subtype to the new building block by doing one of the following:
 - For the same subtype as the selected sibling, pull down the **File** menu and choose the **New→Building Block** options, or click the **Create Building Block** button on the toolbar.
 - For the **TRAM** subtype, pull down the **File** menu and choose the **New→TRAM** options.
 - For a user-defined subtype, pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window. To see the user-defined subtypes, click the plus sign (+) to the left of **Building Block**, and click any lower level plus signs. Select a subtype and click **OK**.

The content table displays the new building block as the last object at the selected level, with the default name in an open text field.

3. Enter the building block name, and then press the enter key.

To place the building block in position, pull down the **View** menu and choose **Refresh**. Or, right-click the folder in the navigation tree and choose **Refresh** from the pop-up menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the folder or sibling and choose the options from the pop-up menu. Or, press control-B for the **Building Block** subtype or control-U for a user-defined subtype. To reverse this action, you can pull down the **Edit** menu and choose **Undo New Building Block**, **Undo New TRAM**, or **Undo Create Subtype**; click the **Undo** button on the toolbar; or press control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

To create a child of an existing building block:

1. In the hierarchical content table, select the parent building block.
2. Pull down the **File** menu and choose the **New**→**Child** options, or click the **Create New Child** button on the toolbar.

The content table displays the new building block as the last child below the parent, with the default name in an open text field.

3. Enter the building block name in the text field, and then press the enter key.

To place the building block in position, pull down the **View** menu and choose **Refresh**. Or, right-click the folder in the navigation tree and choose **Refresh** from the pop-up menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the parent and choose the **New**→**Child** options from the pop-up menu. Or, press control-I. You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Child**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Organizing Building Blocks in a Hierarchy

Within a folder, building blocks can be organized in multiple levels of parents, children, and siblings. The content table shows this hierarchy in the **Name** column, with a plus sign for each building block that has one or more members at lower levels. In the **Number** property column, the values indicate each building block's level in the hierarchy and its sequence within that level, according to the numbered outline style.

To reorganize the hierarchy, you can promote and demote building blocks to higher and lower levels, which changes their relationships and **Number** property values. For example, you may want to promote or demote building blocks after completing certain actions that affect the hierarchy, such as creating building blocks and copying, moving, deleting, and restoring objects. For more information, see [Creating a Building Block](#), earlier in this chapter, and chapter 4, [Maintaining a Project](#).

A promoted building block moves to the next higher level, as a sibling of all other building blocks at the new level. In addition, all of the building block's members move up by one level, including direct children and all lower level descendants. For example:

- A building block previously occupying level 2, with number *n.2*, moves to level 1, with number *n*.
- Children previously occupying level 3, with numbers *n.2.1* and *n.2.2*, move to level 2, with numbers *n.1* and *n.2*.

A demoted building block moves to the next lower level, as a child of the immediately preceding sibling at the previous level. In addition, all of the building block's members move down by one level, including direct children and all lower level descendants. For example:

- A building block previously occupying level 2, with number *n.2*, moves to level 3, with number *n.1.1*.
- Children previously occupying level 3, with numbers *n.2.1* and *n.2.2*, move to level 4, with numbers *n.1.1.1* and *n.1.1.2*.

In both cases, **Number** property values change for successive building blocks at the new level and all lower levels. Also, values change for building blocks that followed the promoted or demoted building block at the previous level. Because building blocks and requirements can occupy the same level, numbers change also for requirements that follow the building block within the new level.

Building blocks cannot be promoted above the folder's top level, shown in the **Number** column by the highest level in the numbered outline style. A top-level number has no following decimal point and consists of one or more digits, for example, *n* and *nm*.

A building block cannot be demoted if:

- Its **Number** value precedes that of any other building block at the top level.
- It is a parent building block's first direct child at any level, for example, numbers *n.1*, *n.n.1*, and *n.n.n.1*.
- Its **Number** value directly follows that of a requirement within the current level.

You can also promote or demote a building block by changing its value in the **Number** column. For more information, see [Editing the Properties of a Selected Object](#), [Editing Properties in Table View Cells](#), or [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.



- You must have **Architect** privilege for the project.
- To promote a building block to the folder's top level, you must have **Modify** permission for the building block and for the folder. For any lower level, you must have **Modify** permission for the building block and for the intended parent building block.
- To demote a building block, you must have **Modify** permission for the building block and for the intended parent building block.

To organize building blocks in a hierarchy:

1. In the hierarchical content table, select each building block that you want to promote or demote.
You can select nonadjacent building blocks by holding down the control key while you click the building blocks. To select adjoining building blocks, click the first building block, hold down the shift key, and click the last building block.
2. Pull down the **Edit** menu and choose **Promote** or **Demote**, or click the **Promote** or **Demote** button on the toolbar.

The content table displays each selected building block at its new level. In the **Name** column, plus signs or minus signs are displayed or removed for affected parent, sibling, and child relationships. In the **Number** column, values are changed according to the new relationships.

To reposition building blocks within their new levels, pull down the **View** menu and choose **Refresh**. Or, right-click the folder in the navigation tree and choose **Refresh** from the pop-up menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 1: To see lower level objects, click the plus signs in the **Name** column. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also right-click the selection and choose **Promote** or **Demote** from the pop-up menu. In addition, you can point to the selection, hold down the left mouse button, and drop the selection within the new level.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Promote** or **Undo Demote**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Live Visio Diagrams

Typically, diagrams are attached to building blocks, which can represent superior and subordinate elements of a system view. Diagrams can be attached also to folders, requirements, and groups.

Diagrams are created and edited in Architect/Requirements through the live Visio interface with Microsoft Office Visio. Each live Visio diagram is synchronized with the database and has a special stencil that contains shapes representing the object types. The live Visio stencil can be used to create, modify, and delete the following:

- Objects in the Architect/Requirements client. The stencil includes shapes for folders, requirements, building blocks, groups, notes, spreadsheets, and diagrams.
- Trace links between objects represented in the diagram. Defining and complying trace links can be managed from the diagram. Trace links can be viewed in the **Trace** subtab of the **Links** tab and window.
- Connections between objects represented in the diagram. Connections can show any type of relationship, such as a physical relationship, a functional relationship, a data flow, or a control flow. Connections can be viewed in the **Relations** subtab of the **Links** tab and window.



While copying from a source diagram, to preserve the layout and retain the members in a different diagram, you can create a new diagram and copy the source diagram to the new diagram using the **Copy** and **Paste from TcSE** commands.

Every member in a diagram is linked to the diagram. When you move a member out of the owner of the diagram (for example, to a folder), the link with the diagram is removed.

Similarly, if the diagram is moved to another owner (for example, to a new folder), the link with its owner is removed.

In the above cases, the Visio diagram deletes all its contents while synchronizing the diagram because the old links are not present. All the members of the new owner of the diagram (in this case the new folder) are created in the diagram while synchronizing the diagram.

Properties of Architect/Requirements objects are mapped to the **Text** properties of Visio shapes through the live Visio stencil. For more information about customizing a stencil, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



- Microsoft Office Visio 2013 or Visio 2016 must be installed.

Microsoft .NET Framework 2.0 and 3.0 or 3.5 must be installed on your computer.

- To change editable property values for objects, you must have the **Modify** permission for the objects and the **Architect** privilege for the project.
- This section assumes that you have experience with Microsoft Office Visio.

User Actions in Live Visio Diagrams

You can create, view, and edit live Visio diagrams using the live Visio interface. This sections lists the various user actions available as pop-up menu commands in the live Visio diagram.

When a live Visio diagram is opened in the Microsoft Office Visio, right-click on the diagram and choose from any of the following pop-up menu commands:

- **Paste From TcSE** to paste the content that was copied from the Architect/Requirements client to the live Visio diagram.
- **Paste and Create In TcSE** to copy any existing object from the same or a different live Visio diagram and create a new object of the same type in the live Visio diagram as well as in the Architect/Requirements client.
- **Make Visio Diagram Non Live** to convert a live Visio diagram to a non-live Visio diagram. After making the diagram non-live, it would not connect with the Architect/Requirements client. You can use this command on a saved diagram on a local disk.
- **Get Connection From Visio Diagram** to copy the existing connection on a live Visio diagram and create a new connection on an existing or a new diagram. To use this command, the two ends to the connection must exist on the diagram.
- **Refresh Entire Visio Diagram** to synchronize the Visio diagram from the Architect/Requirements server. The connections that are not added from Visio are not synchronized.
- **Refresh With Connection** to synchronize the Visio diagram from the Architect/Requirements server with the connections that are not added from Visio.
- **Select Non TcSE Shapes** to select the non-Architect/Requirements shapes on a live Visio diagram.
- **Go To Parent Diagram** to open the parent diagram, if it exists.
- **Hide/Unhide Port** to hide or unhide the port shape on a live Visio diagram.
- **Hide/Unhide Port Name** to hide or unhide the port shape name on a live Visio diagram.
- **Paste as Tunnel Port** to create the port as a tunnel port on a diagram. However, only an external port shape can be pasted as a tunnel port. No new objects are added in the database.
- **Hide/Unhide Connection Name** to hide or unhide the connection shape name on a live Visio diagram.
- **Hide/Unhide Marked Connection Name** to hide or unhide the marked connection name of the shape on a live Visio diagram. Marked connections are the connections selected by using the **Hide/Unhide Connection Name** command.
- **Reset Marked Connection** to reset the marked state of connection to unmarked state.
- **Upgrade Diagram** to recreate the right-click menu options on each shape present on the Visio diagram. **Upgrade Diagram** should be used only if you are working on a Visio diagram that was created with Architect/Requirements 2007.0 or earlier. This action may take considerable time depending on size of the Visio Diagram that you are working with.

When a live Visio diagram is opened in the Microsoft Office Visio, right-click a shape (an object) and choose from any of the following pop-up menu commands.

- **TcSE Properties** to open a dialog window displaying all the properties of the object in Architect/Requirements.
- **Start Link** to start the trace link from the selected object.
- **End Link** to end the trace link at the selected object.
- **End Link Using Subtype** to end the trace link at the selected object as the chosen subtype.
- **Go To TcSE** to navigate to the corresponding object in the Architect/Requirements client.

- **Delete in TcSE** to delete the corresponding object in the Architect/Requirements database.

When a live Visio diagram is opened in Microsoft Office Visio, right-click a shape (an object) and choose from any of the following pop-up menu commands, depending on the object type of the shape.

- **Open Child Diagram** to open or create a live Visio diagram on the selected object. This command is available only on the building block shape.
- **Open Child Diagram w/Stencil** to open or create a live Visio diagram on the selected object using the selected stencil. This command is available only on the building block shape. If no stencil is selected, the stencil of the parent diagram is chosen.
- **Reverse Direction** to reverse the direction of a connection. This is available only on the connection shape.
- **Copy Connection** to copy the connection to the clipboard from a live Visio diagram. This command should be used along with the **Get Connection From Visio Diagram** command.
- **Set Data Definition** to associate a data definition with a connection or a port shape. This command is available only on a connection and a port shape.
- **Unassign Data Definition** to remove a data definition association with a connection or a port shape. This command is available only on a connection and a port shape.
- **Copy Port** to copy and create a tunnel port using the **Paste as Tunnel Port** command. The **Copy Port** command is available only on an external port shape.

To work with ports and connections in a live Visio diagram, you can do any of the following. For more information, see [Ports and Connections](#) later in this chapter.

- Create a port by selecting the port shape from the stencil and attaching it to a building block connection point. On successful attachment, a port object is created in the Architect/Requirements database and the building block's property is updated on the live Visio diagram.
- Move a port by selecting the port shape and move it on the same owner (building block). Moving a port to a different owner is not permitted. Also, if you have configured the port to be attached at any specific location on a building block, then the move honors the configuration on the property mapping XML file.

Moving a port having a connection attached to it moves the connection along with the port.

- Create a connection by selecting the connection shape from the stencil and attaching it on any two building blocks. It creates ports on the respective building blocks and creates connection between them. Ports and connection objects are created in the Architect/Requirements database.
- Create connection between ports by selecting the connection shape from the stencil and attaching it to any two ports. It creates the connection in the Architect/Requirements database and synchronizes the connection's properties in the Microsoft Office Visio.
- Move a connection by changing its two ends. You can detach one end of the connection and attach it to any other shape (building block). The connection between two ends is updated in the Architect/Requirements database.
- You cannot detach a port from a building block. If it is no longer required, you can delete it using the **Delete in TcSE** command.
- You cannot detach a connection from a building block or a port, and a warning message is displayed if you try. If it is no longer required, you can delete it using the **Delete in TcSE** command.

Stencil Diagnostic Utility

Stencil diagnostic utility is available as the following three commands when you right-click on a live Visio diagram:

- **Identify Subshape** to identify the subshapes in a group master shape. You can also identify the sequence in which you have added shapes to a group master shape.
- **Diagnose Connection Point** to diagnose the connection point section on a master shape.
- **Verify Prop Map XML And Stencil** to verify the property mapping XML file, the stencil file master shape mapping, and the availability of the mapped objects in Architect/Requirements. For more information about the property mapping file, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

Creating a Diagram

Diagrams can be attached to building blocks, folders, requirements, and groups. For the object that you select, the live Visio interface generates a diagram containing a shape for each member of the object. Members of a folder, a requirement, or a building block are its direct children. Members of a group are objects that belong to the group while residing in their native locations. For an object of any of those types, the members are shown at the next lower level. For more information, see [Viewing Objects in a Hierarchy](#) in chapter 3, *Using the Architect/Requirements Main Window*.

The object to which you attach the diagram is the diagram owner. A shape for that object does not appear in the diagram. You can select the owner in the hierarchical content table or in either subtab of the **Links** tab or window. In the **Trace** subtab, you can select from either pane. In the **Relations** subtab, you can select from either object table but not from the links tables. For more information, see [Hierarchical Content Table](#), [Links Tab](#), [Trace Subtab](#), [Relations Subtab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Existing connections between the members are included in the diagram. Connections are attached correctly on each member, and you can adjust them for the desired appearance. Connections can be viewed in the **Relations** subtab. For more information, see [Working With Connections in a Live Visio Diagram](#) and [Viewing the Connections for an Object](#), later in this chapter.

Existing trace links between the members are not included in the diagram. However, existing trace links can be added to diagrams, and new trace links can be created from Visio. For more information, see [Working With Trace Links in a Live Visio Diagram](#), later in this chapter.

The diagram object is added to the Architect/Requirements database and is displayed in the **Attachments** tab or window. For more information, see [Attachments Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.



You must have **Architect** privilege for the project.

To create a diagram:

1.

Select the diagram owner, and then pull down the **File** menu and choose the **Visio Live**→**Create Diagram** options.

The Diagram Inputs dialog window is displayed.

2. Do one or both of the following:

- Check the check box for each stencil whose shapes you want to include in the diagram. You can check all check boxes by clicking **Select All**.
- Clear the check box for each stencil whose shapes you want to exclude from the diagram. You can clear all check boxes by clicking **Unselect All**.



To use ports in the diagram, you must select a stencil that supports ports, such as the default **Port Type Stencil**.

You can also do the following:

- In the **Diagram Subtype** field, select a subtype to assign to the diagram.
- In the **Data Dictionary** field, select a data dictionary to attach to the diagram.

3. Click **OK**.

The diagram opens in a live Visio window, containing a shape for each member of the owner. The shapes are automatically arranged on the Visio page so that all are visible with no overlapping.



When you first save the diagram, if its **Copy Snapshot** property value is **Yes**, Architect/Requirements automatically attaches a note to the diagram. Named **Diagram Image** by default, this note contains an image (.gif) of the diagram. The diagram image note is updated each time changes to the diagram are saved. Through this note, the diagram's full content can be targeted by reference links.

Generating the note and updating the image can add time in saving a diagram. Siemens PLM Software recommends that you leave **Copy Snapshot** set to **No**, the default, until just before you create the first reference link to the diagram image note. For more information, see [Creating a Full Content Reference Link](#) in chapter 9, *Working With Object Properties*.

To disable updates to the image note, you can change **Copy Snapshot** from **Yes** to **No**. For more information, see [Editing the Properties of a Selected Object](#) in chapter 9, *Working With Object Properties*.

You can open the diagram image note in Word and resize the image. The new dimensions are applied to the image each time it is updated from Visio. You may want to resize the image periodically for consistency with the Visio diagram layout.

To save the diagram outside the Architect/Requirements database, click the Visio **File** menu and choose **Save As** to display the **Save As** dialog window, where you can store the diagram on a local or shared drive. Later, you can open the diagram and synchronize it with the database by pulling down the Architect/Requirements **File** menu and choosing the **Visio Live**→**Open** options to display the **Open** dialog window.



A live Visio diagram that is stored outside the database can be converted to a standard Visio file, one that is static and disconnected from the database. For more information, see [Disconnecting a Live Visio Diagram](#), later in this chapter.

Procedure Notes

Step 1: Except in the **Links** subtabs and windows, you can right-click the diagram owner and choose the **Visio Live**→**Create Diagram** options from the pop-up menu.

Creating a Child Diagram

A child diagram is a secondary part of a live Visio diagram. You can create additional diagrams using the stencils in the database to create a more defined structure or picture of the information needed in the diagram. If you want to use ports and paths, use a stencil that contains these objects.



By default, a child diagram inherits the stencils and data dictionary of the parent diagram.

To create a child diagram:

1. Select a shape in Visio that represents a building block that is a child of the diagram owner.
2. Right-click the object and select **Open Child Diagram** from the pop-up menu.

The object opens in a new drawing window. If the stencil set of the parent object is not visible, it is in the background of the child diagram and you must resize the new window to use the drawing tools of the parent. Any ports from the parent are visible in the child diagram when it is first opened.

Editing a Diagram

Live Visio diagrams are synchronized with the Architect/Requirements database. When members of a diagram owner are added, modified, or deleted in the Architect/Requirements client, the diagram is dynamically updated with those changes. Conversely, objects are automatically added, modified, or deleted in the client when those actions are done for corresponding shapes in the diagram.

You can use shapes from multiple stencils in the same diagram, although Architect/Requirements stencils contain the only shapes that apply to the database. If you add, modify, or delete shapes from other stencils, such as those provided with Microsoft Office Visio, the database is not affected. Shapes from non-Architect/Requirements stencils are saved within the diagram itself.

To display live Visio shapes, pull down the Visio **Window** menu and choose the Architect/Requirements stencils that you want to use.



- The arrangement of shapes in the diagram does not reflect the hierarchy or the display order of the objects in the Architect/Requirements client. Shapes can be placed anywhere on the Visio page without affecting the objects.
- For the objects represented by the shapes, you can create new diagrams in additional Visio windows. By right-clicking a shape and choosing **Create New Diagram** from the pop-up menu, you open a diagram containing a shape for each member of the object, which becomes the diagram owner. You can rename the diagram in the **Attachments** tab or window.

•

To navigate to an object in the Architect/Requirements client, you can right-click the corresponding shape in the diagram and choose **Go To TcSE** from the pop-up menu.

- A given Architect/Requirements object can appear any number of times in a live Visio diagram.

When the diagram owner is selected, its diagrams are displayed in the **Attachments** tab or window. For a selected member of the owner, the **Where Used** tab or window displays each diagram that contains a shape representing the member object. You can open a diagram for editing from either view. Also, you can open and edit a live Visio diagram that was saved locally on your computer or on a shared drive. For more information, see [Attachments Tab](#), [Where Used Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



You must have **Architect** privilege for the project.

To edit a diagram:

1. Open the diagram by doing one of the following:
 - For a diagram displayed in the **Attachments** or **Where Used** tab or window, select the diagram, and then pull down the **File** menu and choose **Open**.

Live Visio opens with a message stating that the diagram is being refreshed. When the message closes, the diagram is synchronized with the database.

If members of the owner were added, modified, or deleted since the diagram was last saved, the diagram is updated with the changes. By default, new members are cascaded in the lower left corner of the page. Go to step 2.

- For a diagram that is stored on a local drive:

. With any object selected in the content table or the **Links** tab or window, pull down the **File** menu and choose the **Visio Live→Open** options.

A new live Visio session starts with the Open dialog window, which lists existing folders and files in the current drive or folder. If the list does not display the diagram, you can use the **Look in** field to change the drive or folder, or use the **Files of type** field to see other types of files.

. Select the diagram, and then click **Open**.

A message states that the diagram is being refreshed. When the message closes, the diagram is synchronized with the database.

If members of the owner were added, modified, or deleted in the database since the diagram was last saved, the diagram is updated with the changes. By default, new members are cascaded in the lower left corner of the page.



If you open a live Visio diagram while the Architect/Requirements client is not running, a message asks if you want to connect directly to the server. If you click **Yes**, you are prompted to log in, and then to connect to the server. If you click **No**, the diagram is not refreshed, and changes to the diagram are not applied in the database.

2. In the diagram, do any or all of the following:

- To add a shape and a new member of the diagram owner:
 - . In an Architect/Requirements stencil, point to an appropriate shape for the object type and hold down the left mouse button.
 - . Move the pointer to the Visio page, and then release the mouse button.

The shape appears with selection handles and a rotation handle. In the database, a corresponding object is created as a direct child or member of the diagram owner.

- To add a shape for an object that is not a member of the diagram owner:
 - In the Architect/Requirements client, select the object, and then pull down the **Edit** menu and choose **Copy**.

You can also right-click the object and choose **Copy** from the pop-up menu, click the **Copy** button on the toolbar, or press control-C.

- In the diagram, right-click a blank area on the page, and then choose **Paste From TcSE** from the pop-up menu.

The shape appears on the top of the stack at the lower left corner of the page. However, an object is not created in the database.



A given Architect/Requirements object can appear any number of times in a live Visio diagram.

- To copy an existing shape and create a new member of the diagram owner:

- Select the shape, pull down the Visio **Edit** menu, and choose **Copy**.

You can also right-click the object and choose **Copy** from the pop-up menu, click the **Copy** button on the Visio toolbar, or press control-C.

- Right-click a blank area on the Visio page, and then choose **Paste And Create In TcSE** from the pop-up menu.

The shape appears with selection handles and a rotation handle. In the database, the object is created as a member of the diagram owner.



A given Architect/Requirements object can be copied any number of times in a live Visio diagram.

- To create a new diagram for an object represented by a shape, right-click the shape and choose **Create New Diagram** from the pop-up menu.

The diagram opens in a new Visio window, with a shape for each member of the object. That object is the owner of the diagram, which you can rename in the **Attachments** tab or window.

- To change the properties of an object in the Architect/Requirements client:

On the Visio page, right-click the shape for the object, and then choose **TcSE Properties** from the pop-up menu.

Live Visio displays the Teamcenter for systems engineering properties dialog window. Values that you cannot change are dimmed in the **Value** column.

- In the **Value** column, double-click a value that you want to change.

Depending on the property type, a text field or a list field opens.

- Enter the new value in the text field or select the new value in the list field, and then click outside the field.

For each additional value that you want to change, repeat steps b and c.

- . To apply the changes and close the dialog window, click **OK**.
- To rename an object in the Architect/Requirements client:
 - . Select the shape for the object, and then do one of the following:
 - If the object name is mapped to the shape itself, select the shape and press the **F2** key to open a text field.
 - If the object name is mapped to the subshape that contains the name, double-click the shape to open a text field.



The **Name** property of the object in the database is mapped to the **Text** property of the shape through the live Visio stencil. For more information about customizing a live Visio stencil, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about stencils, consult your project administrator.

- . Enter the new name in the text field, and then click outside the shape.
- To create a trace link between objects in the Architect/Requirements client:
 - . Right-click the shape that represents the starting object, and then choose **Start Link** from the pop-up menu.
 - . Right-click the shape that represents the ending object, and then do one of the following:
 - For the default subtype, choose **End Link** from the pop-up menu.
 -

For a user-defined subtype:

 - . Choose **End Link Using Sub-type** from the pop-up menu to display the Select Subtype dialog window.
 - . Select a trace link subtype, and then click **OK** to close the dialog window.

Live Visio displays a message confirming that the link is created. When the starting or ending object is selected in the Architect/Requirements client, the trace link is displayed in the **Links** tab or window. For more information, see [Links Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

- To delete a shape and the corresponding object, right-click the shape and choose **Delete in TcSE** from the pop-up menu.



This option does not support the selection of multiple objects.

The shape is removed from the diagram, and the object is moved to your Architect/Requirements Recycle Bin. You can restore the object until your Recycle Bin is emptied. For more information, see [Restoring Objects](#) or [Emptying Your Architect/Requirements Recycle Bin](#) in chapter 4, *Maintaining a Project*.

- To delete a shape without deleting the corresponding object, select the shape, and then click the **Delete** button on the Visio toolbar or press the delete key.



If the object is a member of the diagram owner, live Visio displays a message stating that the diagram type will be changed to **Static** and asks if you want to continue. Click **Yes** to remove the shape.

The shape is removed from the diagram and the object remains in the client.

If the object is a member of the owner, the diagram's **Diagram Content** property value is changed from **Members** to **Static**. For more information, see [Changing a Diagram Type](#), later in this chapter.



The **Delete** button removes the trace link from the diagram. If you want to remove the trace link from Architect/Requirements, you must select the trace link and click **RMB→Delete In TcSE**.

3. To save your changes, pull down the Visio **File** menu and choose **Save**.

Also, you can save the diagram outside the database by pulling down Visio's **File** menu and choosing **Save As** to display the Save As dialog window. Later, you can open the diagram and synchronize it with the database by pulling down the **File** menu and choosing the **Visio Live→Open** options.

Procedure Notes

Step 1: For a diagram displayed in the **Attachments** or **Where Used** tab or window, you can also right-click the diagram and choose **Open** from the pop-up menu, or double-click the diagram. For a diagram that is stored on a local drive, you can also right-click anywhere in the content table or the **Links** tab or window, and choose **Visio Live→Open** from the pop-up menu.

Copying a Microsoft Office Visio Diagram to Live Visio

Elements of a Microsoft Office Visio diagram can be copied to a live Visio diagram. This process allows you to reuse Visio content to create objects in the Architect/Requirements database.

The source diagram is a standard Visio file that is stored outside the database, for example, on a local or shared drive. You use a special live Visio function to copy the source to the destination, either a new live Visio diagram or one stored in or outside the database. For more information, see [Creating a Diagram](#), earlier in this chapter.



Shapes in the source diagram must be represented in the mapping file for the live Visio stencil. A mapping file assigns shapes in the source to object types and properties in the database. For more information about configuring a mapping file, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about stencils or mapping files, consult your project administrator.



- You must have **Architect** privilege for the project.
- Microsoft Office Visio 2013 or Visio 2016 must be installed.

To copy a Microsoft Office Visio diagram to live Visio:

1. In the source file, select all or any portion of the diagram, and then pull down the Visio **Edit** menu and choose **Copy**.



To import a connection, the starting and ending objects must be selected with the connection shape. Tree connections are not supported.

2. In the live Visio file, right-click the page and choose **Paste And Create In TcSE** from the pop-up menu.



Use only this live Visio option to paste the selection. The standard Visio paste function may give unpredictable and undesired results.



This option is supported only on the first page of the live Visio diagram.

A progress indicator is displayed while shapes are pasted and objects are created. A second progress indicator shows that the server is synchronizing changes.



Do not perform any action in the live Visio file while either progress indicator is displayed. Otherwise, the file may be corrupted.

Live Visio displays the copied shapes initially with Visio selection indicators. For each shape represented in the mapping file, an object is created in the database and displayed in the content table.

Objects are not created for shapes that are not represented in the mapping file. These shapes are highlighted in red, and you can change the color through the standard Visio functions. You can select all of these shapes by right-clicking the live Visio page and choosing **Select Non TcSE Shapes** from the pop-up menu.

Changing a Diagram Type

For every live Visio diagram, the **Diagram Content** property controls the behavior of the shapes that represent the members of the diagram owner in the Architect/Requirements client. This property is displayed in the Edit Properties dialog window and in the **Properties** floating window. You can edit this property for a diagram selected in the **Attachments** or **Where Used** tab or window. For more information, see [Properties Tab](#), [Attachments Tab](#), [Where Used Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



You must have **Architect** privilege for the project.

To change a diagram type:

1. Do one of the following:

- To use the Edit Properties dialog window, select the diagram, and then pull down the **File** menu and choose **Properties**. Or, right-click the diagram and choose **Properties** from the pop-up menu.
- To use the **Properties** floating window:
 - . With the **Properties** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Properties** window.
 - . Select the diagram in the **Attachments** or **Where Used** tab.

2.

In the **Value** column, double-click the **Diagram Content** property value to display the Single-Choice dialog window.

3. Check the check box for one of the following diagram types:

- **Members** fully synchronizes the set of shapes with the owner's member set. As members are added to the owner and moved to other owners in the client, corresponding shapes are added and removed in the diagram.
- **Static** prevents the automatic addition and removal of shapes as the owner's member set changes. When members are added to the owner and moved to other owners in the client, the set of shapes is unchanged. However, the diagram remains connected to the database.
 - . Shapes for new members can be manually added to the diagram, thus creating new objects in the client.
 - . When object properties are changed in the client, corresponding shapes reflect the changes automatically.

Any shape can be manually removed from a **Static** diagram, but the corresponding object is not deleted in the client.

For both diagram types, shapes are automatically deleted when corresponding objects are deleted in the client.



A **Members** diagram is changed to **Static** when a shape is added for a non-member object or when a member shape is removed. For more information, see [Editing a Diagram](#), earlier in this chapter.

4. Click **OK** to close the dialog window.

Disconnecting a Live Visio Diagram

Live Visio diagrams can reside outside the Architect/Requirements database, for example, on a local or shared drive. Any such diagram can be converted to a standard Visio file, one that is permanently disconnected from the database.



A disconnected diagram can no longer be used interactively with the Architect/Requirements client.



Microsoft Office Visio 2013 or Visio 2016 must be installed on your computer.

To disconnect a live Visio diagram:

1. In the diagram, right-click the Visio page and choose **Make Visio Diagram Non Live** from the pop-up menu.



If the diagram is not stored outside the database, pull down the Visio **File** menu and choose **Save As**, and then use the Save As dialog window to specify the drive or folder.

A confirmation message states that the diagram cannot be converted to a live diagram again.

2. To continue, click **Yes**.



This action is permanent and cannot be reversed.

The diagram is disconnected. The current objects and properties remain in the file.

Viewing a Diagram if Microsoft Office Visio Is Not Installed

If Microsoft Office Visio is not installed on your computer, you can open a live Visio diagram through the Microsoft Office Visio Viewer. An ActiveX control, the Visio Viewer displays Visio diagrams in Microsoft Internet Explorer.

You can view and print the diagram, but you cannot edit the diagram in the Visio Viewer. Furthermore, the diagram is not synchronized with the Architect/Requirements database.



- Microsoft Internet Explorer version 7 or 8 (32 bit) must be installed on your computer.
- Microsoft Office Visio Viewer must be installed with Internet Explorer. If necessary, you can download the latest version of the Visio Viewer from the Microsoft Download Center at <http://office.microsoft.com/en-us/officeupdate/default.aspx>.

To view a diagram if Microsoft Office Visio is not installed:

Open the diagram by doing any of the following:

- In the Architect/Requirements client, select the diagram, and then pull down the **File** menu and choose **Open**. You can also right-click the diagram and choose **Open** from the pop-up menu. Or, double-click the diagram.
- In Internet Explorer, pull down the **File** menu and choose **Open**. Then, select the diagram in the Open dialog window.



You can also use the mouse to drop the diagram in the Internet Explorer window.

- In Windows Explorer, double-click the diagram. Or, select the diagram, pull down the **File** menu, and choose **Open**.
- In an E-mail message with the diagram attached, double-click the attachment. Or, right-click the attachment and choose **Open** from the pop-up menu.

The Visio Viewer displays the diagram in Internet Explorer.

- To pan and zoom in the drawing window, you can use toolbar buttons, keyboard shortcuts, or options in the pop-up menu.
- To view the properties of a shape, you can open the Properties and Settings dialog window and then select the shape.

Also in the Properties and Settings dialog window, you can do the following:

- Make rendering and display settings in the **Display Settings** tab.
- Set drawing layer visibility and colors in the **Layer Settings** tab.
- Set annotation visibility and colors in the **Markup Settings** tab.

Data Dictionaries and Data Definitions

Creating a Data Dictionary

A data dictionary is a special folder subtype for containing data definitions. You can create a data dictionary directly below the project node or in a folder at any lower level. Then, you can attach the data dictionary to a diagram in the **Attachments** tab or window. For more information, see [Attaching a Data Dictionary to a Diagram](#), later in this chapter.

To create a data dictionary:

1. In the navigation tree or the hierarchical content table, select the containing folder for the data dictionary.
2. Pull down the **File** menu and choose the **New→Subtype** options.
The Select Subtype dialog window is displayed.
3. Under **Folder**, select the **Data Dictionary** subtype, and then click **OK**.
The content table displays the new data dictionary with a default name in an open text field.
4. Enter a meaningful name for the data dictionary, and then press the enter key.

Procedure Notes

Step 1: You can also right-click the containing folder and select **New→Subtype** from the pop-up menu. Or, click the **Create a new subtype** button on the toolbar or press control-U.

Step 3: To reverse this action, you can pull down the **Edit** menu and choose **Undo Create Subtype**, click the **Undo** button on the toolbar, or press control-Z.

Attaching a Data Dictionary to a Diagram

To assign data definitions to connections in a diagram, you must first attach a data dictionary to the diagram. If a diagram already has a data dictionary, you can attach a different one and assign those data definitions.

To attach a data dictionary to a diagram:

1. In the **Attachments** tab or window, select the diagram, and then pull down the **File** menu and choose **Properties**.
The Edit Properties dialog window is displayed.
2. In the **Value** column, double-click the **Data Dictionary** value.
The Single-Choice dialog window is displayed.
3. Check the check box for the data dictionary, and then click **OK**.
4. Click **Close** to close the Edit Properties dialog window.

Creating Data Definitions

A data definition is a building block subtype that describes a connection between other building blocks. When assigned to a connection in a live Visio diagram, the data definition's user-defined properties provide information about the connection.

Data definitions reside in data dictionaries and can be arranged in hierarchies, with multiple levels of parents, children, and siblings. You can create these hierarchies in the Architect/Requirements client and in a live Visio diagram.

In the Architect/Requirements client, you can create shortcuts to data definitions. These shortcuts allow you to reuse data definitions in a data dictionary. For more information, see [Working With Shortcuts](#) in chapter 4, *Maintaining a Project*.



- You must have **Architect** privilege for the project.
- You must have **Modify** permission for the data dictionary in which you intend to create the data definition.

Creating a Data Definition in the Architect/Requirements Client

As with a building block in a folder, you can create a data definition at any level in a data dictionary hierarchy.

1. To set the data definition's level in the hierarchy, do one of the following:
 - For the top level of a data dictionary, select the data dictionary in the navigation tree or the content table.
 - For the same level as another data definition, select the sibling in the content table.
 - For a child of another data definition, select the parent in the content table.
2. Pull down the **File** menu and choose the **New**→**Subtype** options.
The Select Subtype dialog window is displayed.
3. Under **Building Block**, select the **Data Definition** subtype, and then click **OK**.
The content table displays the new data definition with a default name in an open text field.
4. Enter a meaningful name for the data definition, and then press the enter key.

Creating a Data Definition in a Live Visio Diagram

While you build a diagram, you can create data definitions and attach them to connections. Through the Data Definitions dialog window in live Visio, you can create a data definition at any level of a data dictionary hierarchy. Each new data definition is automatically added to the Architect/Requirements client and database.

1. Right-click a connection and choose **Set Data Definition** from the pop-up menu.

The Data Definitions dialog window displays the current hierarchy for the data dictionary that is attached to the diagram. If the data dictionary contains data definitions, the tree is expanded with the data dictionary at the peak. If it does not contain data definitions, only the data dictionary is displayed.



If other users add or delete data definitions while the dialog window is open, the changes are not reflected in the tree. You can click **Refresh** to ensure that the tree shows the current hierarchy in the data dictionary.

Or, you can check the **Auto Refresh** check box to update the tree each time you create a data definition. However, each update involves a server call, which may affect performance in some cases, for example, a large number of data definitions in a data dictionary or a database on a remote server.

2. In the tree, select the node below which you want to create the data definition.

To navigate to a data definition in the client, you can right-click the data definition in the tree and choose **Go To TcSE** from the pop-up menu.



The tree may also display shortcuts to data definitions. You cannot create a data definition below a shortcut.

3. In the **Name** field, enter a meaningful name for the data definition.

In the **Subtype** field, you can select the subtype for the data definition.

4. Click **Add**.

The tree displays the new data definition at the specified level. In the client and in the database, the data definition is added to the data dictionary hierarchy.



The Data Definitions dialog window remains open. You can repeat steps 2 through 4 to create another data definition.

5. Do one of the following:

- To attach a data definition to the selected connection:

- . In the tree, select the data definition.
- . Click **Attach**.

The Data Definitions dialog window closes, and the data definition name appears on the connection.

- To close the Data Definitions dialog window without attaching the data definition to a connection, click **Exit**.

Ports and Connections

A port is a connection point through which data is communicated across a boundary of a physical system. Connections show the flow of data between two ports.

Creating Ports

In the Architect/Requirements client, you can create a port on any building block or its subtype, including a data definition. In a live Visio diagram, you can create a port on any shape that is mapped to a building block.



- You must have **Architect** privilege for the project.
- You must have **Modify** permission for the building block.

Create a port in the Architect/Requirements client

1. In the content table, select the building block on which you want to place the port.
2. Assign the object type by doing one of the following:
 - For the base type, **Port**, pull down the **File** menu and choose the **New→Port** options.
 - For a user-defined subtype:
 - . Pull down the **File** menu and choose the **New→Subtype** options to display the Select Subtype dialog window.
 - . Under the **Port** type, select the subtype and click **OK**.

If you have questions about subtypes, consult your Architect/Requirements project administrator.

The **Connectivity** tab or window displays the port with a default name in an open text field.

3. Enter a meaningful name, and then press the enter key.

The port is created also in diagrams where the building block is represented.

Procedure Notes

Step 2: You can also right-click the building block and choose the options from the pop-up menu.

Create a port in a live Visio diagram

1. Point to a port shape in the stencil pane.

The shape can represent the base type, **Port**, or a user-defined port subtype. If you have questions about port shapes or subtypes, consult your Architect/Requirements project administrator.

2. Drop the shape on a connection point on the building block shape.

The port is created also in the Architect/Requirements client.

Creating Connections in the Architect/Requirements Client



You must have **Architect** privilege for the project.

1. Select the starting object or objects, and then pull down the **Edit** menu and choose the **Connections**→**Start Connection** options.



- If you select only one starting object, you can select multiple ending objects. If you select multiple starting objects, you can select only one ending object.
- To connect objects in different projects, and if the target project is in a different Architect/Requirements client running in a separate window, you must use the mouse to create the connections. Hold down the left mouse button on the starting selection, move the pointer to the target project in its client window, and then release the mouse button to complete this procedure.

You can also right-click the selection and choose the **Connections**→**Start Connection** options from the pop-up menu. Or, click the **Start a connection** button on the toolbar.

2. Select the ending object or objects.

If only one starting object is selected, you can select multiple ending objects. If multiple starting objects are selected, you can select only one ending object.



If you are connecting objects in different projects, navigate to the target project and select the ending object or objects. For connections in the same project, you can open a different folder and select the ending object or objects.

3. Assign the subtype by doing one of the following:

- To assign the default subtype (**Connection**), pull down the **Edit** menu and choose the **Connections**→**End Connection** options.

You can also right-click the selection and choose the **Connections**→**End Connection** options from the pop-up menu. Or, click the **End a connection** button on the toolbar.

- To assign a user-defined subtype:
 - Pull down the **File** menu and choose the **Connections**→**End Connection Subtype** options.



You can also right-click the selection and choose the **Connections**→**End Connection Subtype** options from the pop-up menu. Or, click the **End a connection subtype** button on the toolbar.

The Select Subtype dialog window displays the connection subtypes for the project.

- Select a subtype, and then click **OK** to close the dialog window.

For starting or ending objects that are ports, the connections are displayed in the **Connectivity** tab or floating window.

For ending objects other than ports, the **Start** pane in the **Relations** subtab displays starting objects in the left table and starting connections in the right table. If you linked to multiple starting objects, select one at a time to see this information.

For starting objects other than ports, the **End** pane in the **Relations** subtab displays ending objects in the left table and ending connections in the right table.

Procedure Notes

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo End Connection**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.



The copy, cut, and paste operations are disabled for port and connection objects in the Architect/Requirements client. The corresponding commands in the **Connectivity** tab and the **Edit** menu are disabled.

However, you can copy a connection to a live Visio diagram. For more information, see [Copying a Connection to a Live Visio Diagram](#) later in this chapter.

Working With Connections in a Live Visio Diagram

In a new or existing live Visio diagram, you can create connections between shapes that represent requirements, building blocks, and other elements of the system design. For each connection shape that you create in a diagram, a corresponding connection object is created in the Architect/Requirements database.

In the Architect/Requirements client, connections between ports are displayed only in the **Connectivity** tab or floating window. For objects other than ports, connections can be viewed in the **Relations** subtab of the **Links** tab and window. In the **Where Used** window, you can see the diagram that references the connection object selected in the **Relations** tab or window. You can also search for specific connection objects in the Search module. For more information, see [Viewing the Connections for an Object](#), later in this chapter; [Relations Subtab](#) and [Where Used Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*; and chapter 11, [Using the Search Module](#).



To navigate to an object in the client, you can right-click the corresponding shape in the diagram and choose **Go To TcSE** from the pop-up menu.

The connection shapes are from the Visio stencil selected when the diagram was created. The stencil consists of a **.vss** file, which determines the connection shapes themselves, and an **.xml** file that determines the information that is mapped to the shapes. Stencils for the project are maintained in the **Reports and Formatting** folder in the Administration module. If you have questions about stencils, consult your project administrator.



You must have **Architect** privilege for the project.

Creating a Connection in a Live Visio Diagram

1. Point to the connection shape in the stencil pane, hold down the left mouse button, and drop the shape on a connection point on the starting object.
2. Point to the end point (+) on the connection shape, hold down the left mouse button, and drop the end point on a connection point on the ending object.

Depending on where you place the connection shape in step 1, you may need to glue the begin point (x) to a starting connection point to create the connection object in the database.



- Multiple connections can be created between the same two objects. A name is automatically assigned to each connection.
- The begin point and the end point can be glued to the same object.
- You can also create a connection by copying and pasting an existing connection shape in the diagram, and then gluing it to the starting and ending objects.
- If the ending object is a building block, a port is automatically created on that object, and the connection end point is glued to the port.

Editing a Connection in a Live Visio Diagram

To reverse the direction between the same two objects, right-click the connection shape and choose **Reverse Direction** from the pop-up menu.

To change the starting or ending object, detach the begin point or the end point of the connection shape from the current connection point, and then glue the begin or end point to a connection point on another object.



If you drop the begin point or the end point anywhere other than on a connection point, live Visio displays a message stating that you are creating a dangling connection. If you click **Yes** to continue, the connection shape is removed from the diagram. The connection object remains in the Architect/Requirements client.

Copying a Connection to a Live Visio Diagram

You can copy a connection from the Architect/Requirements client to a live Visio diagram, and also from one diagram to another.

To copy a connection from the Architect/Requirements client to a diagram:

1.

In the **Relations** subtab, right-click the connection object and choose **Copy to Diagram** from the pop-up menu.

The connection object identifier is placed in the Windows clipboard.

2.

In the diagram, right-click the Visio page and choose **Paste From TcSE** from the pop-up menu.

The connection shape is added to the diagram and is attached to the correct points automatically.

To copy a connection from one diagram to another diagram:



The source diagram and the destination diagram need not be attached to the same owner. To get a connection, it is sufficient if two ends of the connection are present in the diagram.

1.

In the source diagram, right-click the connection shape and choose **Copy Connection** from the pop-up menu.

The connection object identifier is placed in the Windows clipboard.

2.

In the destination diagram, right-click the Visio page and choose **Get connection from Visio diagram** from the pop-up menu.

The connection shape is added to the diagram and is attached to the correct points automatically.

Deleting a Connection

You can delete connections from a live Visio diagram only, and also from both the diagram and the Architect/Requirements client at the same time.

To delete a connection from the diagram only:

Do one of the following:

- Select the connection shape, and then press the delete key or click the **Delete** button on the Visio toolbar.

The connection shape is removed from the diagram. The connection object remains in the Architect/Requirements client.

- Detach the begin point or the end point of the connection shape from the current connection point, and then drop the begin point or the end point anywhere other than on a connection point.

Live Visio displays a message stating that you are creating a dangling connection and asking if you want to continue. Click **Yes** to remove the connection shape. The connection object remains in the Architect/Requirements client.

To delete a connection from the diagram and the Architect/Requirements client:

Do one of the following:

- In the live Visio diagram, right-click the connection shape and choose **Delete in TcSE** from the pop-up menu.
- In the **Relations** subtab, select the connection object, and then pull down the **Edit** menu and choose the **Links**→**Delete Link** options. Or, right-click the connection and choose the **Links**→**Delete Link** or **Delete**→**Delete Link** options from the pop-up menu.

The connection shape is removed from the diagram and the connection object is moved to your Architect/Requirements Recycle Bin. The object remains in the database, and you can restore the connection until you empty your Recycle Bin. For more information, see [Restoring Objects](#) and [Emptying Your Architect/Requirements Recycle Bin](#) in chapter 4, *Maintaining a Project*.

Hiding and Showing Connection Names in a Live Visio Diagram

You can toggle connection names to make them visible or not visible.

- To hide a specific connection name, right-click the connection shape and choose **Hide/Unhide Connection Name** from the pop-up menu.



The first time a connection is hidden, it becomes a marked connection name. Marked names and unmarked names are toggled separately.

- To show a specific connection name, right-click the connection shape and choose **Hide/Unhide Connection Name** from the pop-up menu.
- To hide or show all unmarked connection names, right-click the Visio page and choose **Hide/Unhide Connection Names** from the pop-up menu.



Marked connection names remain in their current states.

- To hide or show all marked connection names, right-click the Visio page and choose **Hide/Unhide Marked Connection Names** from the pop-up menu.



Unmarked connection names remain in their current states.

- To unmark all marked connection names, right-click the Visio page and choose **Reset Marked Connection Names** from the pop-up menu.

The names are returned to the unmarked state and are shown on the connection shapes.

Attaching a Data Definition to a Connection

In a live Visio diagram, multiple connections can be glued to one building block or to one port on a building block. One data definition can be attached to each connection.

1. In the diagram, right-click the connection and choose **Set Data Definition** from the pop-up menu.

The Data Definitions dialog window displays the current hierarchy for the data dictionary that is attached to the diagram.



If other users add or delete data definitions while the dialog window is open, the changes are not reflected in the tree. You can click **Refresh** to ensure that the tree shows the current hierarchy in the data dictionary.

2. Do one of the following:

- To attach an existing data definition:

- . In the tree, select the data definition.

You can use the **Search** section to find the data definition in the tree:

- o

Enter the name in the **Search** field and click the **Search** button to highlight the data definition.

- o

Click **First** to move the selection to the top data definition.

- o

Click **Previous** to move the selection up by one data definition.

- o

Click **Next** to move the selection down by one data definition.

- o

Click **Last** to move the selection to the bottom data definition.

To navigate to a data definition in the client, you can right-click the data definition in the tree and choose **Go To TcSE** from the pop-up menu.

If the tree displays shortcuts to data definitions, you can select a shortcut and attach it to the connection.

- . Click **Attach**.

The Data Definitions dialog window closes, and the data definition name appears on the connection.

- To create and attach a new data definition:

- . Select the node below which you want to create the data definition.

- . Enter the name in the **Name** field.

You can also select a subtype in the **Subtype** field.

- . Click **Add**.

The tree displays the new data definition at the specified level.

- Click **Attach**.

The Data Definitions dialog window closes, and the data definition name appears on the connection.

Detaching a Data Definition From a Connection



You must have **Architect** privilege for the project.

In the **Connectivity** tab or window:

1. Select the connection to which the data definition is attached.
2. Pull down the **Edit** menu and choose the **Data Definition**→**Clear Data Definition Link** options.

You can also right-click the connection and choose **Data Definition**→**Clear Data Definition Link** from the pop-up menu.

In a live Visio diagram:

1. Right-click the connection to which the data definition is attached.
2. Choose **Un-Assign Data Definitiion** from the pop-up menu.

Viewing the Connections for an Object

For the object selected in the hierarchical content table, connections are displayed in the **Relations** subtab of the **Links** tab and window. The subtab shows connections to the object, connections from the object, and the starting or ending object for each connection.

In the **Links** window, the **Relations** subtab can show such information for the object selected in the **Where Used** or **Versions** tab. For more information, see [Hierarchical Content Table](#), [Relations Subtab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



Connections between ports are displayed only in the **Connectivity** tab or floating window.

To view the connections for an object:

Do one of the following:

- For an object in the hierarchical content table, select the object and click the **Links** tab, and then click the **Relations** subtab.

You can open the **Links** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.

- For an object in the **Where Used** or **Versions** tab:
 - With the **Links** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Links** window, and then click the **Relations** subtab.
 - Select the object in the **Where Used** or **Versions** tab.

In the **Start** pane, beginning connections to the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the object from which each beginning connection originates.

In the **End** pane, ending connections from the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the object at which each ending connection is completed.



In each pane, the rows in the object table and the link table remain synchronized at all times. For example, when you sort or scroll a link table, the corresponding object table is automatically sorted or scrolled to match the new view in the link table.

Procedure Notes

To see lower-level objects for selection in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click the object and choose **Expand All** from the pop-up menu.

To display the **Links** tab, you can also click the object's links indicator.

Working With Trace Links in a Live Visio Diagram

In a new or existing live Visio diagram, you can create trace links between shapes that represent requirements, building blocks, and other elements of the system design. For each trace link shape that you create in a diagram, a corresponding trace link object is created in the Architect/Requirements database.

In the Architect/Requirements client, trace links between objects can be viewed in the **Relations** subtab of the **Links** tab in the notebook pane. For more information, see [Overview of Trace Links](#) and [Creating Trace Links](#) in chapter 7, *Showing Object Relationships With Trace Links*.



You must have the **Architect** privilege for the project.

Creating a Trace Link in a Live Visio Diagram

1. Drag the trace link shape in the live Visio stencil pane to a connection point on the shape that represents the starting object.
2. Glue the trace link shape's end point to a connection point on the shape that represents the ending object.



The trace link is created in the database, but a trace link shape is not added to the diagram.

Copying a Trace Link to a Live Visio Diagram

1.

In the **Relations** subtab, right-click the trace link object and choose **Copy to Diagram** from the pop-up menu.

The trace link object identifier is placed in the Windows clipboard.

2.

In the diagram, right-click the Visio page and choose **Paste From TcSE** from the pop-up menu.

The connection shape is added to the diagram and is attached to the correct points automatically.

Chapter 7: Showing Object Relationships With Trace Links

This chapter discusses the role of trace links in joining objects in Architect/Requirements to one another within a single project, between two projects, and to objects in other Teamcenter products. Instructions are provided for creating, viewing, and deleting trace links, and for navigating to linked objects.

Overview of Trace Links

A trace link creates a directional relationship between two objects, a relationship conveyed by the terms *defining* and *complying*. A defining object specifies a condition that a product or a component must fulfill. A complying object partially or completely fulfills a condition specified by a defining object. Such a relationship establishes a path in which one object precedes the other.

For example, functional requirement *A*, for target tracking, defines hardware requirement *B*, for a CPU with a certain instruction rate. In this example:

- Requirement *A* precedes requirement *B*, with requirement *B* directly *downstream* in the complying path.
- Requirement *B* succeeds requirement *A*, with requirement *A* directly *upstream* in the defining path.

An object can assume both defining and complying relationships, continuing the path upstream and downstream. For example, a weight requirement may define a requirement to use aluminum. That complying requirement may in turn define temperature or environmental requirements consistent with the properties of aluminum. An object can have any number of defining and complying trace links.

Trace links can be created between objects within a single Architect/Requirements project, and also between objects in two different projects. In addition, trace links can be created from objects in Architect/Requirements to objects in other Teamcenter products, which are always complying objects. This two-way relationship allows for change analysis in both directions.

If the customer is considering a change to a product requirement, its trace links can be followed downstream, to all complying requirements that were derived from the product requirement, and eventually to all design elements that must comply. Conversely, an engineer working on a design element that must meet detailed, low level requirements can follow those trace links upstream, to the original customer requirements that define the design constraints.

Creating Trace Links

Trace links can be created from folders, requirements, building blocks, and groups to other objects of those types within Architect/Requirements. Also, objects in Architect/Requirements can be linked to objects in other Teamcenter products.

Linking Objects Within Architect/Requirements

You can link one defining object to one or more complying objects at the same time. Also by one action, you can link one or more defining objects to one complying object. However, multiple defining and complying objects cannot be linked simultaneously, and trace links cannot be linked to other trace links.

The defining and complying objects can reside together in the same Architect/Requirements project. Or, the complying objects can reside in a different project, in the same Architect/Requirements installation or in another installation. Defining and complying objects can be selected in the hierarchical content table and the **Links** and **Versions** tabs and floating windows. For more information, see [Hierarchical Content Table](#), [Links Tab](#), [Versions Tab](#), and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

In a given project, trace links within the project and interproject trace links are displayed together in the **Links** tab and floating window, the **Complying Object Traceability** view, and the Search Results dialog window. You can delete interproject trace links and navigate to linked objects in the same way as you do for those within a project. For more information, see [Viewing Object Relationships](#), [Deleting Trace Links for an Object](#), and [Navigating to a Linked Object](#), later in this chapter.

Each trace link receives certain system-defined properties, including the **Subtype** property. You can assign the default subtype, **Trace Link**, or a user-defined subtype created by your project administrator. Each trace link may also receive user-defined properties. The subtype and all other editable properties can be changed after the trace link's creation. For more information, see chapter 9, [Working With Object Properties](#) and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

You can create two or more trace links between the same objects if you assign a different subtype to each trace link. For example, defining requirement *A* is linked to complying requirement *B*, with the default subtype (**Trace Link**) assigned to the trace link. A second defining trace link, with user-defined subtype *SI*, can be created from requirement *A* to complying requirement *B*.

You cannot create and delete a trace link between two objects if they belong to any common baseline.

Subtypes also allow you to create trace links in both directions between the same objects. For example, defining requirement *A* is linked to complying requirement *B*, with subtype *SI* assigned to the trace link. By creating a trace link with user-defined subtype *S2*, you can link requirement *B*, as the defining object, back to requirement *A* as the complying object.



Only one trace link of a given subtype is allowed between the same two objects, regardless of the direction. However, you can create as many trace links between the objects as the number of available trace link subtypes.

Trace links can also be created in live Excel. For more information, see [Creating Trace Links in Live Excel](#) in chapter 9, *Working With Object Properties*.

To link objects in Architect/Requirements:

1. Select the defining object or objects, and then pull down the **Edit** menu and choose the **Links**→**Start Trace Link** options.



- If you select only one defining object, you can select multiple complying objects. If you select multiple defining objects, you can select only one complying object.
- If you are linking objects in different projects, and if the target project is in a different Architect/Requirements client running in a separate window, you must use the mouse to create the trace links. Hold down the left mouse button on the starting selection, move the pointer to the target project in its client window, and then release the mouse button to complete this procedure.

2. Select the complying object or objects.

If only one defining object is selected, you can select multiple complying objects. If multiple defining objects are selected, you can select only one complying object.



If you are linking objects in different projects, navigate to the target project and select the complying object or objects. For trace links within the same project, you can open a different folder and select the complying object or objects.

3. Assign the subtype by doing one of the following:

- To assign the default subtype (**Trace Link**), pull down the **Edit** menu and choose the **Links**→**End Trace Link** options, or click the **End Trace Link** button on the toolbar.



To assign a user-defined subtype:

- Display the Select Subtype dialog window by pulling down the **File** menu and choosing the **New**→**Subtype** options, by pressing control-U, or by clicking the **End Trace Link Subtype** button on the toolbar.

If you choose the **New**→**Subtype** options or press control-U, click the plus sign (+) to the left of **Trace Link** to see the user-defined subtypes. If you click the **End Trace Link Subtype** button, only trace link subtypes are shown, and user-defined subtypes are expanded automatically.

- Select a subtype, and then click **OK** to close the dialog window.

For each complying object, the objects in the defining path are shown in the **Defining Trace** column of the **Trace** subtab in the **Links** tab or window. In the **Relations** subtab, the **Start** pane shows defining objects in the left table and defining trace links in the right table. If you linked to multiple complying objects, select one at a time to see this information.

For a defining object, the objects in the complying path can be viewed in the **Complying Trace** column of the **Trace** subtab in the **Links** tab or window. In the **Relations** subtab, the **End** pane shows complying objects in the left table and complying trace links in the right table.

Procedure Notes

Step 1: You can also right-click the selection and choose the **Links**→**Start Trace Link** options from the pop-up menu. Or, click the **Start Trace Link** button on the toolbar.

Step 3: You can assign the default subtype also by right-clicking the selection and choosing the **Links**→**End Trace Link** options from the pop-up menu. To display the Select Subtype dialog window, you can also right-click the selection and choose **New**→**Subtype** from the pop-up menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo End Trace Link**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.



When a link is created, its name in the database is blank by default. On retrieving a link's name for display, if it is blank, a name is constructed from the names of the linked objects. In such case, changes to the names of the linked objects are reflected in the links name. You can also directly set a link's name. The links name is fixed after setting the links name and it does not use the linked object names. This behavior applies to all the user-visible link types, including trace links, connections, reference links, and generic links.

Linking to an Object in Another Teamcenter Product

You can create a trace link from a folder, a requirement, a building block, or a group in Architect/Requirements to an object in any of the following Teamcenter products:

- Engineering Process Management
- Teamcenter Enterprise
- Project

The object in Architect/Requirements is the defining object, and the trace link is bidirectional. In the other product's database, a complying trace link is created from the target object to the defining object.

You can select the defining object in the hierarchical content table or the **Links** tab or floating window. For more information, see [Hierarchical Content Table](#), [Links Tab](#), and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To link to an object in another Teamcenter product:

1. Select the defining object, and then pull down the **Edit** menu and choose one of the following options:

-

Links→**Link To Application**

Architect/Requirements displays the **Select Object Chooser** dialog window. Select a product from the list, and then click **OK**.

- **Links**→**Link To TcEngineering**
- **Links**→**Link To TcEnterprise**
- **Links**→**Link To TcProject**

The product's object chooser is displayed. If necessary, enter your user name and password for the product.

2. In the chooser, navigate to and select the target object, and then click **Done**.



This action clears the queue for the **Undo** option and cannot be reversed.

A confirmation message is displayed, and then the chooser closes. The **Links** tab or window displays the object in the **Complying Trace** column.

Procedure Notes

Step 1: You can also right-click the defining object and choose the options from the pop-up menu.

Viewing Object Relationships

A given object may have both defining and complying relationships with other objects. Defining objects reside in an Architect/Requirements project. Complying objects may reside in a project and in other Teamcenter products. You can view the objects in a defining or complying path, from the next in the path to all objects upstream or downstream. You can also view the trace links themselves, as independent objects.

Viewing Objects in a Defining or Complying Path

For the object selected in the hierarchical content table, defining and complying objects are displayed in the **Trace** subtab of the **Links** tab and window. Each pane shows linked objects in the order of their precedence in the defining or complying path. In the **Links** window, the subtab can show such paths for the object selected in the **Where Used** or **Versions** tab. For more information, see [Hierarchical Content Table](#), [Trace Subtab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

In addition, the **Defining Object Traceability** window displays defining objects, and the **Complying Object Traceability** window displays complying objects, for the object selected in the hierarchical content table. For a selected folder, each window shows all objects in the folder, whether or not the folder defines those objects. The objects are shown in the order of their precedence in the defining or complying path.

To view defining and complying objects in the Links tab or window:

Do one of the following:

- For an object in the hierarchical content table, select the object and click the **Links** tab, and then click the **Trace** subtab. You can open the **Links** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
- For an object in the **Where Used** or **Versions** tab:
 - . With the **Links** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Links** window, and then click the **Trace** subtab.
 - . Select the object in the **Where Used** or **Versions** tab.

The **Defining Trace** column displays the objects that define the selected object, its direct predecessors in the defining path. A plus sign (+) is shown for each defining object that complies with other objects, continuing the path upstream.

The **Complying Trace** column displays the objects that comply with the selected object, its direct successors in the complying path. A plus sign (+) is shown for each complying object that defines other objects, continuing the path downstream.

You can view the objects in a continuing path by doing the following:

- To view the direct predecessors or successors, click the plus sign for the object.
- To view all predecessors or successors simultaneously, right-click the object and choose **Expand All** from the pop-up menu. In the **Links** tab, you can also select the object, pull down the **View** menu, and choose **Expand All**.

To view defining objects in the Defining Object Traceability window:

Select the object in the hierarchical content table, and then pull down the **Tools** menu and choose the **Traceability**→**Defining** options.



You can also right-click the object and choose the **Traceability**→**Defining** options from the pop-up menu.

The **Defining Object Traceability** window displays the objects that define the selected object, its direct predecessors in the defining path. In the **Name** column, a plus sign (+) is shown for each defining object that complies with other objects, continuing the path upstream.

You can view the objects in a continuing path by doing the following:

- To view the direct predecessors, click the plus sign for the object.
- To view all predecessors simultaneously, right-click the object and choose **Expand All** from the pop-up menu.

The property columns provide additional information. You can add and remove columns, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), in chapter 9, *Working With Object Properties*.



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can move, resize, minimize, and maximize the window through the standard Microsoft Windows functions. By right-clicking objects to display a pop-up menu, you can also do the following:

- Export objects to Microsoft Excel or Microsoft Word. For more information, see [Exporting Objects to Microsoft Office Excel](#) and [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.
- Open linked requirements in Microsoft Word to edit or view their content. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Navigate to linked objects. For more information, see [Navigating to a Linked Object](#), later in this chapter.
- Open a live Excel file from a previous session. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

To view complying objects in the Complying Object Traceability window:

Select the object in the hierarchical content table, and then pull down the **Tools** menu and choose the **Traceability**→**Complying** options.



You can also right-click the object and choose the **Traceability**→**Complying** options from the pop-up menu.

The **Complying Object Traceability** window displays the objects that comply with the selected object, its direct successors in the complying path. In the **Name** column, a plus sign (+) is shown for each complying object that defines other objects, continuing the path downstream.

You can view the objects in a continuing path by doing the following:

- To view the direct successors, click the plus sign for the object.
- To view all successors simultaneously, right-click the object and choose **Expand All** from the pop-up menu.

The property columns provide additional information. You can add and remove columns, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You can move, resize, minimize, and maximize the window through the standard Microsoft Windows functions. By right-clicking objects to display a pop-up menu, you can also do the following:

- Export objects to Microsoft Excel or Microsoft Word. For more information, see [Exporting Objects to Microsoft Office Excel](#) and [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.
- Open linked requirements in Microsoft Word to edit or view their content. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) and [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*.
- Navigate to linked objects. For more information, see [Navigating to a Linked Object](#), later in this chapter.
- Open a live Excel file from a previous session. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

Viewing Trace Links for an Object

Trace links are displayed in the **Relations** subtab of the **Links** tab and window. For the object selected in the hierarchical content table, the subtab shows defining trace links, complying trace links, and the starting or ending object for each trace link.

In the **Links** window, the **Relations** subtab can show such information for the object selected in the **Where Used** or **Versions** tab. For more information, see [Hierarchical Content Table](#), [Relations Subtab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To view trace links for an object:

Do one of the following:

- For an object in the hierarchical content table, select the object and click the **Links** tab, and then click the **Relations** subtab.

You can open the **Links** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
- For an object in the **Where Used** or **Versions** tab:
 - With the **Links** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Links** window, and then click the **Relations** subtab.
 - Select the object in the **Where Used** or **Versions** tab.

In the **Start** pane, defining trace links to the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the object from which each defining trace link originates.

In the **End** pane, complying trace links from the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the object at which each complying trace link is completed.



In each pane, the rows in the object table and the link table remain synchronized at all times. For example, when you sort or scroll a link table, the corresponding object table is automatically sorted or scrolled to match the new view in the link table.

Navigating to a Linked Object

From any of the following trace link views, you can navigate to a linked object in its native location:

- The **Links** tab and floating window display defining objects, complying objects, and trace links for the object selected in the hierarchical content table. The **Links** window displays linked objects and trace links for the object selected in the **Where Used** or **Versions** tab.
- The **Defining Object Traceability** window displays defining objects, and the **Complying Object Traceability** window displays complying objects, for the object selected in the hierarchical content table. For a selected folder, each window also displays all objects in the folder, whether or not the folder complies with or defines those objects.

For more information, see [Viewing Object Relationships](#), earlier in this chapter; and [Hierarchical Content Table](#), [Links Tab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

The linked object's native location may be within the same Architect/Requirements database. Or, the linked object may reside in one of the following Teamcenter products:

- Engineering Process Management (version 8.1.1 or later)
- Teamcenter Enterprise (version 3.1 or later)
- Project (version 3.2 or later)

To navigate to a linked object:

Do one of the following:

- In the **Links** tab or window, select the object, and then pull down the **View** menu and choose the **Go To**→**Go To Object** options. Or, right-click the object and choose the **Go To Object** option from the pop-up menu.
- In the **Defining Object Traceability** window or the **Complying Object Traceability** window, right-click the object and choose the **Go To Object** option from the pop-up menu.

If the object is within the same database, the hierarchical content table displays and highlights that object.

If the object is in another Teamcenter product, that product displays and highlights the object. If necessary, enter your user name and password for the product.



The **Go To Object** option is also available on the **View** menu and a pop-up menu in the Search Results dialog window. For more information, see [Basic Search View](#) or [Intermediate Search View](#) in chapter 11, *Using the Search Module*.

Deleting Trace Links for an Object

You can delete trace links for the object selected in the hierarchical content table or the **Where Used** or **Versions** tab or window. The trace links are displayed in the **Relations** subtab of the **Links** tab and window.

Deleted trace links are moved to your Architect/Requirements Recycle Bin and can be restored until your Recycle Bin is emptied. For more information, see [Restoring Objects](#) or [Emptying Your Architect/Requirements Recycle Bin](#) in chapter 4, *Maintaining a Project*.

To delete trace links for an object:

1. Do one of the following:
 - For an object in the content table, select the object and click the **Links** tab, and then click the **Relations** subtab.
You can open the **Links** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
 - For an object in the **Where Used** or **Versions** tab:
 - With the **Links** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Links** window, and then click the **Relations** subtab.
 - Select the object in the **Where Used** or **Versions** tab.

In the **Start** pane, defining trace links to the object are displayed in the link table, at the right of the pane. At the left, the object table displays the object from which each defining trace link originates.

In the **End** pane, complying trace links from the object are displayed in the link table, at the right of the pane. At the left, the object table displays the object at which each complying trace link is completed.

2. In either link table, select the trace links that you want to delete.
You can select one trace link or a group of nonadjacent or adjoining trace links. You cannot select from both link tables at the same time.
3. Pull down the **Edit** menu and choose **Delete**.



Delete is available also when objects are selected in the **Trace** subtab. However, trace links are identified explicitly only in the link tables of the **Relations** subtab. Siemens PLM Software recommends that you delete only from the link tables unless you are certain of the trace links between objects in the other subtab.



Delete is disabled if the selection is in either object table.

A message asks if you are sure you want to delete the selected objects.

4. Click **Yes** to move the selected trace links to your Recycle Bin.

Procedure Notes

Step 3: You can also right-click the selection and choose **Delete** from the pop-up menu. You can reverse this action by pulling down the **Edit** menu and choosing **Undo**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Generic Links

Generic link is a new database object type that can be sub-typed to support the application-specific linking needs. Specifically, the generic link supports the **DiagramElement** link for the Sparx interface.

Generic links and the linked objects appear on the **Relations** subtab of the **Links** tab. There is no visual indication of whether an object participates in the generic links.



- There are no user interface commands for creating generic links. The generic links can be created only through the API methods. However, once created, you can view and interact with generic links as described in this section.
- The standard object-level access mechanisms of the Architect/Requirements apply to the generic links.
- The generic link type definition is created while creating or upgrading the database. Application developers are responsible for the installation of the application-specific subtypes.
- The **Create Version** command creates a copy of a generic link to the new version. This means when an object with generic links is versioned, copies of the generic links exist to both the new and the old versions. For more information, see [Creating Versions](#) in chapter 10, *Working With Versions*.

Customizing Generic Links

The properties of a generic link can be viewed and modified in the same way as a trace link. The default behavior of a generic link is similar to that of a trace link for database operations such as import, export, and delete.

The behavior of a generic link can be customized using:

- Property settings on a generic link type (subtype) definition.
- Standard API methods, such as, **createLinks**, **deleteLinks**, **setValue**, **getValue**, and **getList**. For more information, see the *Systems Architect/Requirements Management API Reference* manual.
- The activators. For more information about the activators, see the *Systems Architect/Requirements Management API Reference* manual.

Chapter 8: Recording Supplementary Information With Notes

This chapter contains an overview of notes and instructions for attaching notes to objects and for editing and viewing notes.

Overview of Notes

Whereas requirements define the product that the customer wants, notes record supplementary information about the product development project. For example, notes can record phone conversations with the customer, minutes of project meetings, and new regulations that affect the project.

Depending on the importance of its information, a note may be either permanent or temporary. A note communicating a management decision could exist throughout the project's duration. In contrast, a note presenting a comment or a question to other team members may be deleted after it serves its purpose.

For informal purposes, or for recording information that applies only to certain objects, notes provide greater flexibility than do object properties. Each object type has a predefined set of properties, which apply to all objects of that type, and every object of a given type will always have those same properties. However, notes can be attached to a particular object at any time, in any number, and for any reason.

Although notes serve a relatively unstructured purpose, some formal usage conventions can be helpful. For example, note names based on purpose and content provide greater clarity for team members and for other users. Consistent naming conventions make notes easier to find, and make their interest to a user more apparent without the need for opening and reading the content. Names like **Decision Rationale**, **Design Note**, and **Testing Comments** are common in Architect/Requirements.

For entering note content, Architect/Requirements provides the following:

- A plain text editor.
- A rich text editor.
- An interface with Microsoft Office Word.

Attaching a Note to an Object

You can attach a note to an object selected in any of the following views:

- The navigation tree.
- The hierarchical content table.
- The **Attachments**, **Links**, and **Connectivity** tabs and windows.



In the **Attachments** tab and window, you can create a note on another note or on any other object. Notes are displayed hierarchically, and the new note becomes a child at the next lower level.

When you create a note, you choose the format in which you want to enter the initial content. After the note's creation, its **Text Format** property value shows the content format. You can choose one of the following formats:

- Plain text, shown by the **Text** value.
- Rich text, shown by the **HTML** value.
- Microsoft Office Word, shown by the **MHTML** value.

You also assign a subtype when you create the note. The subtype can be the default subtype, **Note**, or a user-defined subtype created by your project administrator. You can change the **Text Format** and **Subtype** properties after the note's creation. For more information, see [Editing the Properties of a Selected Object](#) in chapter 9, *Working With Object Properties*.

Creating a Plain Text Note

1.

Select the owning object for the note, pull down the **File** menu, and choose **New→Note Format→Plain Text Note**.

The **Create Note On** dialog window is displayed.



You can also display this dialog window by clicking the **Create New Note** button on the toolbar. However, this button invokes the editor for the last note you created, which may or may not be plain text.

2. In the **Name** field, enter a name that reflects the note's purpose.

3. In the **Type** field, select the subtype for the note.



When your project administrator adds or deletes note subtypes, the changes are not automatically reflected in the **Type** field. You can click **Refresh** to ensure that this field shows the current list of note subtypes.

4. Do one of the following:

- Enter initial plain text content in the text pane, and then click **OK**.



Instead of clicking **OK**, you can click **Use Rich Text** to transfer the plain text to the rich text editor. To transfer the plain text to Microsoft Office Word, you can click **Open in Word** instead of **OK**. For more information, see [Creating a Rich Text Note](#) and [Creating a Note in Microsoft Office Word](#), later in this chapter.



You cannot go back to rich text from Word.

- To create an empty note without initial content, click **OK**.

The **Attachments** tab or window displays the new note.



If the owning object is in that tab or window, you may need to click the owner's plus sign (+) to see the new note.

Procedure Notes

Step 1: You can also right-click the owning object and choose **New→Note Format→Plain Text Note** from the pop-up menu. Or, press control-N.

Step 4: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating a Rich Text Note

1.

Select the owning object for the note, pull down the **File** menu, and choose **New→Note Format→Rich Text Note**.

The Create Note On dialog window is displayed.



If the **Create Note On** dialog window is already displayed in the plain text mode, click **Use Rich Text** at the bottom of the dialog window.



You can also display this dialog window by clicking the **Create New Note** button on the toolbar. However, this button invokes the editor for the last note you created, which may or may not be rich text.

2. In the **Name** field, enter a name that reflects the note's purpose.

3. In the **Type** field, select the subtype for the note.



When your project administrator adds or deletes note subtypes, the changes are not automatically reflected in the **Type** field. You can click **Refresh** to ensure that this field shows the current list of note subtypes.

4. Do one of the following:

- Enter initial rich text content in the text pane, and then click **OK**.



Instead of clicking **OK**, you can click **Open in Word** to transfer the rich text to Microsoft Office Word. For more information, see [Creating a Note in Microsoft Office Word](#), later in this chapter.



You cannot go back to rich text from Word.

Use the buttons above the pane to apply rich text formatting. You can rest the pointer on each button to see its description in a tooltip.



You can create hyperlinks, but you cannot follow them from the rich text editor. To navigate to the target object of a hyperlink in the text pane, display the note content in the **Preview** tab or open the note in Word. Then, follow the hyperlink from that content.

- To create an empty note without initial content, click **OK**.

The **Attachments** tab or window displays the new note.



If the owning object is in that tab or window, you may need to click the owner's plus sign (+) to see the new note.

Procedure Notes

Step 1: You can also right-click the owning object and choose **New**→**Note Format**→**Rich Text Note** from the pop-up menu.

Step 4: You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Note**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating a Note in Microsoft Office Word

Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

1.

Select the owning object for the note, pull down the **File** menu, and choose **New→Note Subtype→MS Word Note**.

You can also right-click the object and choose the options from the pop-up menu.

The Create Note On dialog window is displayed.



You can also display this dialog window by clicking the **Create New Note** button on the toolbar. However, this button invokes the editor for the last note you created, which may or may not be Microsoft Office Word.

2. In the **Name** field, enter a name that reflects the note's purpose.

3. In the **Type** field, select the subtype for the note.



When your project administrator adds or deletes note subtypes, the changes are not automatically reflected in the **Type** field. You can click **Refresh** to ensure that this field shows the current list of note subtypes.

4. Do one of the following:

- To enter initial content:

.

Click **Open in Word**.

The note is created in the **Attachments** tab or window while an **.mhtml** file opens in Word. You can reverse this action by immediately closing the file, and then pulling down the **Edit** menu and choosing **Undo New Note**, clicking the **Undo** button on the toolbar, or pressing control-Z.

.

Enter the content and save the Word file to commit the content to the database. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) in chapter 5, *Managing Requirements*.

- To create an empty note without initial content, click **OK**.

The **Attachments** tab or window displays the new note. You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Note**, clicking the **Undo** button on the toolbar, or pressing control-Z.



If the owning object is in that tab or window, you may need to click the owner's plus sign (+) to see the new note.



You can also create an **MHTML** note through the Select Subtype dialog window. To display this dialog window, pull down the **File** menu or right-click the selected object, and then choose **New→Subtype**. Or, click the **Create a new subtype** button on the toolbar or press control-U.

When you select a note subtype and click **OK**, the **Attachments** tab or window displays the note with a default name in an open text field. Enter the name and press the enter key. Then, you can open the note in Word to enter content.

To reverse this action, you can pull down the **Edit** menu and choose **Undo Create Subtype**, click the **Undo** button on the toolbar, or press control-Z.

Editing a Note

Each note has a **Text Format** property, whose value shows the note's content format. Architect/Requirements provides a content editor for each of the following formats:

- Plain text, shown by the **Text** value.
- Rich text, shown by the **HTML** value.
- Microsoft Office Word, shown by the **MHTML** value.

You can add the **Text Format** column to the **Attachments** tab and window. Or, you can view this property for a selected note by pulling down the **File** menu and choosing **Properties** to display the Edit Properties dialog window. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.



To edit an **MHTML** note, Microsoft Office Word 2003 or 2007 must be installed on your computer.

To edit a note:

1.

For the object to which the note is attached, display the notes by doing one of the following:

- For an object in the navigation tree, the hierarchical content table, or the Recycle Bin, select the object and click the **Attachments** tab.

You can open the **Attachments** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.



Deleted notes in the Recycle Bin cannot be edited. However, notes attached to other deleted objects can be displayed in the **Attachments** tab or window, and can then be edited.

- For an object in the **Attachments** tab or window, including a top level note, click the plus sign (+) to display the notes at the next lower level.
- For an object in the **Links, Connectivity, Where Used**, or **Versions** tab:
 - . With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
 - . In the **Links, Connectivity, Where Used**, or **Versions** tab, select the object to display its notes in the **Attachments** window.

2. Select the note, and then pull down the **File** menu and choose **Open**.



If necessary, click the plus signs for child notes to display the note that you want to select.

For a plain text or rich text note, the Edit dialog window is displayed.

For a Microsoft Word note, Word opens an **.mhtml** file.

3. Do one of the following:

- In the Edit dialog window:

- . Enter content changes in the text pane.



In the rich text editor, use the buttons above the pane to apply rich text formatting. You can rest the pointer on each button to see its description in a tooltip.

You can also do the following:

- Change the name in the **Name** field.
- Change the subtype in the **Type** field.

When your project administrator adds or deletes note subtypes, the changes are not automatically reflected in the **Type** field. You can click **Refresh** to ensure that this field shows the current list of note subtypes.

- . Click **OK** to close the dialog window and save the changes in the database.



- Instead of clicking **OK**, you can click **Use Rich Text** to transfer plain text changes to the rich text editor.
- To switch from plain text or rich text to Microsoft Word, you can click **Open in Word** instead of **OK**.

-

In Microsoft Word:

- . Enter content changes as you do in a typical Word file.

For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) in chapter 5, *Managing Requirements*.

- . To save the changes in the database, choose **Save** from Word's **File** menu.

Procedure Notes

Step 1: You can also right-click the note and choose **Open** from the pop-up menu. Or, double-click the note.

Enabling Text Property Editing

When editing is disabled in a **Text** property cell, the note has a **Text Format** property value of **MHTML**. This value indicates content saved in Microsoft Office Word.

Disabled cell editing is indicated also by certain visual cues in the **Text** property cell. Depending on the client view where the property is displayed, the cues are:

- A shaded background and blue text for disabled cells in the **Attachments** tab and window, the **Links** tab and window, and the Search Results dialog window.
- Dimmed text for disabled cells in the **Properties** floating window and the Edit Properties dialog window.



Although you can edit disabled **Text** property cells in a live Excel file, edits are not saved in the Architect/Requirements database. An error message is displayed when you click outside the cell. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

For a note with such a cue, you can enable direct editing in the **Text** property cell by changing the **Text Format** property value to **Text**. For more information, see [Editing Text Property Cells](#), earlier in this chapter.



The next time the cell is edited, current headings, body text, lists, and table text are converted to plain text, and all text is concatenated into one paragraph. All formatting is removed, including graphics, table grids, and other content elements that may be saved in Word. For more information, see [Editing a Note](#), earlier in this chapter.

To enable Text property editing:

1. Select the note, and then do one of the following:
 - To use the **Properties** window, click the **Properties** tab, and then click the **Open tab** button on the notebook pane toolbar.
 - To use the **Edit Properties** dialog window, pull down the **File** menu and choose **Properties**. Or, right-click the note and choose **Properties** from the pop-up menu.

2.

In the **Values** column, double-click the **Text Format** property value.

The Single-Choice dialog window is displayed.

3. Select the **Text** option button, and then click **OK**.



This action changes the text format to plain text and removes all other content elements, effective the next time the cell is edited.

A warning message states that saving as plain text causes all formatting, pictures, and objects in the text to be lost on the next text edit.

4. To continue, click **OK** to close the Single-Choice dialog window.
5. Click **Close** to apply the change and close the Edit Properties dialog window.

Editing Text Property Cells

Plain text content can be edited directly in the cells of the **Text** property column. You can add this column in the **Attachments** tab and window, where notes are displayed for the object selected in the navigation tree or the content table. Also, notes can be output to the Search Results dialog window, where the **Text** property can be added. The **Text** property is displayed by default in the **Properties** floating window and the Edit Properties dialog window. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.

You can use any of these client views to edit the **Text** property for one note. For two or more notes, you can use only the Edit Properties dialog window.

To edit a Text property cell for one note:

In the **Attachments** tab or window, the **Links** tab or window, or the Search Results dialog window:

1. With the **Text** property added, double-click the cell that you want to edit.



A shaded background and blue text indicate that cell editing is disabled. The note's **Text Format** property value is **MHTML**, for content saved in Microsoft Office Word. Cell editing can be enabled by changing the **Text Format** value to **Text**. For more information, see [Enabling Text Property Editing](#), later in this chapter.

A text field opens in the cell. Scroll bars are provided to the right if the text length exceeds the width of the **Text** column.

2. Enter your changes, and then press the enter key or click outside the field.

In the **Properties** floating window or the Edit Properties dialog window:



To see more text in the **Text** column, change the width before you begin. Scroll bars are not provided in the cell.

1. Double-click the **Text** property value to open a text field in the cell.



Dimmed text indicates that cell editing is disabled. The note's **Text Format** property value is **MHTML**, for content saved in Microsoft Office Word. Cell editing can be enabled by changing the **Text Format** value to **Text**. For more information, see [Enabling Text Property Editing](#), later in this chapter.

2. Enter your changes, and then press the enter key or click outside the field.

To display the **Properties** window, select the object to which the note is attached, click the **Properties** tab, and then click the **Open tab** button on the notebook pane toolbar.

To display the Edit Properties dialog window, select the note, pull down the **File** menu, and choose **Properties**.

To edit Text property cells for two or more notes:



The **Text** property cannot be displayed if any one selected note has a **Text Format** property value of **MHTML**, for content saved in Microsoft Office Word. Before you start this procedure, ensure that the **Text Format** value is **Text** for each note that you intend to edit. You can set this value if necessary. For more information, see [Enabling Text Property Editing](#), later in this chapter.

1. Select the notes in the **Attachments** tab or window, the **Links** tab or window, or the Search Results dialog window.

You can select nonadjacent and adjoining notes of the same subtype or of mixed subtypes.

2. Pull down the **File** menu and choose **Properties**.

The Edit Properties dialog window is displayed.

3. In the **Value** column, double-click the **Text** property cell.

A text field opens in the cell.

4. Enter your changes, and then press the enter key or click outside the field.
5. Click **OK** to close the dialog window.

Procedure Notes

Step 2: You can also right-click the selection and choose **Properties** from the pop-up menu.

Viewing Note Content

The **Attachments** tab and floating window display the notes for the object selected in the navigation tree, the hierarchical content table, or your Recycle Bin. For the object selected in the **Links, Connectivity, Where Used, or Versions** tab, notes are displayed in the **Attachments** window.

Each note has a **Text Format** property, whose value shows the note's content format. A read-only mode is provided in the content editor for each of the following formats:

- Plain text, shown by the **Text** value.
- Rich text, shown by the **HTML** value.
- Microsoft Word, shown by the **MHTML** value.

Also, the **Preview** window can display read-only content in all formats. For more information, see [Preview Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

You can add the **Text Format** column to the **Attachments** tab and window. Or, you can view this property for a selected note by pulling down the **File** menu and choosing **Properties** to display the Edit Properties dialog window. For more information, see [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.



To view an **MHTML** note in Word, Microsoft Word 2013 or Microsoft Word 2016 must be installed on your computer.

To view note content:

1. For the object to which the note is attached, display the notes by doing one of the following:
 - For an object in the navigation tree, the hierarchical content table, or the Recycle Bin, select the object and click the **Attachments** tab.

You can open the **Attachments** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.



Deleted notes in the Recycle Bin cannot be opened for viewing. However, notes attached to other deleted objects can be displayed in the **Attachments** tab or window, and can then be opened.

- For an object in the **Attachments** tab or window, including a top level note, click the plus sign (+) to display the notes at the next lower level.
- For an object in the **Links, Connectivity, Where Used, or Versions** tab:
 - With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
 - In the **Links, Connectivity, Where Used, or Versions** tab, select the object to display its notes in the **Attachments** window.



In the **Attachments** tab or window, you can rest the pointer on the note's **Text** property cell to see the plain text content in a tooltip.

2. Select the note, and then do one of the following:



If necessary, click the plus signs for child notes to display the note that you want to select.

- Pull down the **File** menu and choose **Open Read-Only**.

For a plain text or rich text note, the Edit dialog window displays the content in read-only mode.

For a Microsoft Word note, the content is displayed in a read-only **.mhtml** file. You can also print the content and send it by E-mail and fax.

- Use the **Preview** floating window by doing one of the following:

- . For an object in the navigation tree, the hierarchical content table, or the Recycle Bin:

- Select the object, and then click the **Attachments** tab to display the object's notes.

You can open the **Attachments** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.

- With the **Preview** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Preview** window.
- In the **Attachments** tab or window, select the note to display its content in the **Preview** window.

- . For an object in the **Links, Connectivity, Where Used, or Versions** tab:

- With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
- In the **Links, Connectivity, Where Used, or Versions** tab, select the object to display its notes in the **Attachments** window.
- With the **Preview** tab on top, click the **Open tab** button to open the **Preview** window.
- In the **Attachments** window, select the note to display its content in the **Preview** window.

To print the content, you can right-click in the content area and choose **Print** from the pop-up menu.

Procedure Notes

Step 2: You can also right-click the note and choose **Open Read-Only** from the pop-up menu.

Chapter 9: Working With Object Properties

This chapter provides an overview of property types and values and contains instructions for editing properties in the Architect/Requirements client and with live Excel, the Architect/Requirements interface with Microsoft Excel.

Overview of Object Properties

Properties are named values that determine the nature and behavior of all objects of a given object type. Every property belongs to one of two property types, either *system-defined* or *user-defined*.

System-Defined Properties

Architect/Requirements assigns system-defined properties to each built-in object type. The system-defined properties of an object type are inherited by all subtypes based on that object type.

Some system-defined properties record general information, such as an object's creation time, that applies to all object types. Other system-defined properties have a more specific purpose that applies to only one or two object types, for example, the paragraph number of a requirement, or the text of a requirement or a note.

Each system-defined property falls into one of two categories:

- A *read-only* property has a value that can be changed only by the system itself. The user cannot change such a value, except as an indirect result of certain other actions on an object. For example, an object's change time is a read-only property.
- An *editable* property has a default value that can be changed directly by the user. For example, an object's name is an editable property.

The **Properties** tab and floating window display all viewable system-defined properties for the object selected in the content table. Other views in the Systems Engineering and Requirements Management module display certain system-defined properties by default. In these views, default properties can be removed and other system-defined properties can be added. For more information, see [Properties Tab](#) and [Using Tabs in Floating Windows](#) in chapter *Using the Architect/Requirements Main Window*; [Adding and Removing Columns](#), later in this chapter; and appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).

User-Defined Properties

User-defined properties are created locally by an Architect/Requirements project administrator. By assigning these properties to type definitions in the Administration module, the project administrator can extend the built-in object types for specific purposes. Subject to user permissions, all user-defined properties are editable.

Property Values

Whether system-defined or user-defined, each property value is one of the following:

-

Choice

The value is selected from a predefined list of choices. A choice list can be defined as either of the following:

- o *Single-choice*

From a single-choice list, you can select only one choice for the value.

- o *Multiple-choice*

From a multiple-choice list, you can select any combination of choices, and that combination becomes the value.

-

Date

The value is a calendar date, and can be defined to include the time of day. User inputs are checked, and only valid calendar dates are accepted. Sorting is by the date's chronological value, regardless of its alphanumeric representation.

-

Numeric

The value is a number, and can be defined to allow a variety of floating-point formats. User inputs are checked, and only valid numeric values are accepted. Sorting is by the number's binary value, regardless of its string representation.

-

Text

The value is any string of plain text, including a blank string. Sorting is alphanumeric.

Releasing Reservation of Objects

When you open an object, the Architect/Requirements places a reservation or lock on that object that prevents other users from modifying it simultaneously. You can release this reservation by choosing the **Tools**→**Release Reservation** command. You can only release your own reservations, however, administrators can also release reservations owned by other users.

The **Release Reservation** command should not be used when you are editing the contents of an object. However, it is helpful to use this command if there was a system problem that caused the reservation to be held even after closing the object after modifications.

One of the common uses of the **Release Reservation** command is when the activators are edited in multi-pane editors. You must use this command when other users want to edit the same activator. This command is not needed if you want to edit the activator again during the same session.

Customizing Views of Property Columns

Custom column settings let you easily see and work with a set of properties. For example, in a given folder, you can group objects by a common purpose, and apply a view with property columns related to the purpose.

Custom views are available in:

- The hierarchical content table.
- The **Attachments** tab or window.
- The **Trace** subtab of the **Links** tab or window.
- The **Connectivity** tab or window.
- The **Versions** tab or window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.
- The Search Results dialog window.



Custom views are available also through the **Saved Views** filtering option in the **Properties** tab and window. For more information, see [Filtering Property Tables](#), later in this chapter.

Views are project specific, residing in the **Reports and Formatting** folder in the Administration module. If you have questions about custom property views, consult your project administrator.



- Views are available also in the Basic pane of the Search module. You can specify the property columns for each search by selecting a view in the field at the bottom of the **Output Options** section. The columns in the selected view are output to the Search Results dialog window.
- Views are not enabled when the root node or your Architect/Requirements Recycle Bin is selected in the navigation tree.
- When a child folder is created in the content table of the same type as the parent, the child inherits the default view named in the parent's **Default View** property value if that value has been set. If the child folder is of a different subtype, it does not inherit the parent's **Default View** property value. However, if the parent's view is modified locally, the child inherits the column settings in the database, not the local modifications. For more information, see [Modifying a View Locally](#), later in this chapter.

If no default view is set for the parent, the child's view is determined by the conditions described in table [Conditions Determining View in Content Table](#), later in this chapter.

- When a folder's subtype is changed, the folder's **Default View** property value does not change.

If the folder is a parent of other folders, the subtype change does not extend to the children.

Conditions That Determine the Column View in the Content Table

Each folder selected in the navigation tree can display a unique set of columns in the content table. Architect/Requirements evaluates certain conditions to determine the default view for the content table. Table 9-1 describes these conditions and views, from the first condition evaluated to the last.

Table 9-1. Conditions Determining View in Content Table

Condition	View
Local modifications to column settings are recorded on your computer.	The columns that you set only by adding, removing, rearranging, resizing, or sorting columns. These settings are not saved in the database. For more information, see Modifying a View Locally , later in this chapter.
Local modifications are not recorded or are overridden by the <Default View> option in the View field.	The view named in the Default View property value for the folder. For more information, see Setting the Default View for a Folder , later in this chapter.
No default view is set for the selected folder.	The default view for the folder type definition in the Administration module. If you have questions about type definitions, consult your project administrator.  For the selected folder, the Default View property value is blank when the type definition's default view is used.
No default view is set for the folder type definition.	The default view for your user name. For more information, see Setting the Default View for Your User Name , later in this chapter.
No default view is set for your user name.	The system default view. In this view, the property columns are Attachment Count , Trace Link Count , Number , ROIN , Create Time , Create User , and Type Name .  The View list displays *System Settings* when the system default view is used.

Applying a View in the Content Table

For each folder that you select in the navigation tree, you can apply a unique view of property columns to display in the content table. Using the **View** list, you can apply *public* views, which are visible to all users in the project. Also, you can apply non-public views that you create and save in the database. For more information, see [Creating a Column View](#), later in this chapter.

The **View** list is located to the right of the **Address** bar above the content table. Your selection is recorded only on your computer, not in the database, and persists each time you select the folder.

To apply a view in the content table:

- Do either of the following:
 - Click the **View** list and select the name of the view.
 - Right-click inside the content table, choose **Select View** from the pop-up menu, and choose the name of the view.

The content table displays the columns that are saved in the database for the view. The **View** list shows the view name, and also your user name if you created the view and it is not yet public.

The view persists when you refresh the data by pulling down the **View** menu and choosing **Refresh**, or by clicking the **Refresh** button on the toolbar.



- The **<Default View>** option overrides any local modifications by applying the default view for the folder. If you select **<Default View>** in the **View** list, the list displays a blank value instead of the actual name of the default view. The default view is determined by the conditions described in table [Conditions Determining View in Content Table](#), earlier in this chapter. For more information, see [Modifying a View Locally](#), later in this chapter.

- The ***System Settings*** option applies the Architect/Requirements default columns. The default columns are the following:

- **Attachment Count**
- **Trace Link Count**
- **Number**
- **ROIN**
- **Create Time**
- **Create User**
- **Type Name**

Applying a View in the Notebook Pane

You can apply a custom view in the **Attachments**, **Connectivity**, and **Versions** tabs and windows. You can also apply a view in the **Trace** subtab of the **Links** tab and window, and in the **Defining Object Traceability** and **Complying Object Traceability** windows.

To apply a view in the notebook pane:

- Right-click inside the tab or window, choose **Select View** from the pop-up menu, and choose the name of the view from the submenu.



You can apply the Architect/Requirements default columns for the tab or window by choosing ***System Settings***.

The tab or window displays the columns that are saved in the database for the view. The view name is checked on the pop-up submenu.

The view persists when you refresh the data by pulling down the **View** menu and choosing **Refresh**, or by clicking the **Refresh** button on the toolbar.



In the **Properties** tab and window, you can use the **Saved Views** filtering option to apply a custom view. For more information, see [Filtering Property Tables](#), later in this chapter.

Creating a Column View

Using the current column settings, you can create a new view in the database. You can then set unique columns for the new view.

You can create a custom view in:

- The hierarchical content table.
- The **Attachments**, **Connectivity**, and **Versions** tabs or windows, and the **Trace** subtab of the **Links** tab or window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.
- The Search Results dialog window.



If you change column settings before step 1, you modify the current view on your computer. The changes are recorded locally, not in the database, and they persist until you reapply that view. For more information, see [Modifying a View Locally](#) and [Applying a View in the Content Table](#), later in this chapter.

To create a column view:

1.

Pull down the **View** menu and choose **Save As** to display the Save View dialog window.

You can also right-click inside the content table or the tab or window, and then choose **Save As** from the pop-up menu.

2. Enter a unique name in the **Enter new view name** field, and then click **OK**.



If you enter a view name that already exists in the project, an error message is displayed and you must repeat steps 1 and 2.

In the content table, the **View** field displays the view name and your user name. In the tab or window, the view name is checked on the **Select View** pop-up submenu.

To customize the view, you can choose unique column settings, and then pull down the **View** menu and choose **Save**. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), later in this chapter.



As the creator, you can mark the view to be made visible to all users in the project. First, switch to the Administration module and select the view in the **Reports and Formatting** folder. Next, in the **Properties** tab or window, double-click the **Shared State** property value to display the Single-Choice dialog window. Then, check the **Pending** check box and click **OK**. A project administrator can set the value to **Public**.

Setting the Default View for a Folder

For the folder selected in the navigation tree, the default view in the content table is determined by the conditions described in table [Conditions Determining View in Content Table](#), earlier in this chapter.



You cannot set the default view in the notebook pane.

Architect/Requirements references the default view for the folder when one of the following conditions exists:

- Column settings are not changed locally. For more information, see [Modifying a View Locally](#), later in this chapter.
- Local modifications are overridden by the **<Default View>** option in the **View** field. For more information, see [Applying a View in the Content Table](#), earlier in this chapter.

A default view may be set for the folder type definition in the Administration module.

Architect/Requirements references that default view if one is not set for the individual folder in the Systems Engineering and Requirements Management module.

By setting the default view for the folder, you override the type definition's default view with one that you specify. Also, you can change the folder's default view if one already exists.



This procedure requires one of the following:

- You must have **Project Administrator** privilege for the project.
- Your user name must be included in the **Full Control** property value of the security profile that is applied to the folder. If you have questions about security profiles, consult your project administrator.

To set the default view for a folder:



This procedure does not change the current view and column settings.

1. In the navigation tree or the hierarchical content table, select the folder.
The **Properties** tab displays the folder's properties.
2. In the **Value** column, double-click the **Default View** value.
3. In the Single-Choice dialog window, check the check box for the default view and click **OK**.

Procedure Notes

Step 1: Instead of the **Properties** tab, you can use the **Properties** floating window by clicking the **Open tab** button on the notebook pane toolbar. Or, pull down the **File** menu and choose **Properties** to use the Edit Properties dialog window.

Setting the Default View for Your User Name

To determine the view for a selected folder, Architect/Requirements references the default view for your user name when both of the following conditions exist:

- Column settings are not changed locally, or local modifications are overridden by the **<Default View>** option in the **View** field. For more information, see [Modifying a View Locally](#) or [Applying a View in the Content Table](#), earlier in this chapter.
- A default view is not set for the folder, or for the folder type definition in the Administration module. For more information, see [Setting the Default View for a Folder](#), earlier in this chapter. If you have questions about type definitions, consult your project administrator.

Your default view applies to each folder in the navigation tree for which all conditions exist. For more information, see table [Conditions Determining View in Content Table](#), earlier in this chapter.

The project administrator may set a default view on your user object in the Administration module. You can change that setting in the Systems Engineering and Requirements Management module, or you can set your own default view if one does not exist. If you have questions about user objects, consult your project administrator.



You cannot set the default view in the notebook pane.

To set the default view for your user name:

- With the view applied, pull down the **View** menu and choose **Set View As Default**.

The view name becomes the value of your user object's **Default View** property.

Modifying a View Locally

Local modifications to column settings override the view that is currently applied. These column settings are saved only on your computer.

The column view can be modified in the following:

- The hierarchical content table.
- The **Attachments**, **Connectivity**, and **Versions** tabs or windows, and the **Trace** subtab of the **Links** tab or window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.
- The Search Results dialog window.

You can add and remove columns, rearrange columns, resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), later in this chapter.



- In the content table, local modifications apply only to the folder selected in the navigation tree. The settings revert to those in the database if the modified view is the folder's default view and you select **<Default View>** in the **View** list. For more information, see [Applying a View in the Content Table](#), earlier in this chapter.
- You do not need special privileges or permissions to modify any view locally. Your changes do not affect the view object in the database.
- Local modifications are not retained when you upgrade to a newer version of Architect/Requirements.

Adding and Removing Columns

You can add and remove property columns in the hierarchical content table; the **Attachments**, **Links**, **Where Used**, and **Versions** tabs and windows; and the Search Results dialog window.

In each table, you can add and remove columns for all property types, for one type, or for mixed types. You can choose a different set of columns in each view, and in each project to which you have access. Architect/Requirements saves your last settings from the current session, and displays those columns when you start a new session.



- You cannot add or remove columns in the **Properties** tab and window.
- You cannot remove the **Name**, **Defining Trace**, or **Complying Trace** column, nor can you remove the object type indicator column in the **Attachments** and **Where Used** tabs and windows.

To add and remove columns:

1. Do one of the following:
 - With the content table or a fixed tab activated, pull down the **View** menu and choose **Format Columns**.
 - In a floating tab window, right-click any column heading.

Architect/Requirements displays the Column Settings dialog window, which lists the names of all property columns for the view. The list shows whether each property is system-defined, editable, and displayed by default.

2. Do one or more of the following:
 - To add all columns for all property types, click **All** in the **Select by Group** list.
 - To remove all property columns, click **None** in the **Select by Group** list.
 - To display only the default columns, click **Default** in the **Select by Group** list.
 - To add all columns for only one property type, do one of the following in the **Select by Group** list:
 - Select **System** for properties that Architect/Requirements assigns automatically to built-in object types. For more information, see appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#).
 - Select **User-Defined** for properties that your project administrator assigns to built-in object types and custom subtypes.
 - Select **Editable** for properties whose values you can change.
 - To add or remove columns of mixed property types:
 - To add an undisplayed column, do one of the following:
 - Check the check box to the left of the property name in the table.
 - Select the property name in the table, and then click **Show**.
 - Double-click the property name in the table.
 - To remove a displayed column, do one of the following:
 - Clear the check box to the left of the property name in the list.
 - Select the property name in the list, and then click **Hide**.
 - Double-click the property name in the list.
3. Click **OK**.

The Column Settings dialog window closes, and the table displays your choices.

Rearranging Columns

You can change the position of the property columns in the hierarchical content table; the **Attachments**, **Links**, **Where Used**, and **Versions** tabs and windows; and the Search Results dialog window. For more information, see [Hierarchical Content Table](#), [Attachments Tab](#), [Links Tab](#), [Where Used Tab](#), [Versions Tab](#), and [Using Tabs in Floating Windows](#), earlier in this chapter; and chapter 11, [Using the Search Module](#).

In each table, you can use the Column Settings dialog window to move multiple columns at the same time. Also, you can use the mouse to move one column at a time. You can set different column positions in each table.



- You cannot rearrange columns in the **Properties** tab and window.
- You cannot move the leftmost column in a table. The leftmost column is the **Name**, **Defining Trace**, or **Complying Trace** column in the related view, or the object type indicator column in the **Attachments** and **Where Used** tabs and windows.

To rearrange columns with the Column Settings dialog window:

1. To display the Column Settings dialog window, do one of the following:
 - With the content table or a fixed tab activated, pull down the **View** menu and choose **Format Columns**.
 - In a floating tab window, right-click any column heading.
2. Select a property name in the dialog window, and then do one of the following:
 - To move the column to the first position after the leftmost column in the table, click **Move to Top**.
 - To move the column one position to the left in the table, click **Move Up**. You can click **Move Up** until the name reaches the intended position.
 - To move the column one position to the right in the table, click **Move Down**. You can click **Move Down** until the name reaches the intended position.

For each additional column that you want to move, repeat step 2.

3. Click **OK** to close the dialog window and move the columns to their new positions.

To rearrange columns with the mouse:

- For each column that you want to move, point to the heading, hold down the left mouse button, and drop the heading in the new position.

Resizing Columns

You can change the width of every column in all tables.

To resize columns:

1. Point to the right border of a column heading until the pointer becomes a horizontal double arrow.
2. Hold down the left mouse button, move the border left or right, and then release the mouse button.

For each additional column that you want to resize, repeat steps 1 and 2.



You can resize most columns automatically by double-clicking the right border of the column heading.

Setting the Sort Column

You can sort information by any column in each of the following:

- The hierarchical content table and your Architect/Requirements Recycle Bin (a sequential table).
- The **Properties**, **Attachments**, and **Where Used** tabs and windows (sequential tables), and the **Links** and **Versions** tabs and windows (hierarchical tables).
- The Search Results dialog window, either a hierarchical or a sequential table depending on the criteria specified in the Search module.

For example, you can sort by the **Name** column to see the objects in alphabetical order of their names. Or, you can sort by the **Create Time** column to arrange the information by the dates and times the objects were created. Every column can be sorted in ascending or descending order, including columns that contain indicators.

Although the sort column applies to all displayed rows in the table, hierarchical and sequential tables differ in the way the rows are sorted:

- In a hierarchical table, the objects remain at their present levels in the hierarchy, shown in the leftmost column. The rows are sorted within each level according to the sort column.
- In a sequential table, all rows are sorted in a continuous series according to the sort column. Objects in the Recycle Bin are displayed without regard to their levels in the hierarchies from which they were deleted.



This procedure changes only the order of display in the view. It does not affect the order of the objects or the properties in the database.

To set the sort column:

Point to the heading of the intended sort column, and then do one of the following:

- For ascending order, click the heading until it displays an up arrow.
- For descending order, click the heading until it displays a down arrow.

Modifying a View in the Database

When you change the column settings of an applied view, the modifications are recorded only on your computer. You can use the local column settings to overwrite those for the named view in the database. Then, the view can be applied consistently. For more information, see [Modifying a View Locally](#), [Applying a View in the Content Table](#), and [Applying a View in the Notebook Pane](#), earlier in this chapter.

The column view can be modified in the following:

- The hierarchical content table.
- The **Attachments**, **Connectivity**, and **Versions** tabs or windows, and the **Trace** subtab of the **Links** tab or window.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.
- The Search Results dialog window.



To overwrite a public view, you must meet one of the following conditions:

- You must have **Project Administrator** privilege for the project.
- Your user name must be included in the **Full Control** property value of the security profile that is applied to the view. If you have questions about security profiles, consult your project administrator.

To modify a view in the database:

1. With the view applied, choose the new column settings.

The modifications are recorded locally on your computer.

2. Pull down the **View** menu and choose **Save**.

The modifications are recorded in the view object, which is located in the **Reports and Formatting** folder in the Administration module.

Procedure Notes

Step 1: You can add and remove columns, rearrange and resize columns, and set the sort column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), earlier in this chapter.

Filtering Property Tables

In the **Properties** tab and floating window and the Edit Properties dialog window, you can set the properties that you want to see in the property tables. By default, each of these tables displays all viewable properties for the selected object or objects. For more information, see [Properties Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

You can limit the properties to a specific subset through filtering options on the **View** and pop-up menus and through filter buttons in each table. For example, when you export objects to Microsoft Excel, you can use a filter to see only those properties in the table and in the Excel file. For more information, see [System-Defined Properties](#) and [User-Defined Properties](#), earlier in this chapter; appendix [System-Defined Properties in the Systems Engineering and Requirements Management Module](#); and [Exporting Objects to Microsoft Office Excel](#).



- The filter buttons in the **Properties** tab are hidden each time you start a new Architect/Requirements session. In the **Properties** tab window and the Edit Properties dialog window, the buttons are hidden each time you open the window.
- If you choose a filter from a menu, all tables display the same properties. To display different properties in a table, use the filter buttons in that table.

To show or hide the filter buttons:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Toggle Controls**.
- Right-click the table and choose **Filter**→**Toggle Controls** from the pop-up menu.
- Click the down or up arrow on the horizontal split bar above the table.



To show the buttons, you can also point to the split bar until the pointer becomes a double vertical arrow. Then, click the split bar to size the filter bar automatically, or drag the split bar to the size that you want.

To display only properties whose values you can change:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Editable**.
- Right-click the table and choose **Filter**→**Editable** from the pop-up menu.
- Click the **Editable** filter button.

To display only properties whose values cannot be changed:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Read-only**.
- Right-click the table and choose **Filter**→**Read-only** from the pop-up menu.
- Click the **Read Only** filter button.

To display only the custom properties for the project:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**User Defined**.
- Right-click the table and choose **Filter**→**User Defined** from the pop-up menu.
- Click the **User Defined** filter button.

To display only system-defined properties:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**System**.
- Right-click the table and choose **Filter**→**System** from the pop-up menu.
- Click the **System** filter button.

To display the properties in a saved view:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**View Specific**.
- Right-click the table and choose **Filter**→**View Specific** from the pop-up menu.
- Click the **Saved Views** filter button, and then select the view from the list to the right.

If you choose a menu option and if the filter buttons are hidden, you cannot select a specific view. The properties are displayed from the saved view previously selected as the filter.

The saved view filters are unavailable if there are no public views for the project and no private views for your user name. For more information, see [Customizing Views of Property Columns](#).

To display all viewable properties:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**All**.
- Right-click the table and choose **Filter**→**All** from the pop-up menu.
- Click the **All** filter button.

Editing the Properties of a Selected Object

For a selected object, you can change the values of editable properties in the **Properties** tab or floating window. You can also use the Edit Properties dialog window to change editable values while displaying another tab or window.

You can select the object in the hierarchical content table; the **Attachments, Links, Where Used, and Versions** tabs and floating windows; the **Complying Object Traceability** view; and the Search Results dialog window.



- You must have **Modify** permission for the object.
- To edit properties for a building block, a TRAM, or a diagram, you must have **Architect** privilege for the project.

To edit the properties of a selected object:

1. Select the object type indicator for the object, and then do one of the following:
 - To display the **Properties** tab, click the tab.
 - To display the Edit Properties dialog window, pull down the **File** menu and choose **Properties**.

Values that you cannot change are dimmed in the **Value** column.

2.

In the **Value** column, double-click the value that you want to change.

If the property has a choice value, one of the following dialog windows appears:

- In the Single-Choice dialog window, you can select only one choice.
- In the Multi-Choice dialog window, you can select any combination of choices.



The choice list may be set dynamically, and you may see different choices from time to time. If you have questions about the choice list, consult your project administrator.

A text field opens if the property has a date, numeric, or text value. For a date value, a calendar icon is displayed to the right of the text field.



- For date and numeric values, you can rest the pointer on the text field to see the valid format in a tooltip.
- For the system-defined property named **Text**, dimmed text in a cell indicates that editing is disabled because MHTML content is saved in Microsoft Word. You can enable this cell by changing the requirement's **Text Format** property value to **Text**, and then edit the **Text** property. For more information, see [Enabling Text Property Editing for a Requirement](#) in chapter 5, *Managing Requirements*.

3. Do one of the following:

- In the Single-Choice dialog window, click the button for the new value, and then click **OK** or press the enter key.

You can click **Set to Default** to select the button for the default value. If the default value is blank, no button is selected. If you have questions about default values, consult your project administrator.

- In the Multi-Choice dialog window:
 - To select the choices for the value, do one or both of the following:
 - Check the check box for each choice that you want to add, or click **Select All** to check all check boxes simultaneously.
 - Clear the check box for each choice that you want to remove, or click **Unselect All** to clear all check boxes simultaneously.

You can click **Set to Default** to check the check box for the default choice and clear any other checked choices. If the property has no default value, all check boxes are cleared. If you have questions about default values, consult your project administrator.

- Click **OK** or press the enter key.
- In the text field, enter the new value, and then press the enter key.

For a date or numeric property, the value must match the valid format shown in the tooltip.

For a text value, all keyboard characters are valid.



For a date property, you can click the calendar icon to display a calendar in a dialog window. To enter the new value from the calendar, do one of the following:

- Select the month, year, and day to enter that date.
-

Click **Today** to enter today's date automatically.

-

Click **TBD** to enter the value **TBD**, which marks the value for entry at a later date.

Click **OK** to close the calendar and apply the value.

To change an additional value, repeat steps 2 and 3.



When filtering is on and a particular filter value for a column is selected, the data are filtered according to the selected choice. Only objects matching the filter value are shown in the columns.

If another filter value is selected after the first filter is implemented, only the objects in the window are filtered with the new filter value. If the objects in the current view do not match the filter chosen, the row is not filtered and the objects could be removed from view until a new filter is used.

You cannot use more than one filter at one time. Filters can be applied as layers to further define the view. Restoring the objects to view removes all applied filters.

Procedure Notes

Step 1: You can also right-click the object and choose **Properties** from the pop-up menu. For a note or a diagram, you can display the dialog window also from the **Attachments** tab.

Step 3: To reverse this action, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Property Editing for Multiple Object Selections

In the Edit Properties dialog window, you can change the values of editable properties for two or more selected objects. You can select the objects in the following hierarchical tables:

- The content table.
- The **Attachments, Links, Where Used, or Versions** tab or floating window.
- The **Complying Object Traceability** view.
- The Search Results dialog window.

The Edit Properties dialog window displays the editable properties that are currently shown in the table. Before you open the dialog window, you may want to set the columns for the properties that you want to see in the dialog window. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), earlier in this chapter.

Conditions of Property Display for Multiple Selections

When two or more objects are selected in a hierarchical table, certain properties that may be shown are subject to the following conditions:

- A property that has different values among the selected objects, such as the **Name** property, has a blank value in the dialog window.
- A property whose value is the same for all selected objects has that value in the dialog window.
- The **Subtype** property is displayed in the dialog window only if the selected objects are of the same base type.
- The **Number** property is not displayed in the dialog window.
- The **Text** property cannot be displayed if any one selected requirement has a **Text Format** property value of **MHTML**. This value indicates that the content is saved in Microsoft Word.



Ensure that the **Text Format** value is **Text** for each requirement whose **Text** property you intend to edit. You can set this value if necessary. For more information, see [Enabling Text Property Editing for a Requirement](#), in chapter 5, *Managing Requirements*.

Editing Properties for Multiple Objects



- You must have **Modify** permission for the objects.
- To edit properties for building blocks, TRAMs, or diagrams, you must have **Architect** privilege for the project.
- Your changes are applied to the current value of each selected object.
- If a change fails for any value, no values are changed in the database.

1.

Select the objects, pull down the **File** menu, and choose **Properties**.



You can select nonadjacent and adjoining objects of the same type or of mixed types.

Architect/Requirements displays the Edit Properties dialog window. Values that you cannot change are dimmed in the **Value** column.



The Edit Properties dialog window shows only those properties that apply to all selected object types.

2. In the **Value** column, double-click the value that you want to change.

If the property has a choice value, one of the following dialog windows is displayed:



In the Single-Choice dialog window, you can select only one choice.



In the Multi-Choice dialog window, you can select any combination of choices.



Each choice list may be set dynamically, and you may see different choices from time to time. If you have questions about a choice list, consult your project administrator.

A text field opens if the property has a date, numeric, or text value. For a date value, a calendar icon is displayed to the right of the text field.



- For date and numeric values, you can rest the pointer on the text field to see the valid format in a tooltip.
- For the system-defined property named **Text**, dimmed text in a cell indicates that editing is disabled because MHTML content is saved in Microsoft Word. You can enable this cell by changing the requirement's **Text Format** property value to **Text**, and then edit the **Text** property. For more information, see [Enabling Text Property Editing for a Requirement](#) in chapter 5, *Managing Requirements*.

3. Do one of the following:

- In the Single-Choice dialog window, select the new value in the value pane, and then click **OK** or press the enter key.

You can click **Set to Default** to select the default value. This button also selects the **Set** button in the **Action** group.



If the property has no default value, all values are cleared. If you have questions about default values, consult your project administrator.

- In the Multi-Choice dialog window:

- Do one of the following:

- To add choices to the current values, click **Add**, and then check the check box for each choice.

All previous choices are retained.

- To change the current values to different choices, click **Set**, and then check the check box for each new choice.

All previous choices are cleared. The checked choices become the new value for each object.

- To remove choices from the current values, click **Remove**, and then check the check box for each choice.

All other previous choices are retained.

You can use the buttons in the **Selection** group as follows:

- Click **Select All** to check all check boxes.
- Click **Unselect All** to clear all check boxes.
- Click **Set to Default** to check the check box for the default choice and clear any other checked choices. This button also selects the **Set** button in the **Action** group.



If the property has no default value, all check boxes are cleared. If you have questions about default values, consult your project administrator.

-

Click **OK** or press the enter key.

The **Action** column in the Edit Properties dialog window displays your action from the Multi-Choice dialog window. The displayed action is retained if you change the value back to the original one in the database.

- In the text field, enter the new value, and then press the enter key.

For a date or numeric property, the value must match the valid format shown in the tooltip.

For a text value, all keyboard characters are valid.



For a date property, you can click the calendar icon to display a calendar in a dialog window. To enter the new value from the calendar, do one of the following:

- Select the month, year, and day to enter that date.



Click **Today** to enter today's date automatically.



Click **TBD** to enter the value **TBD**, which marks the value for entry at a later date.

Click **OK** to close the calendar and apply the value.

To change an additional value, repeat steps 2 and 3.

4. To close the Edit Properties dialog window and apply your changes in the database, click **OK** or press the enter key.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Editing Properties in Table View Cells

In the following table views, you can change the values of editable properties directly in the cells:

- The hierarchical content table.
- The **Attachments** tab and window.
- The **Trace** subtab of the **Links** tab and window.
- The **Versions** tab and window.
- The Search Results dialog window.



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

You may want to add, remove, rearrange, resize, or sort columns. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), earlier in this chapter.



- You must have **Modify** permission for the objects.
- To edit properties for building blocks, TRAMs, or diagrams, you must have **Architect** privilege for the project.

To edit property values in table view cells:

1.

Click a cell containing a value that you want to change.

If the property has a choice value, one of the following dialog windows appears:

- In the Single-Choice dialog window, you can select only one choice.
- In the Multi-Choice dialog window, you can select any combination of choices.



The choice list may be set dynamically, and you may see different choices from time to time. If you have questions about the choice list, consult your project administrator.

A text field opens if the property has a date, numeric, or text value. For a date value, a calendar icon is displayed to the right of the text field.



- For date and numeric values, you can rest the pointer on the text field to see the valid format in a tooltip.
- For the system-defined property named **Text**, dimmed text in a cell indicates that editing is disabled because MHTML content is saved in Microsoft Word. You can enable this cell by changing the requirement's **Text Format** property value to **Text**, and then edit the **Text** property. For more information, see [Enabling Text Property Editing for a Requirement](#) in chapter 5, *Managing Requirements*.

2. Do one of the following:

- In the Single-Choice dialog window, click the button for the new value, and then click **OK** or press the enter key.

You can click **Set to Default** to select the button for the default value. If the default value is blank, no button is selected. If you have questions about default values, consult your project administrator.

- In the Multi-Choice dialog window:
 - To select the choices for the value, do one or both of the following:
 - Check the check box for each choice that you want to add, or click **Select All** to check all check boxes simultaneously.
 - Clear the check box for each choice that you want to remove, or click **Unselect All** to clear all check boxes simultaneously.

You can click **Set to Default** to check the check box for the default choice and clear any other checked choices. If the property has no default value, all check boxes are cleared. If you have questions about default values, consult your project administrator.

- Click **OK** or press the enter key.
- In the text field, enter the new value, and then press the enter key.

For a date or numeric property, the value must match the valid format shown in the tooltip.

For a text value, all keyboard characters are valid.



For a date property, you can click the calendar icon to display a calendar in a dialog window. To enter the new value from the calendar, do one of the following:

- Select the month, year, and day to enter that date.

-

Click **Today** to enter today's date automatically.

-

Click **TBD** to enter the value **TBD**, which marks the value for entry at a later date.

Click **OK** to close the calendar and apply the value.

To change an additional value, repeat steps 1 and 2.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Calculating Numeric Property Values With Formulas

Your project administrator can define numeric properties to use formulas that compute values automatically. When such numeric properties are assigned to folders, requirements, building blocks, or groups, you can use the formulas to update the values for selected parent objects.

In a given numeric property, the formula may be **Average**, **Maximum**, **Minimum**, **Multiply**, or **Sum**. The formula operates on that property's value for each direct child of the selected parent, and the corresponding cell for the parent is updated with the calculated value.

For example, assume that a numeric property named **Cost** uses the **Sum** formula and applies to requirements. Also, requirement *A* has two direct children: requirement *B*, whose **Cost** value is *10*; and requirement *C*, whose **Cost** value is *20*. When the **Cost** value is calculated for requirement *A*, that cell displays *30*, the total cost of the two direct children.



Calculations apply only to objects at the next level below a selected object. If a selected object has no children, its values are not updated.

You can select the parent objects in the hierarchical content table or the **Links** or **Versions** tabs or window. You may want to add, remove, rearrange, resize, or sort columns. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#), earlier in this chapter. For information about modifying a numeric property definition to use a formula, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about numeric properties, consult your project administrator.



- You must have **Modify** permission for the objects.
- To calculate properties for building blocks or TRAMs, you must have **Architect** privilege for the project.

To calculate numeric property values with formulas:

1. Select the parent objects, pull down the **Tools** menu, and choose **Calculate Properties**.



A shaded cell with the number in blue indicates that the value is not a formula result but was entered manually. If you select the corresponding object, the formula does not operate on the children and no calculation is made. The cell must be blank before the value can be calculated.

The Calculate Properties dialog window displays each numeric property that contains a formula and applies to a selected object type.

2. Check the check box for each property that you want to update, and then click **Calculate**.

In the column for each property, the values for all direct children are computed according to the formula. The result is shown in the corresponding cell for each selected parent.

Procedure Notes

Step 1: You can select nonadjacent and adjoining objects of the same type or of mixed types.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Updating Properties From an AP 233 STP File

To edit properties automatically, you can import property values from an AP 233 data file (.stp) that contains records for objects in the database. A .stp file contains data in the format defined by AP 233, the STEP⁴ application protocol for systems engineering data representation. In Architect/Requirements, the data represents folders and requirements, including properties and values, that were previously exported to the file. Any property that can be exported to an AP 233 file can be updated from the same file. For more information, see [Exporting Objects to an AP233 STP File](#) in chapter 4, *Maintaining a Project*.

You match file records with database objects by specifying a property named in the file as a unique identifier, a key that Architect/Requirements uses to locate the objects in the database. For all other properties in the file except the key, values are overwritten in the database for each object in the file. Updates are added to each object's change log. For more information, see [Viewing a Change Log](#) in chapter 12, *Using the Change Management Package*.



- You must have **Modify** permission for the objects.
- To update properties for building blocks, TRAMs, or diagrams, you must have **Architect** privilege for the project.

To update properties from an AP 233 STP file:

1. Do one of the following:
 - For a project's primary level, select the project node in the navigation tree.
 - For a folder's top level, select the folder in the navigation tree or in the hierarchical content table.
2. Pull down the **File** menu and choose **Import**→**AP233 (.stp)**.
The Open dialog window is displayed.
3. Select the data file and click **Open**.
The Import AP233 dialog window is displayed.
4. Select **Update Existing Objects**, and then enter the key in the **Index Property** field.



- The property value must be unique for each object. If duplicate values are found, processing stops and no properties are updated.
- The property name is case sensitive and the property must be recorded in the file.

⁴ Standard for the Exchange of Product Model Data (ISO 10303).

In the **Index Property** field, you can enter any user-defined property name or one of the following system-defined property names:

- **Name**
- **Text**
- **ROIN**
- **Number**
- **Source Paragraph**
- **Source Filename**
- **Baseline**
- **Version Type**
- **Security Profile**
- **Subtype**

5. Click **OK**, or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

The Import AP233 dialog window closes, and a message is displayed to indicate that the import is in progress. When the import is complete, a confirmation message shows the number of objects for which the property is updated.

Using the Live Excel Interface

Through the Architect/Requirements live Excel interface, you can use Microsoft Excel to edit properties for selected objects. Live Excel generates an Excel file in which you can change editable property values for objects in the following views:

- The hierarchical content table.
- The **Links** and **Versions** tabs and floating windows.
- The **Defining Object Traceability** and **Complying Object Traceability** windows.
- The Search Results dialog window.



- You must have **Modify** permission for the objects.
- To edit properties for building blocks, TRAMs, or diagrams, you must have **Architect** privilege for the project.
- Microsoft Office Excel 2013 or Excel 2016 and the Microsoft .NET Framework 4.0 CLR must be installed on your computer.

Types of Live Excel Sessions

You can use live Excel while the Architect/Requirements client is running and connected to the database. You can also open an existing live Excel file in an independent live Excel session, without running the client but with the option to connect to the database.

Live Excel Sessions With the Architect/Requirements Client

With the Architect/Requirements client running, you can create a new live Excel file that contains a row and column for each object and property that you export from the active view. Or, you can open an existing live Excel file from a previous session.

Preparing Architect/Requirements Property Columns for Live Excel Export

Before starting live Excel, you may want to display or hide the children of parent objects, add columns to display other properties, or remove displayed property columns. For more information, see [Viewing Objects in a Hierarchy](#) in chapter 3, *Using the Architect/Requirements Main Window*, and [Adding and Removing Columns](#), earlier in this chapter.

Object and Property Selection for Live Excel Export

For a new file, you specify the objects and properties that you want to export. You can export only the currently selected objects, or you can export all displayed objects whether or not they are selected.

In addition, you can use an Excel template to specify the properties and other information to export, or you can export only the properties that are currently displayed in the view. For more information about Excel templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about Excel templates, consult your project administrator.

Live Excel Synchronization

During a live Excel session, the worksheet is synchronized with the Architect/Requirements database and client. Editable values that you change in the worksheet are immediately applied to the database, and then are displayed in the client. Values that you or other users change in the client are updated automatically in the worksheet.

Independent Live Excel Sessions

You can use an existing live Excel file to change editable property values without running the Architect/Requirements client. You can also start the client from an existing live Excel file.

Database Connection Options for Independent Live Excel Sessions

When you open an existing live Excel file independent of the Architect/Requirements client, you have the following options:

- Connect to the Architect/Requirements server. Changes in the live Excel file are immediately applied to the corresponding properties in the Architect/Requirements database.
- Remain disconnected from the Architect/Requirements server. Changes in the live Excel file are accumulated until pending changes are applied to the corresponding properties in the Architect/Requirements database.

Starting Architect/Requirements From an Independent Live Excel Session

You can start Architect/Requirements from a live Excel file by doing the following:

1. In the open file, select an object type indicator.
A login page is displayed.
2. Enter your user name and password.
The main window displays the object highlighted in the content table.

For more information, see [Starting a New Architect/Requirements Session](#) in chapter 3, *Using the Architect/Requirements Main Window*.

User Actions in Live Excel Files

You can copy or move a value to any other cell, and you can create Excel formulas to edit values automatically. With Excel's AutoFilter feature, you can hide and show rows according to criteria for any property column. And you can navigate to objects in the view by selecting object type indicators in the file.



- Some actions clear the queue for Excel's **Undo** option and cannot be reversed. Subject to those conditions, you can reverse an action by clicking **Undo** button on Excel's title bar, or by pressing control-Z.
- Some changes that you can make in the file do not affect the database or the view. For example, inserting or deleting a row in a worksheet does not create or delete an object in the database. Nor does inserting, deleting, rearranging, or resizing columns in a worksheet change the property columns displayed in the view.

Considerations for Sorting Data

If you sort the data in a live Excel file, the object type indicators and the objects may be mismatched. This condition occurs in Microsoft Office Excel when a cell contains a graphic that is taller than the row height. For a height that accommodates the indicators, increase the size of Excel's standard font.

The effective size depends on the standard font that you use. With Arial, for example, a size of 11 points allows the indicators to be sorted with the correct objects. To increase the size, click Excel's **Office Button**, click **Excel Options**, and select the new size in the **Popular**→**When creating new workbooks** section.



You must exit and restart Excel for this change to take effect. If you want to change the font size, do so before creating or opening a live Excel file.

Microsoft Office Excel Comment Indicators

If Excel's comment indicators are enabled when you select a cell, an indicator marks the cell if it contains a property value. The comment shows the property name and whether the property is editable. If the property has a choice value, the comment also shows the choices.



Values that you cannot change are dimmed in the cells.

- To enable or disable indicators, click Excel's **Office Button**, click **Advanced**, and select the related option from the **Display** section in the right pane.
- To view a comment, rest the pointer on the cell.

Editing Properties in a Live Excel Session With the Architect/Requirements Client

1. With the view activated, do one of the following:



To create a new file:

.

Pull down the **File** menu and choose the **Export**→**Excel Spreadsheet** options to display the Export To Excel dialog window.

You can also right-click inside the view and choose the **Export**→**Excel Spreadsheet** options from the pop-up menu.

.

Under **Object Selection**, do one of the following:

- o To export only the selected objects, click **Export Selected Objects**.
- o To export all objects in the view, click **Export All Objects in View**.

.

Under **Formatting**, do one of the following:

- o To export the properties in the currently displayed columns, click **Use Current View Columns**.
- o To export the properties specified in an Excel template, click **Use Excel Template**.

The template selection field becomes available. You can accept the default Excel template or select a different template from the list.



- o A template may contain multiple worksheets, each worksheet with different properties. Therefore, data may be separated on worksheets in the live Excel file.
- o If properties in the template do not apply to an object type specified for export, the inapplicable property values are indicated in the related cells.

For more information about Excel templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about templates, consult your project administrator.

.

Under **Output**, click **Excel Live**, and then click **OK**.



If your live Office and Architect/Requirements versions do not match, a message asks you to reinstall live Office. Click **OK** to start the uninstaller and follow the instructions. Click **Finish** to start the installer, follow the instructions, and click **Finish**. A message may ask you to restart your computer to complete the installation. After installation, repeat steps a. through d.

Live Excel opens a read-only **.html** file. Values that you cannot change in the database are dimmed in the worksheets. Go to step [2](#).

- To open an existing file, pull down the **File** menu and choose **Excel Live Open**.

You can also right-click inside the view and choose the **Excel Live**→**Open** options from the pop-up menu.

Live Excel starts with Excel's Open dialog window. When you select the file and click **Open**, one of the following is displayed:

- . A message stating that there are no pending changes in the file.

This file does not contain a new value for any property in the database. However, the database may contain new values that are not reflected in the file. When you click **OK**, all worksheets in the file are updated with changes to the database since the file was last saved. Go to step 2.

- . A message asking if you want to check in pending changes before refreshing the file from the Architect/Requirements server.

This file contains new values that were entered without a connection to the server, and these changes have not been applied to the properties in the database. For other properties, the database may contain new values that are not reflected on any worksheet in the file.

- o To apply the pending changes to the database, click **Yes**.

First, pending changes are applied to the database and displayed in the main window. Second, all worksheets in the file are updated with other changes to the database since the file was last saved.



During this process, other users may edit the same properties for which the file contains pending changes. Therefore, those values also may be updated in the file when the process is complete.

- o To discard the pending changes, click **No**.

All worksheets in the file are updated with changes to the database since the file was last saved.

2. In the live Excel file, do any or all of the following:



Values that you cannot change in the database are dimmed in the worksheets.



To change a value:

- Double-click the cell that contains the value.



Do not change the value in the Architect/Requirements client while the cell is in edit mode. Otherwise, live Excel displays an error message stating that an edit is in progress.

A pop-up list appears if the property has a choice value.



The choice list may be set dynamically, and you may see different choices from time to time. If you have questions about the choice list, consult your project administrator.

A text field opens if the property has a date, numeric, or text value.



For the system-defined property named **Text**, the value cannot be changed in the Architect/Requirements database for a requirement with MHTML content saved in Microsoft Word. Although you can edit the cell in the live Excel file, when you click outside the cell, an error message states that you cannot overwrite rich text.

You can enable this cell by changing the requirement's **Text Format** property to **Text**, and then edit the **Text** property. For more information, see [Enabling Text Property Editing for a Requirement](#) in chapter 5, *Managing Requirements*.

- Do one of the following:
 - o In the pop-up list, select the choice or choices for the new value, and then click another cell to close the list.

Buttons indicate a single-choice list. You can select only one choice.

Checkboxes indicate a multiple-choice list, in which you can select any combination of choices. Selecting an unchecked choice adds it to the value, and selecting a checked choice removes it from the value.
 - o In the text field, enter the new value, and then press the enter key.

For a date or numeric property, the value must match the valid format for the property.

For a text value, all keyboard characters are valid.

You can also select the cell and enter the new value directly.



If you create an Excel formula, you can automatically update values in all dependent cells when you change the value in a precedent cell. For more information, see [Microsoft Office Excel Help](#).

- To copy or move a value:
 - Select the source cell, and choose **Copy** or **Cut** on Excel's **Home** ribbon.
 - Select the destination cell, and then click **Paste** on Excel's **Home** ribbon.



If you cut the value and paste it into a different row, the value remains with the source object in the database. Therefore, the value is not moved in the client view.

- To hide or show rows, click the button in the heading of a property column, and then select a filtering criterion from the pop-up list.



To open a requirement in Word for editing or viewing:

- Select any data cell except **Home** in the row for the requirement.
- Do one of the following:
 - In Excel, on the **Add-Ins** ribbon:

- For editing, click the **Open** button.

When you save the Word file, your edits are applied to the requirement's **Text** property value.



If the live Excel file contains the **Text** property, click **Refresh** on the **Add-Ins** ribbon to update the value. This action also updates the value in Architect/Requirements. Conversely, clicking **Refresh** on the Architect/Requirements toolbar also updates the value in live Excel.

- For viewing, click the **Open Read-only** button.

Although you can edit this file, the changes cannot be saved in live Excel or in Architect/Requirements.



To navigate to an object in the main window view, do one of the following in the **Home** column of the worksheet:

- Click the blank space in the cell containing the object type indicator.
- Use the arrow keys to select the cell containing the object type indicator.

The object is displayed and highlighted in the view.



You can update the file with the latest data from the Architect/Requirements database by clicking the **Refresh** button on the **Add-Ins** ribbon.

3. To end the live Excel session, click the Excel's **Office Button** and choose **Exit Excel**.

Live Excel displays a message asking if you want to save the changes.

- If you click **No**, your changes are applied in the database but a permanent file is not created.

The temporary file remains on your computer until you exit Architect/Requirements. Then the file is deleted from your computer.

- If you click **Yes**, Excel displays the Save As dialog window.

For this permanent file, assign the file name, file type, and location outside the Architect/Requirements database, for example, on a local drive.



- . For a live Excel file, Siemens PLM Software recommends that you assign the **.xlsm** file type. Only the **.xlsm** file type supports the full capability of live Excel.
- . If you save a live Excel file as the **.mhtml** file type, it becomes static and loses the live capability permanently.

Editing Properties in an Independent Live Excel Session

1. With the live Excel file open, initiate the live Excel session by selecting any cell on any worksheet.



If your live Office version does not match your Architect/Requirements version, a message asks if you want to reinstall live Office. Click **OK** to start the uninstaller and follow the instructions. Click **Finish** to start the installer, follow the instructions, and click **Finish**. A message may ask you to restart your computer to complete the installation. After installation, repeat this step.

A message asks if you want to connect to the Architect/Requirements server.

2. Do one of the following:

- To log in to the server for immediate database changes, click **Yes**.

Microsoft Internet Explorer displays the Teamcenter systems engineering login page. Enter your Architect/Requirements user name and password in the **User Name** and **Password** fields, and then click **Log In**.

Live Excel displays one of the following:

- o A message stating that there are no pending changes in the workbook.
This worksheet does not contain a new value for any property in the database. However, the database may contain new values that are not reflected in the worksheet.
When you click **OK**, a progress indicator appears, and the worksheet is updated with changes to the database since the file was last saved.
- o A message asking if you want to check in pending changes before refreshing the worksheet from the Architect/Requirements server.

This worksheet contains new values that were entered without a connection to the server, and these changes have not been applied to the corresponding properties in the database. For other properties, the database may contain new values that are not reflected in the worksheet.

- To apply the pending changes to the database, click **Yes**.

A progress indicator appears during this process, which has two parts. First, pending changes are applied to the database. Second, the worksheet is updated with other changes to the database since the file was last saved.



During this process, other users may edit the same property values for which the worksheet contains pending changes. Therefore, those values also may be updated in the worksheet when the process is complete.

- To discard the pending changes, click **No**.

A progress indicator appears, and the worksheet is updated with changes to the database since the file was last saved.

- To remain disconnected from the server and accumulate changes, click **No**.

Live Excel displays a message stating that you are working offline. Changes are accumulated and can be applied to the database by logging in later. Values that you cannot change in the database are dimmed in the worksheet.

3. In the live Excel file, do any or all of the following:



Values that you cannot change in the database are dimmed in the worksheets.



To change a value:

- Double-click the cell that contains the value.

A pop-up list appears if the property has a choice value.



The choice list may be set dynamically, and you may see different choices from time to time. If you have questions about the choice list, consult your project administrator.

A text field opens if the property has a date, numeric, or text value.



For the system-defined property named **Text**, the value cannot be changed in the Architect/Requirements database for a requirement with MHTML content saved in Microsoft Word. Although you can edit the cell in the live Excel file, when you click outside the cell, an error message states that you cannot overwrite rich text.

You can enable this cell in this live Excel session if the file contains the **Text Format** property and is connected to the Architect/Requirements server. For more information, see [Enabling Text Property Editing for a Requirement](#) in chapter 5, *Managing Requirements*.

If the file is disconnected from the Architect/Requirements server, cell editing cannot be enabled in this live Excel session.

- Do one of the following:

- In the pop-up list, select the choice or choices for the new value, and then click another cell to close the list.

Buttons indicate a single-choice list, in which you can select only one choice.

Checkboxes indicate a multiple-choice list, in which you can select any combination of choices. Selecting an unchecked choice adds it to the value, and selecting a checked choice removes it from the value.

- In the text field, enter the new value, and then press the enter key.

For a date or numeric property, the value must match the valid format for the property.

For a text value, all keyboard characters are valid.

You can also select the cell and enter the new value directly.



If you create an Excel formula, you can automatically update values in all dependent cells when you change the value in a precedent cell. For more information, see Microsoft Excel Help.

- To copy or move a value:
 - Select the source cell, and choose **Copy** or **Cut** on Excel's **Home** ribbon.
 - Select the destination cell, and then click **Paste** on Excel's **Home** ribbon.



If you cut the value and paste it into a different row, the value remains with the source object in the database. Therefore, the value is not moved in the view, although the value is moved to the destination cell in Excel.

- To hide or show rows, click the button in the heading of a property column, and then select a filtering criterion from the pop-up list.
4. To end the live Excel session, click the Excel's **Office Button** and choose **Exit Excel**.

Live Excel displays a message asking if you want to save the changes.

- Click **Yes** to display Excel's **Save As** dialog window.
- Assign the file name, file type, and location outside the Architect/Requirements database, for example, on a local drive.
- Click **Save** to close the file.



- For a live Excel file, Siemens PLM Software recommends that you assign the **.xlsm** file type. Only the **.xlsm** file type supports the full capability of live Excel.
- If you save a live Excel file as the **.mhtml** file type, it becomes static and loses the live capability permanently.

Creating Objects in Live Excel

In addition to editing properties, you can use live Excel to add new folders, requirements, building blocks, and groups to the Architect/Requirements database. For an existing object in a live Excel file, you create a sibling object that has the same object type and subtype.



- The live Excel file must be connected to the Architect/Requirements server. For more information, see [Types of Live Excel Sessions](#), earlier in this chapter.
 - To create building blocks, you must have **Architect** privilege for the project.
 - You cannot create notes, diagrams, or spreadsheet objects in live Excel.
 - Microsoft Excel 2013 or Excel 2016 must be installed. The Microsoft .NET Framework 4.0 CLR must also be installed. For more information, see [Prerequisites for Using the Live Office Interface](#) in chapter 2, *Installing the Architect/Requirements Client with Office Integration*.
- Right-click a cell for the existing sibling, and then choose the **Teamcenter for systems engineering**→**New Object** options from the pop-up menu.



- o Ensure that the entire row is not selected. Otherwise, a different pop-up menu is displayed.
- o You can right-click any cell in the row within the table of objects and properties. You cannot select the options if you right-click a cell outside that table.

In the live Excel file, the new object is added on a new row directly below the sibling. In the client, the new object is placed at the level of the sibling according to the current sort sequence.



If the sibling has a property set by an activator, the new object's value is blank in the corresponding cell. If you have questions about activators, consult your project administrator. For more information about activators, see the *Systems Architect/Requirements Management API Reference* manual.

Repeat this procedure for each additional object that you want to create.

Creating Trace Links in Live Excel

You can use live Excel to create trace links between objects in Architect/Requirements. For more information, see chapter 7, [Showing Object Relationships With Trace Links](#).

For defining and complying objects, you can do the following:

- Start the trace links in the live Excel file and end the trace links in the Architect/Requirements client.
- Start the trace links in the Architect/Requirements client and end the trace links in the live Excel file.
- Start and end the trace links in the live Excel file.

The live Excel file must be connected to the Architect/Requirements server. For more information, see [Types of Live Excel Sessions](#), earlier in this chapter.



Microsoft Excel 2013 or Excel 2016 must be installed. The Microsoft .NET Framework 4.0 CLR must also be installed. For more information, see [Prerequisites for Using the Live Office Interface](#) in chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

1. Select the defining object or objects, right-click the selection, and choose the **Teamcenter for systems engineering**→**Start Link** options from the pop-up menu.



- Ensure that the entire row is not selected. Otherwise, a different pop-up menu is displayed.
- If you select only one defining object, you can select multiple complying objects. If you select multiple defining objects, you can select only one complying object.

2. Select the complying object or objects, and then right-click the selection and do one of the following:
 - For the default trace link subtype, choose the **Teamcenter for systems engineering**→**End Link** options from the pop-up menu.
 - For a user-defined trace link subtype:
 - Choose the **Teamcenter for systems engineering**→**End Link as Subtype** options from the pop-up menu to display the Select Subtype dialog window.
 - Select a trace link subtype, and then click **OK** to close the dialog window.

Disconnecting a Live Excel Spreadsheet

Any live Excel spreadsheet can be permanently disconnected from the Architect/Requirements database. When disconnected, the spreadsheet is made static and cannot be used to edit object properties interactively.



Microsoft Excel 2013 or Excel 2016 must be installed.

The Microsoft .NET Framework 4.0 CLR must also be installed on your computer.

1. Do one of the following:

- In Microsoft Office Excel, click the **Non-Live** button on the **Add-Ins** ribbon.

A confirmation message states that the spreadsheet cannot be converted to a live spreadsheet again, and asks if you want to continue.

2. Click **Yes**.



This action is permanent and cannot be reversed.

The spreadsheet is disconnected, and the current objects and properties remain in the static file.

Attaching a Spreadsheet to an Object

To store equations in the Architect/Requirements database, you can create spreadsheet objects containing worksheets from existing Microsoft Excel files. Each spreadsheet can store equations so that they are not overwritten by updated property values when the spreadsheet is opened.

For example, equations may be entered in property cells in a file containing object data previously exported from Architect/Requirements. If a spreadsheet is created from a live Excel file, equations entered in property columns are automatically preserved when the spreadsheet is opened and the properties are refreshed. A spreadsheet can also be created from an Excel file that does not contain equations or object data. For more information, see [Using the Live Excel Interface](#), earlier in this chapter, and [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.

You select the spreadsheet owner in the hierarchical content table. For that object, the new spreadsheet is displayed in the **Attachments** tab or window. The spreadsheet remains in the database and the equations can be edited in Excel until the spreadsheet object or its owner is deleted. In addition, the spreadsheet object can be copied or moved to a different owner. For more information, see [Attachments Tab](#) or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*; and [Copying Objects](#), [Moving Objects](#), or [Deleting Objects](#) in chapter 4, *Maintaining a Project*.

To attach a spreadsheet to an object:

1.

Select the object, and then pull down the **File** menu and choose the **New→Spreadsheet** options.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder. If the list does not display the spreadsheet, you can use the **Look in** field to change the drive or folder, or use the **Files of type** field to see other types of files.

2. In the list, select the spreadsheet that you want to use for the new spreadsheet object, and then click **Open**.

A message is displayed while the spreadsheet is created. Then the **Attachments** tab or window displays the new spreadsheet object, with a default name in an open text field.

3. Enter the spreadsheet name in the text field, and then press the enter key.

Procedure Notes

Step 1: You can also right-click the object and choose the **New→Spreadsheet** options from the pop-up menu.

Storing Property Values Through Reference Links

A reference link is a placeholder for the current value of a property. You can create reference links in the Microsoft Word content of requirements, notes, change logs, and change approval objects. Each time a containing object is opened in Word, reference link values are updated automatically. Thus, a single property of a target object can be referenced by many other objects.



Changes to either a containing object or the target object do not trigger change events on the other end of the reference link.

Reference Link Concepts

This section provides conceptual information about reference links.

Reference Link Types

You can create the following types of reference links:

- Plain text, consisting solely of a text string.

This type of reference link targets a plain text property. Plain text properties apply to all object types and can be displayed in all views in the Systems Engineering and Requirements Management module.

- Full content, which can consist of HTML and embedded elements such as tables, graphics, and OLE objects.

This type of reference link targets the full content of a requirement, a note, a change log, a change approval object, or a live Visio diagram. Only these object types can be targeted by full content reference links.



Right-click a live Visio diagram and select **Copy Snapshot**. This copies the diagram's snapshot image, which can then be pasted into a requirement in Microsoft Word.

The full content is referenced through a note that is automatically attached to the diagram object when it is saved. This note contains an image (.gif) of the diagram.

The diagram image note must be selected as the target, not the diagram object itself.

For more information, see [Creating a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

Reference Link Target Objects

The target object can be selected in the following:

- The hierarchical content table.
- The **Attachments, Links, Connectivity, Where Used, or Versions** tab or window.
- The Search Results dialog window.
- The **Complying Object Traceability** window or the **Defining Object Traceability** window.

If a shortcut is selected as the target object, the reference link relation is created to the selected shortcut and not to its primary object. If the shortcut has its own unique value for the referenced property, the value of the shortcut appears when the reference is resolved into the referencing object's text. However, if the shortcut does not have its own unique value, the information from the primary object is used when the reference is resolved. In either case, the reference relation appears in the **Where Used** and **Links/Relations** tabs for the shortcut object and not for the primary object. The link is broken if the shortcut is deleted, even if the primary object exists.

Behavior of Reference Links

In Architect/Requirements, objects that contain rich text, requirements, and notes can dynamically include information from other objects. The referenced information can be either a property value or the text content of a requirement or note. When a reference targets a versionable object, Architect/Requirements determines which version of the information to use. This section describes the behavior of these reference links.

Fixed and Dynamic Version References

A reference to a versionable object can be either fixed or dynamic. Fixed references always get the information from the same version of the object. Dynamic references get the information from the currently effective version of the object.

There are two ways to specify how dynamic links behave. The **Project Settings** property of a project controls the default behavior for reference links in that project. Selecting the **Promote References** choice indicates dynamic references. By default, a project uses fixed references.

The behavior for a specific link can be controlled with the **Reference Settings** property (which can be accessed by right-clicking a reference link). Double-clicking the **Reference Settings** property allows you to select or deselect the **Promote References** value. The **Reference Settings** property allows a specific reference link to override the default behavior.



Although the **Reference Settings** property is a multi-select property, the **Promote References** and **Fixed References** options are mutually exclusive. If you select both the options, **Promote References** takes precedence.

The use of fixed or dynamic references does not affect how the references are stored in the database. The setting for a project can be changed at any time to change the behavior of the existing references.

The **Reference Settings** property requires **Read & Write** privileges to the objects for which the reference link is created. When nothing is selected in **Reference Settings**, the project's default setting is used. Choosing **Promote References** or **Fixed References** overrides the project setting for the reference link.



The **Promote References** behavior is supported for references to versionable objects, requirements, buildings blocks, and shortcuts to versionable objects. References to notes do not support the **Promote References** behavior.

Reference Promotion and Demotion

When fixed references are used and a new version of a referenced object is created, Architect/Requirements decides which version to use for the reference based on the state of the referencing object. If the referencing object is baselined or frozen, then the reference remains

with the old version. This behavior prevents modifications to the new version from indirectly modifying the content of a frozen or baselined object.

If the referencing object is not frozen, then the reference is promoted to the new version. This behavior ensures that the non-frozen objects reference the latest information.

When the current non-frozen version of an object is deleted, any references to it are shifted back to the prior frozen version. When dynamic references are used, the reference shows information from the currently effective version of the referenced objects. The promotion and demotion of the reference is dynamic. Even if the referencing object is frozen or baselined, the content can appear to change. This does not compromise the content of a baseline. Baselined content is preserved but the baseline effectivity must be set to display it.

Indirect Changes to Text Content

Modifying a referenced object indirectly causes a change to the text content of the referencing object because reference links dynamically access information from the referenced object. This behavior may not be desirable if the referencing object is frozen or baselined.

Architect/Requirements allows references from frozen to non-frozen objects to exist. A warning is displayed when an object is frozen that has a reference to a non-frozen object. You can take necessary steps to avoid this situation. If needed, activators can be used to prevent indirect changes to text content.

Broken References

In cases where the referenced information cannot be retrieved, the **[Broken Link: ...]** string is inserted in the text in place of the reference and the previously referenced information is included as plain text. References are broken when the referenced object is deleted and there is no prior version to use instead. With dynamic references, a reference appears broken when there is no version of the referenced object that appears in the current effectivity. In this case, a different broken link indicator is used, such as **[no effective version for <name of referenced object>: ...]**.

Illustration

Here is an illustration of how a reference behaves in a particular scenario. For the illustration:

- . Create two requirements, Req1 and Req2-A.
- . Edit the text of Req1 and include a reference to the name of Req2-A.
- . Freeze Req2-A, create a new version from Req2-A and name it as Req2-B.
- . Create a baseline that includes Req1 and Req2-B.
- . Create a new version of Req2-B named Req2-C.
- . Freeze Req2-C and create a version named Req2-D.

If fixed references are used, the content of Req1 displays the name of the baselined version of Req2 and Req2-B regardless of the effectivity setting. The reference was promoted to Req2-B when Req2-A was versioned because Req1 was not frozen at that time. After Req1 was baselined, references are no longer promoted; the reference remains at Req2-B.

If dynamic references are used, then the content of Req1 changes based on effectivity. Req2-B is displayed in the baseline effectivity, Req2-C in current frozen version, and Req2-D in current version. Req2-A could be displayed in a frozen as of date effectivity.

When the **Promote References** option is not enabled, effectivity has no impact on reference links. References are to a specific version. Effectivity will control which objects you will see in the user

interface: A, A-1, B, B-1, B-2, or C. However, for any object that appears, its content that is referenced from other objects comes from a specific version of that object, regardless of the effectivity.

For example, frozen A-1 references frozen B-1, and C references the non-frozen B-2:

- A-1's content will always show information pulled from B-1, even if **Effectivity** is **Current Version** and it is B-2 that currently appears in the client.
- Likewise, C's content will always show information pulled from B-2, even if **Effectivity** is **Current Frozen Version** and it is B-1 that currently appears in the client.

When the **Promote References** property setting is enabled, reference links are sensitive to effectivity. For example, consider a frozen object A referencing a non-frozen object B. B1 is created as the new version of B. Enabling the **Promote References** option resolves the reference link from A to B in **Current Frozen Version** effectivity and to B1 in **Current Version** effectivity.

The above example (with objects A, B, and B1) is used to illustrate the similarities and differences in the resolution of reference links with and without enabling the **Promote References** option. The behavior can be seen in the **Links→Relations** and the **Where Used** tabs in the notebook pane.

Table 9-2 lists the resolution of the reference links with the **Promote References** option disabled. Table 9-3 lists the resolution of the reference links with the **Promote References** option enabled.

Table 9-2. Reference Links with Promote References Disabled

Tab in the Notebook pane	Effectivity: Current Version	Effectivity: Current Frozen Version (same behavior)
Links→Relations	Selecting A displays B as the target. Selecting B1 displays no reference.	Selecting A displays B as the target. Selecting B displays A as the source.
Where Used	Selecting B1 displays no reference.	Selecting B displays A.

Table 9-3. Reference Links with Promote References Enabled

Tab in the Notebook pane	Effectivity: Current Version	Effectivity: Current Frozen Version (same behavior)
Links→Relations	Selecting A displays B1 as the target. Selecting B1 displays A as the source.	Selecting A displays B as the target. Selecting B displays A as the source.
Where Used	Selecting B1 displays no reference.	Selecting B displays A.

Copying Objects That Contain Reference Links

When copying objects that contain reference links:

- If the reference link is not located within the copy region (but is located within the same project), the new reference link points to the same target it originally pointed to.
- If the reference link is located within the copy region, the new reference link points to the new object created inside the copy region.
- If the reference link is located in another project and is not within the copy region, the new reference link is disabled and is represented by the label [**Broken Link: contents**].

The *contents* variable represents the last actual value of the property before the reference link was disabled.

Modifying Objects That Contain Reference Links

If object *A* contains reference links to object *B*:

- Reference links between *A* and *B* cause no special restriction on modifying or deleting either one.

If more access control is desired, security profiles can be applied to the objects.

For more information about access control and security profiles, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

- If referenced properties of *B* are changed, the reference link values in *A* are indirectly changed, even without the **Modify** permission for *A*.

The reference link values are updated automatically the next time *A* is opened in Microsoft Word. If *A* is currently open, the reference link values are not updated until the Word file is closed and then reopened.

All the above is true even if *A* is frozen.

Deleting Objects That Contain Reference Links

If object *A* contains reference links to object *B*:

- If *A* is deleted, the reference links to *B* are also deleted. The **Modify** permission for *B* is not necessary.
- If *B* is deleted, the reference link values in *A* are preceded by the label **[Broken Link]**, and the values are resolved as follows:
 - Plain text property references include the old value. For example:

[Broken Link: 1234-5]



If object *A* contains a reference link to a plain text property *X* of object *B*, the above behavior also applies if the type definition for *B* is changed so that property *X* is no longer used for that object type.

For information about modifying a type definition, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

- For full content reference links, the old value may consist of complex elements such as tables, graphics, and OLE objects. This value may exceed the effective size for display. Therefore, the value is a URL that is an artifact of how the body text references are handled as subdocuments in Word. For example:

[Broken Link: url]

- If the deleted *B* is restored from Recycle Bin, the references in *A* are fully restored.



If you save a Word document containing broken links, the reference link values are not restored even if you restore the deleted object from your Recycle Bin.

- Once *B* is discarded from Recycle Bin, references are lost and only the **[Broken Link]** labels remain.

All the above is true even if *A* is frozen.

For more information, see [Deleting Reference Links](#), later in this chapter.

Freezing Objects That Contain Reference Links

If object A references information from object B, and neither is frozen:

- When you freeze A making it A-1, a warning message displays **References content from unfrozen object**.
- Freezing A does not impose any special access control or limitations on B.
- If B remains unfrozen, then A-1's content will change whenever the referenced content in B changes.
- It is up to the user whether and when to freeze B.

If objects A-1 and C reference information from object B-1, and A-1 and B-1 are both frozen, but C is not:

- When a new version B-2 is created for B, it continues to refer to the frozen B-1 because A-1 is frozen.
- When a new version B-2 is created for B, it now refers to the new version B-2 because C is not frozen.

Restating the above examples as a rule:

- References from a non-frozen object to other objects are always promoted to reference the latest versions of those other objects.
- References from a frozen object to other objects are never promoted to reference new versions of those other objects.

Exporting and Importing Objects That Contain Reference Links

This section provides information on importing and exporting objects that contain reference links to XML data files, to AP 233 data files, and to Microsoft Word.

XML Export/Import

If the reference link is located within the export region, the new reference link points to the new object created inside the export region.

If the reference link is located in another project and is not within the export region, the new reference link is disabled and is represented by the label **[Broken Link: contents]**. The *contents* variable represents the last actual value of the property before the reference link was disabled.

- For a plain text property, the value is the entire value.
- For a full content property, the value is only the plain text of the content and does not include elements such as tables, graphics, or OLE objects.

If both ends of the reference (A & B) are internal to the export region, a new reference is created at import between the newly created A' and B'.

If A is exported, but the referenced object (B) is external to the export region:

- Plain text property references appear in the imported A' as local text.
- Full content references will appear in the imported A' as an invalid URL artifact.

AP 233 Export/Import

- The AP-233 standard has no concept of reference links.
- References in object A's text are fully expanded when exported.
- References to full content are expanded using only the plain text property value. Therefore, markup, tables, graphics and OLE objects are lost from referenced content.
- At import, all of A's text is local, with no references. Therefore, all reference links are lost in an AP-233 export and import round trip.

Microsoft Word Export

- All referenced content is exported with fresh, current values.
-

For the folder containing the export documents, the document template has a new property named **Document Template Rules**.

- When this property value includes **Preserve References**, reference links to full content are exported as hyperlink URLs.
- Otherwise, reference links are expanded inline for a clean document.
- Reference links are never created during document import.

Access Control for Objects That Contain Reference Links

Create a reference in object *A* to content of *B*:

- You must have Modify permission for *A* where the reference is being created.
- Read access to *B* is adequate to create a reference to its content.

View object *A* that includes content from *B*:

- You must have Read rights to *A* to see that object at all.
- If you can view *A*, but do not have Read rights to *B*, then the token [No Access] appears in *A*'s text where the reference occurs.
- If you edit *A* when the [No Access] token is present:
 - If you leave the token in place when you save, the reference remains intact and other users with Read rights to both *A* and *B* still see the referenced information.
 - If you remove the token, the reference is deleted.

For more information about access control and security profiles, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

Dynamic vs. Static Behavior

If object *A* references information from object *B*:

- When values in *B* change, there is no immediate update to the database content of *A*. The change log for *A* never reflects changes due to the referenced content from *B*.
- When *A* is opened for edit in Word, it always includes fresh content of references from *B*. When the edit is saved, the current values from *B* are included within *A*'s saved text.
- When you search, string matching against *A* will see those embedded *B* values last saved from editing *A*. Those values may or may not be the true current values in *B*.
- When *A*'s text is displayed in the Preview tab, it always includes fresh content for all references. This does not refresh the database content for *A*.
- For export to Word, fresh content is exported, and there is no database change.
- For **getValue** (Text, HTML, MHTML) calls, fresh content is exported, and there is no database change: A **getValue/setValue** round trip on MHTML refreshes the database content. This can be an expensive process when graphics and OLE objects are involved.

Creating a Plain Text Reference Link

1. Select the requirement, note, change log, or change approval object in which you want to create the reference link.
2. Select **File**→**Open** to open an **.mhtml** Word file.
3. Display the target object.
4. In the row for the target object, right-click the plain text property value and choose **Copy Reference Link** from the pop-up menu.



For a requirement, a note, a change log, or a change approval object, right-clicking in the leftmost column copies the object's full content rather than a plain text value. For an object of any other type, right-clicking in the leftmost column copies the object name.

You can also right-click the plain text property value in the **Properties** tab or window or the Edit Properties dialog window.

5. In the Word file, paste the reference link at the desired position in the content.



For plain text properties, you can paste reference links that point to the same object that is open in Word. For example, within the Word file for a given object, you can paste a reference link to the name of that same object.

The value is inserted as a hyperlink, which you can click to navigate to the object.

6. Click Microsoft Word's **Office Button** →**Save**.

If the property value is changed after you close the file, the reference link value remains the same until you reopen the object. At that time, the reference link value is updated automatically to match the property value.

Creating a Full Content Reference Link

1. Select the object to contain the link, pull down the **File** menu, and choose **Open**.
2. Display the target object.



For a target diagram, open and save the diagram if the following are true:

- It is from an Architect/Requirements version earlier than 2007.1.
- This is the first full content reference link created to the diagram.

Saving the diagram automatically generates a note containing the diagram image (.gif), through which the diagram's full content is referenced.

3. In the row for the target object, right-click the leftmost column, and then choose **Copy Reference Link** from the pop-up menu.

Depending on the view, the leftmost column is:

- The **Name** column in the content table, the **Versions** tab or window, the **Complying Object Traceability** window, the **Defining Object Traceability** window, and the Search Results dialog window.
- The **Defining Trace** or **Complying Trace** column in the **Trace** subtab.
- The object type indicator column in any other view.



- If you right-click in a property column, you copy a plain text value rather than the object's full content.
- For a live Visio diagram, select the diagram image note as the target, not the diagram object itself.

For more information, see [Creating a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*

You cannot copy full content from the **Properties** tab or window or the Edit Properties dialog window.

4. In the Word file, paste the reference link at the desired position in the content.



You cannot paste a reference link that points to the full content of the same object that is open in Word.

The current value is inserted. Initially, the reference link value displays only the plain text in the content and the URL of the object, with the entire value formatted as a hyperlink. The reference link resolves to the full content after you store it in the database, close this file, and reopen the object in Word.

5. Click Microsoft Word's **Office Button** → **Save**.

After you close this file and reopen the object:

- The reference link value is preceded by a down arrow and a right arrow. You can click the down arrow to open the target object for editing. You can click the right arrow to navigate to the target object.
- The reference link is displayed as a subdocument in Word's outline view.

If the content is changed after you close the file, the reference link value is updated automatically the next time you open the object.

Viewing Reference Links

In the **Preview** tab and window, you can view reference link values within the content of the requirement selected in the hierarchical content table. For requirements, notes, change logs, and change approval objects, the **Preview** window displays reference link values within the content of the object selected in the **Attachments**, **Links**, **Where Used**, or **Versions** tab. For more information, see [Hierarchical Content Table](#), [Attachments Tab](#), [Relations Subtab](#), [Where Used Tab](#), [Versions Tab](#), or [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.



- Content formatting is determined by the document template for the containing folder. To apply other formatting, you can change the **Document Template** property value for the folder.

For more information, see [Editing the Properties of a Selected Object](#), earlier in this chapter.



To view full content reference link values as hyperlink URLs, the document template's **Document Template Rules** property must include the **Preserve Links** value.

For information about modifying a document template, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

- The reference link value displays the text **No Access** if you do not have at least **Read Only** access privilege to a referenced object.

In the **Relations** subtab of the **Links** tab or window, you can view the reference link objects:

- For the object selected in the hierarchical content table, the subtab shows reference links to the object, reference links from the object, and the starting or ending object for each reference link.
- In the **Links** window, the **Relations** subtab can show such information for the object selected in the **Attachments**, **Where Used**, or **Versions** tab.

To view reference link values in the Preview tab or window:

Do one of the following:

- For an object in the hierarchical content table, select the object and click the **Preview** tab. You can open the **Preview** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
- For an object in the **Attachments**, **Links**, **Where Used**, or **Versions** tab:
 - With the **Preview** tab visible in the **Notebook** pane, click the **Open tab** button on the toolbar to open the **Preview** window.
 - Select the object in the **Attachments**, **Links**, **Where Used**, or **Versions** tab to display the content in the **Preview** window.

To view reference link objects in the Relations subtab of the Links tab or window:

Do one of the following:

- For an object in the hierarchical content table, select the object and click the **Links** tab, and then click the **Relations** subtab.

You can open the **Links** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.

- For an object in the **Attachments**, **Where Used**, or **Versions** tab:
 - With the **Links** tab visible in the **Notebook** pane, click the **Open tab** button on the toolbar to open the **Links** window, and then click the **Relations** subtab.
 - Select the object in the **Attachments**, **Where Used**, or **Versions** tab.



For a live Visio diagram, select the diagram image note as the target, not the diagram object itself.

For more information, see [Creating a Diagram](#) in chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

In the **Start** pane, reference links that target the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the object that contains each reference link.

In the **End** pane, reference links contained in the selected object are displayed in the link table, at the right of the pane. At the left, the object table displays the target object for each reference link.



In each pane, the object table and the link table remain synchronized at all times. For example, when you sort or scroll a link table, the corresponding object table is automatically sorted or scrolled to match the new view in the link table.

Deleting Reference Links

References can be deleted while editing text in Microsoft Word and from the Architect/Requirements client. When editing in Word, the delete actions are different for references to full text and references to property values.

- To delete the reference links between two objects:
 - . Select one of the objects in the hierarchical content table.
 - . Go to the **Links**→**Relations** tab.
 - . The **Relations** tab displays the reference links in which the object is participating. For each of the reference links to be deleted, right-click the reference link and click **Delete**.

The [**Broken Link xxx**] label appears in the objects that were using the deleted reference links, where *xxx* is the referenced text as it last appeared.

- When editing in Word, you can delete references to simple properties by selecting the range of text that is highlighted as a link and deleting those characters. When the edited text is saved, the reference link relation is deleted.
- To remove a reference to the full text of another object while editing in Word, switch to **View**→**Outline**. In the outlining toolbar, click **Remove Subdocument**. This command converts the embedded full text reference into local text. The local text can be selected and deleted, or blended into the remainder of the object's content. In either case, when the current text is saved, the reference link relation is deleted.

Chapter 10: Working With Versions

This chapter contains an overview of object versioning and instructions for working with versions and variants.

Overview of Versions and Variants

The majority of product development involves changing and improving an existing product, rather than starting from scratch. As a result, the product structures, requirements, and functions remain essentially the same as in the original design, but change slightly as the product matures. To accommodate this evolutionary process, Architect/Requirements employs versions and variants for requirements and building blocks.

Each version is an independent object, with its own relationships to other objects, including child objects and defining and complying objects. A version has all of the same child objects as the object from which it was created. Also, a version can be created from a prior version. The relationships among multiple versions of an object, from the original object, through prior versions, to the current *working* version, are illustrated by a *version tree*.

For objects with multiple versions, *effectivity* determines which version is displayed in the Architect/Requirements main window. The **Effectivity** fields provide controls for filtering versions according to *effectivity rules*, each of which specifies a category for display.

Like a version, a variant is an independent object, although it differs in its relationships. A variant is a copy without any connections to the original parent. The ROIN numbering is an additional count to the existing hierarchy. The variant has a different icon which is a requirement with a green circle on the left hand side of the object. You can have multiple variants at the same time. It's a way to create a new object that is related to an existing object without retaining the structure. A variant occupies a branch in the tree structure, with no relation to the rest of the tree.

For example, consider a requirement for miles per gallon for a compact car.

The requirement is refined to get better mileage per year. The requirement is the same but has new revisions each year. However, if you need a requirement for a midsize car, it is a different requirement and needs to be revised on its own each year. Such a requirement can be a variant of the requirement of the original compact car.

Deleting a variant does not affect the overall tree structure.

Architect/Requirements does not limit the number of versions and variants in a project. However, usability should be considered. Users may become confused if many different versions and variants of the same requirement or building block are in effect for a single product release. If a user does not see a

particular version, the cause may be that the effectivity is set so that the version is not in effect, and thus not displayed.

Enabling Versions for a Project

In the Architect/Requirements Administration module, the value for a project's **Packages** property determines whether versions are enabled. When **Version** is added to the value, users can do the following in the Systems Engineering and Requirements Management module:

- Create versions of requirements, building blocks, and their subtypes, and also create versions of prior versions of those objects. For more information, see [Creating Versions](#), later in this chapter.
- Choose effectivity rules through controls displayed in the **Address** bar. For more information, see [Filtering Versions by Effectivity Rule](#), later in this chapter.
- View a version tree in the **Versions** tab and window in the notebook pane. For more information, see [Viewing a Version Tree](#), later in this chapter, and [Versions Tab](#) and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.
- Create baselines of requirements, building blocks, and their subtypes. For more information, see [Creating a Baseline](#), later in this chapter.
- View a baseline in the hierarchical content table. For more information, see [Viewing a Baseline](#), later in this chapter.



You must have **Project Administrator** privilege for the project.

To enable versions for a project:

1. On the module bar, click the **Administration** button to gain access to the Architect/Requirements Administration module.
2. In the navigation tree, select the **Projects** node.
The content table displays all projects to which you have access.
3. Select the project in the content table.
The **Properties** tab or window displays all viewable properties for the project.
4.
In the **Value** column, double-click the value for the **Packages** property.
The Multi-Choice dialog window is displayed.
5. Check the **Version** check box, and then click **OK** to close the dialog window.
A confirmation message states that the **Version** package cannot be removed and asks if you want to continue.
6. Click **Yes**.



This action cannot be reversed. The **Version** package cannot be removed from the project.

The **Version** value is added to the **Packages** property, indicating that versions are enabled for the project.

Filtering Versions by Effectivity Rule

For objects with multiple versions, effectivity determines which version is displayed in the hierarchical content table. You choose an effectivity rule to filter the display according to a version category.

To filter versions by effectivity rule:

In the **Effectivity** list, do one of the following:

- Select **Current Frozen Version** to display the most recently frozen versions, together with non-frozen objects.
- Select **Current Version** to display the most recently created versions, whether frozen or non-frozen, together with non-frozen objects.
- Select **Frozen As Of Date** to display frozen versions from a past date to the present, together with non-frozen objects.
- Select **Version As Of Date** to display both frozen and non-frozen versions from a past date to the present, together with non-frozen objects.
- Select **Baseline** to display only the objects in a particular baseline.
- Select **Baseline with in-work** to display objects in a baseline together with non-frozen objects.

The content table displays the versions that match the selected rule. In the notebook pane, the matching version is highlighted in the **Versions** tab or window.



The **Number** property for a requirement or a building block is displayed as a - (hyphen character) if the object is not effective. The non-effective objects can be seen only in the **Versions** tab in the **Notebook** pane.

If you add the **Number** property in the **Versions** tab, the **Number** property of an effective object is displayed in a position relative to its parent. However, a non-effective object has - in its **Number** property.

Freezing Objects

To create a version of a requirement or a building block, that object must first be made static, or frozen. You can freeze one object at a time, or you can freeze multiple objects at the same time. If you select only one object, you can also freeze all of its lower level descendants in a single action. For a selected folder, you can freeze all requirements and building blocks at all levels in the folder.



- For an object to which a security profile is applied, one of the following conditions must be true:
 - Your user name must be included in either the **Full Control** or **Modify and Read Access** property value of the security profile.
 - You must have **Project Administrator** privilege for the project.
- To freeze building blocks or TRAMs, you must have **Architect** privilege for the project.

To freeze objects:

1. In the hierarchical content table, select each object that you want to freeze.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

2. Pull down the **Tools** menu, and then do one of the following:
 - To freeze only the objects in the selection, choose the **Versions**→**Freeze**→**Selected Objects** options.
 - To freeze one selected object and all of its descendants, choose the **Versions**→**Freeze**→**Deep** options.

A message asks you to confirm this action. Click **Yes** to continue.

A lock symbol appears on the object type indicator for each specified object, showing that the object is now static. The object cannot be modified, although versions of the object can be created and modified.

Procedure notes

Step 1: To see lower-level objects, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Unfreezing Objects

A static requirement or building block can be unfrozen to return the object to its editable state. You can unfreeze one object at a time, or you can unfreeze multiple objects at the same time. If you select only one frozen object, you can also unfreeze all of its lower level descendants in a single action. For a selected folder, you can unfreeze all requirements and building blocks at all levels in the folder.



- For an object to which a security profile is applied, one of the following conditions must be true:
 - o Your user name must be included in the **Full Control** property value of the security profile.
 - o You must have **Project Administrator** privilege for the project.
- To unfreeze building blocks or TRAMs, you must have **Architect** privilege for the project.
- You cannot unfreeze objects that are baselined. For more information, see [Creating a Baseline](#), later in this chapter.

To unfreeze objects:

1. In the hierarchical content table, select each object that you want to unfreeze.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

2. Pull down the **Tools** menu, and then do one of the following:
 - To unfreeze only the objects in the selection, choose the **Versions**→**UnFreeze**→**Selected Objects** options.
 - To unfreeze one selected object and all of its descendants, choose the **Versions**→**UnFreeze**→**Deep** options.

A message asks you to confirm this action. Click **Yes** to continue.

The lock symbol is removed from the object type indicator for each specified object, showing that the object is now unfrozen.

Procedure notes

Step 1: To see lower-level objects, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating Versions

You can create versions of requirements and building blocks. The new version has the same properties and values, child objects, defining and complying objects, and notes as the object from which it was created. A version of a requirement has the same text as the prior requirement.

A version's ROIN is signified by the **-2** attached to the ROIN.

Versions play a part with effectivity. Only one version is viewable at a time. Versions keep a history of the object through the life cycle of the project.



- You can only create a new version from the current version. You cannot create versions from prior versions of requirements or building blocks.
- The object must be frozen before you can create the version. For more information, see [Freezing Objects](#), earlier in this chapter.
- You must have the **Modify** permission for the object from which you intend to create the version.
- To create versions of building blocks or TRAMs, you must have the **Architect** privilege for the project.

To create versions:

1. In the hierarchical content table, select each object for which you want to create a version.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

2. Pull down the **Tools** menu and choose the **Versions**→**Create Version** options.

The content table displays the new version or versions:

- If you created one version, it is displayed with an open text field around the default name. Enter the version name in the text field, and then press the enter key.
- If you created multiple versions, you can rename each one separately. For more information, see [Renaming an Object](#) in chapter 4, *Maintaining a Project*.

Each new version is also added to the appropriate version tree in the **Versions** tab and window. For more information, see [Viewing a Version Tree](#), later in this chapter, and [Versions Tab and Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Procedure notes

Step 1: To see lower-level objects, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also press control-E. You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Version**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating Variants

You can create variants of requirements and building blocks, and of prior versions and variants of those objects. The new variant has the same properties and values, defining and complying objects, and notes as the object from which it was created. A variant of a requirement has the same text as the prior requirement. However, the new variant does not retain the base object's relationships to child objects.



- You must have **Modify** permission for the object from which you intend to create the variant.
- To create variants of building blocks or TRAMs, you must have **Architect** privilege for the project.

To create variants:

1. In the hierarchical content table, select each object for which you want to create a variant.

You can select nonadjacent objects by holding down the control key while you click the objects. To select adjoining objects, click the first object, hold down the shift key, and click the last object.

2. Pull down the **Tools** menu and choose the **Versions**→**Create Variant** options.

The content table displays the new variant or variants. You can rename each one separately. For more information, see [Renaming an Object](#) in chapter 4, *Maintaining a Project*.

Each new variant is also added to the appropriate version tree in the **Versions** tab and window. For more information, see [Viewing a Version Tree](#), later in this chapter, and [Versions Tab](#) and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

Procedure notes

Step 1: To see lower-level objects, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Step 2: You can also press control-I. You can reverse this action by pulling down the **Edit** menu and choosing **Undo New Variant**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Viewing a Version Tree

A version tree shows the relationships among multiple versions of a requirement or a building block. You view a version tree in the **Versions** tab or window. For more information, see [Versions Tab](#) and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To view a version tree:

Do one of the following:

- For an object in the content table, select the object and click the **Versions** tab, or click the object's versions indicator.
You can open the **Versions** window for this object by clicking the **Open tab** button on the notebook pane's toolbar.
- For an object in the **Links** tab:
 - With the **Versions** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Versions** window.
 - Select the object in the **Links** tab.

The **Name** column displays the versions and variants in a hierarchy, from the earliest to the latest. A plus sign is shown for each version or variant that has later versions or variants, extending the hierarchy to lower levels.

You can view the successive versions or variants by doing the following:

- To view the direct successors, click the plus sign for the object.
- To view all successors, right-click the object, and then choose **Expand All** from the pop-up menu.

In the **Versions** tab, you can also select the object, pull down the **View** menu, and then choose **Expand All**.

Procedure notes

To see lower-level objects in the content table, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Deleting a Version or a Variant

Deleting a version has a large impact on the overall structure of the version tree. A variant occupies a branch in the tree structure, with no relation to the rest of the tree. Therefore, deleting a variant does not affect the overall tree structure, but merely removes it from the tree.



- You cannot delete a frozen object or a baselined object. For more information, see [Freezing Objects](#), earlier in this chapter, and [Creating a Baseline](#), later in this chapter.
- To delete a version or variant of a building block or a TRAM, you must have **Architect** privilege for the project.

To delete a version or a variant:

Select the version or variant in the content table, and then pull down the **File** menu and choose **Delete**.

The version or variant is moved to your Architect/Requirements Recycle Bin.

Procedure notes

To see lower-level objects, click the plus signs in the **Name** column. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

You can also right-click the version or variant and choose **Delete** from the pop-up menu. Or, click the **Delete** button on the toolbar.

Creating a Baseline

A baseline allows you to fix versionable objects permanently in an unalterable state. Versionable objects are requirements, building blocks, and their system-defined and user-defined subtypes. You can baseline these objects to record their history at a given time, for example, after a milestone such as requirements approval.

The baseline process freezes and labels all versionable objects at all levels in a selected container. You can select a project to baseline all of its versionable objects simultaneously. Or, you can select one or more folders to baseline only those objects.

First in this process, unfrozen versionable objects are frozen, while objects that are already frozen are skipped. Then, each frozen object is labeled with the baseline name that you assign. This label is shown by the **Baseline** property value for each baselined object. Shortcuts and groups are ignored for the baseline.

A baseline cannot be deleted, nor can a baselined object be unfrozen. However, new versions and variants can be created from a baselined object.



Do not create a baseline if you intend for the frozen state of the objects to be reversible. Instead, you can freeze the objects without assigning them a baseline name. If the **Baseline** property value is blank, a frozen object can be unfrozen to return it to an editable state. For more information, see [Freezing Objects](#) and [Unfreezing Objects](#), earlier in this chapter.

Baselines can be renamed through the **Baseline** property definition for the project. This multiple choice property definition resides in the **Property Definitions** folder in the Administration module. For more information about modifying a choice definition, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



- To create a baseline of a building block or TRAM, you must have **Architect** privilege for the project.
- When you assign the baseline name, that value is added to the choice list for the project's **Baseline** property in the Administration module. Therefore, you must have **Project Administrator** privilege for the project in which you want to create the baseline in the Systems Engineering and Requirements Management module.
- The **Effectivity** field must first be set to **Current Version**. For more information, see [Filtering Versions by Effectivity Rule](#), earlier in this chapter.

To create a baseline:

1. Select the container whose objects you want to baseline.

In the navigation tree, you can select only one project or folder.

In the hierarchical content table, you can select one or more folders. To select nonadjacent folders, hold down the control key while you click the folders. To select adjoining folders, click the first folder, hold down the shift key, and click the last folder.

2.

Pull down the **Tools** menu and choose the **Versions**→**Create Baseline** options.

The Baseline Name dialog window is displayed.

3. Enter the baseline name in the text field, and then click **OK** or press the enter key.

A confirmation message states that this irreversible operation freezes and labels all versionable descendants of the selected containers.

4. To continue, click **Yes** or press the enter key.



- This action cannot be reversed.
- Baselined objects cannot be unfrozen.
- The baseline cannot be deleted.

In the content table, a lock symbol is shown on the object type indicator of each versionable object in the container. When one of these objects is selected, the baseline name is shown by the **Baseline** property value in the **Properties** tab or window.

When the **Effectivity** field is set to **Baseline**, the baseline name can be used to filter the view of objects in the hierarchical content table. For more information, see [Viewing a Baseline](#), later in this chapter.

Procedure notes

Step 1: To see lower-level objects, click the plus signs. Or, select an object with a plus sign, pull down the **View** menu, and choose **Expand All**. You can also right-click an object and choose **Expand All** from the pop-up menu.

Viewing a Baseline

Above the hierarchical content table, the **Effectivity** fields provide controls for filtering displayed objects according to a baseline. For the project or folder selected in the navigation tree, you can display only the objects in a baseline, or you can display baselined objects together with non-frozen objects. For more information, see [Creating a Baseline](#) and [Filtering Versions by Effectivity Rule](#), earlier in this chapter.

To view a baseline:

1.

In the **Effectivity** list, do one of the following:

- To display only objects in a particular baseline, select the **Baseline** effectivity rule.
- To display objects in a baseline and non-frozen objects, select the **Baseline with in-work** effectivity rule.

The Select a Baseline dialog window is displayed.

2. Check the check box for the baseline, and then click **OK** or press the enter key.

The content table displays each object whose **Baseline** property value includes the baseline name.

- If you selected the **Baseline** effectivity rule, all other objects are filtered from the content table.
- If you selected the **Baseline with in-work** effectivity rule, the content table also displays any non-frozen objects in the folder. These are the most recent versions.



The baseline name is displayed in the read-only field to the right of the **Effectivity** list. To view another baseline by the selected effectivity rule, you can display the **Select a Baseline** dialog window by clicking the button to the right of the read-only field.



If the referenced object is not in the same baseline as the referencing object, the reference is not effective when the effectivity is set to **Baseline**. The reference is effective only if the effectivity is set to the version of the baseline date.

Procedure Notes

Step 2: If necessary, click the plus sign (+) for the folder that contains the baselined objects. Or, select the folder, pull down the **View** menu, and choose **Expand All**. You can also right-click the folder and choose **Expand All** from the pop-up menu.

Chapter 11: Using the Search Module

This chapter contains an overview of the Teamcenter Systems Engineering and Requirements Management Search module and instructions for performing basic, intermediate, and advanced searches.

Overview of the Search Module

Through the Architect/Requirements Search module, you can quickly find objects within a project. In its concepts, functions, and appearance, the Search module is similar to the familiar Search feature of Microsoft Windows Explorer.

A search is always contained within a specified starting container, such as a project or a folder, or a hierarchical requirement or building block. Also, searches are always within a project, not across projects. The starting point for the search is determined by the way in which you activate the Search module.

You can perform basic, intermediate, and advanced searches. From any view in the Search module, you can save a search and recall it later. However, certain features, such as sorting the search results, are available only in the Advanced search view.

The Architect/Requirements Search module consists of the following views:

- The Basic search view, in which you can perform searches based on simple queries as in Microsoft Windows Explorer.
- The Intermediate search view, in which you can perform more complicated searches based on detailed criteria relating to object properties.
- The Advanced search view, in which you can perform complex searches based on queries that you construct, using statements similar to a structured query language.

In all views, the toolbar includes the **New**, **Open**, and **Save** buttons.

Activating the Search Module

The starting point for the search is determined by the way in which you activate the Search module:

- From the Systems Engineering and Requirements Management module or the Administration module, click the **Search** button on the module bar.



When the user changes projects, the **Search** button on the module bar is momentarily disabled until the project change initialization is completed.

The starting point is the object selected in the navigation tree in the Systems Engineering and Requirements Management module. However, this is not the case if the user previously activated the search module, specified a different starting point (for example, the object selected in the content table), and has not made a selection in the navigation tree since.

- From the Systems Engineering and Requirements Management module, click the **Search** button on the main window's toolbar.
 - If the navigation tree is active, the starting point is the project node or folder selected in the navigation tree.
 - If the content table is active, the starting point is the object selected in the content table.
- From the Systems Engineering and Requirements Management module, with an object selected in either the navigation tree or the content table, pull down the **Tools** menu and choose the **Search** option.
 - If the navigation tree is active, the starting point is the project node or folder selected in the navigation tree.
 - If the content table is active, the starting point is the object selected in the content table.
- From the Systems Engineering and Requirements Management module, with an object selected in either the navigation tree or the content table, right-click inside the pane and choose **Search** from the pop-up menu.

The starting point is the object selected in the right-clicked pane.

The **Look in** field displays the starting point for all methods of Search module activation.



When the starting point is the object selected in the content table, it does not change if the user activates the search module via the **Search** button on the module bar, unless the user selects an object in the navigation tree again.

Search Results



If you encounter an out of memory error when the search result is large, you may need to increase the maximum memory available for the Architect/Requirements client.

For information on increasing the available memory, see [Appendix D - Changing the maximum memory available for rich client](#).

Each view in the Search module contains the Basic pane. In this pane, you enter simple criteria such as an object name, a string of text, a ROIN of a requirement, an object type or subtype, or a specific location within the project.

Searching subordinates is incorporated in the search choices that are found in the dropdown menu. Select **Search subordinates: via Query** to perform search using the database query. Select **Search subordinates: via Relations** to perform search by traversing over the descendants of the selected object (specified in the **Look in** field). Select **Search members only** to limit search on the objects owned by the selected object.

You can also choose from two toggle choices, **Include Subtypes** and **Exclude Shortcuts**.

Additionally, the **Case Sensitive** check box is available to non-English locales, and defaults to case-insensitive search.

Also in the Basic pane, you use the **Output Options** section to choose the format in which the search results are displayed. You can choose from the following options:



For the Search Results dialog window:

- o Choose **Reuse Results Window** to output each search to the same instance of the Search Results dialog window. The window is cleared for each new search and refreshed with those results.
- o Choose **New Results Window** to output each search to a separate instance of the Search Results dialog window.

The property column settings in the Search Results dialog window are determined by the saved view that you select in the field at the bottom of the **Output Options** section. For more information, see [Search Results Dialog Window](#), later in this chapter, and [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*.



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.



Choose **Microsoft Word** to output the search results to a Word document. The document contains data for each object that matches your search criteria. The data in the document, and the Word formatting styles applied to the data, are determined by the document template that you select in the field at the bottom of the **Output Options** section. For more information, see [Exporting a Report to Microsoft Office Word](#), later in this chapter.



Microsoft Office Word 2013 or Word 2016 must be installed on your computer.

- Choose **Microsoft Excel** to output the search results to an Excel spreadsheet. The spreadsheet contains data for each object that matches your search criteria. The data in the spreadsheet is determined by the Excel template that you select in the field at the bottom of the **Output Options** section. For more information, see [Exporting a Report to Microsoft Office Excel](#), later in this chapter.



Microsoft Office Excel 2013 or Excel 2016 must be installed on your computer.

- Choose **Microsoft Visio** to output the search results to a Visio diagram. The diagram contains a shape for each object that matches your search criteria. The shapes in the diagram are determined by the Visio stencil that you select in the field at the bottom of the **Output Options** section. For more information, see [Exporting a Report to Microsoft Office Visio](#), later in this chapter.



Microsoft Office Visio 2013 or Visio 2016 must be installed on your computer.

- Check the **Keep Previous Settings** check box to save the search criteria for using it with a different project or a folder.

Search Results Dialog Window

When you choose the Search Results dialog window, each object that matches your search criteria is displayed in a table. For an object selected in the Search Results dialog window, you can do the following:

- Navigate to the object in the content table by pulling down the **View** menu and choosing **Go To Object**, or by right-clicking the object and choosing **Go To Object** from the pop-up menu.
- Edit object property values directly in the table cells. You can also edit properties in the Edit Properties dialog window, which you display by pulling down the **File** menu and choosing **Properties**, or by right-clicking the object and choosing **Properties** from the pop-up menu. For more information, see chapter 9, [Working With Object Properties](#).



For a value that exceeds the cell size, you can rest the pointer on the cell to see the entire value in a tooltip. Multiple lines are used if necessary.

- Copy or move an object to the Clipboard by pulling down the **Edit** menu and choosing **Copy** or **Cut**, or by right-clicking the object and choosing **Copy** or **Cut** from the pop-up menu. You can then paste the object in another location. For more information, see [Copying Objects](#) or [Moving Objects](#) in chapter 4, *Maintaining a Project*.
- Rename an object by pulling down the **File** menu and choosing **Rename**, or by right-clicking the object and choosing **Rename** from the pop-up menu. For more information, see [Renaming an Object](#) in chapter 4, *Maintaining a Project*.

- Delete an object by pulling down the **File** menu and choosing **Delete**, or by right-clicking the object and choosing **Delete** from the pop-up menu. For more information, see [Deleting Objects](#) in chapter 4, *Maintaining a Project*.

-

Export the Search Results to Microsoft Excel by pulling down the **File** menu and choosing the **Export**→**Excel Spreadsheet** options. You can also right-click in the table and choose the **Export**→**Excel Spreadsheet** options from the pop-up menu. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.

- Export the Search Results to Microsoft Word by pulling down the **File** menu and choosing the **Export**→**Word Document** options. You can also right-click in the table and choose the **Export**→**Word Document** options from the pop-up menu. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.
- Add and remove property columns, rearrange and resize columns, and sort by any column. For more information, see [Adding and Removing Columns](#), [Rearranging Columns](#), [Resizing Columns](#), and [Setting the Sort Column](#) in chapter 9, *Working With Object Properties*.
- Copy an object's URL to the Clipboard by pulling down the **Edit** menu and choosing **Copy URL**→**Include Full Name** or **URL Only**, or by right-clicking the object and choosing **Copy URL**→**Include Full Name** or **URL Only** from the pop-up menu. You can then paste the URL into Windows programs that support hyperlinks. For more information, see [Copying Object URLs](#) in chapter 4, *Maintaining a Project*.
- Open a folder, or open a requirement or a note for editing, by pulling down the **File** menu and choosing **Open**, or by right-clicking the object and choosing **Open** from the pop-up menu. For more information, see [Entering and Changing Requirement Content in Microsoft Office Word](#) in chapter 5, *Managing Requirements*, or [Editing a Note](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Open a requirement or a note for viewing by pulling down the **File** menu and choosing **Open Read-Only**, or by right-clicking the object and choosing **Open Read-Only** from the pop-up menu. For more information, see [Viewing Requirement Content](#) in chapter 5, *Managing Requirements*, or [Viewing Note Content](#) in chapter 8, *Recording Supplementary Information With Notes*.
- Create trace links by pulling down the **Edit** menu and choosing **Links**→ options, or by right-clicking the object and choosing **Links**→ options from the pop-up menu. For more information, see [Creating Trace Links](#) or [Linking to an Object in Another Teamcenter Product](#) in chapter 7, *Showing Object Relationships With Trace Links*.

Saving, Recalling, and Running Searches and Reports

From the Basic, Intermediate, or Advanced view in the Search module, you can save a search and recall it later. For example, you can save a search that you frequently perform, and recall it to use the same criteria for convenience. Or, you can open a saved search and modify its criteria, and then output those results or save your changes as a different search. A search that you recall is automatically displayed in the same view from which it was saved.

Saved search reports are specific to a project. Searches are stored as schema objects in the project's **Reports and Formatting** folder in the Administration module. If you have questions about search reports, consult your project administrator.



You can copy the URLs of these objects to run reports independently of the Architect/Requirements client. In the content table, right-click a search object and choose **Copy URL to Run** from the pop-up menu.

Then, paste the URL into a Windows program that supports hyperlinks, such as Microsoft Outlook. By clicking the URL hyperlink in an E-mail message, for example, a recipient can run the report without running the client. The user must log in with an Architect/Requirements user name and password.

The **Shared State** property is assigned to each new search. This property value determines the report's availability among the users in the project:

- With the **Private** value, the report can be recalled only by its creator, and only that user can change the report. **Private** is the default value for each new report.
- With the **Public** value, the report can be recalled by all users in the project. The report can be changed only by a user with **Project Administrator** privilege for the project. A project administrator can also delete a public report. In addition, a project administrator can assign the **Public** value to a private report.
- With the **Pending** value, a private report is marked as a request for the project administrator to make the report public. The report's creator can assign the **Pending** value to mark the request. Then, the project administrator can assign the **Public** value to make the report available to the project's other users.

Private, public, and pending reports can be exported from the Administration module as can other schema objects. However, only public reports can be imported. For more information about exporting and importing project data, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

To save search criteria:

1.
On the Search module toolbar, click the **Save** button to display the Saved Report Information dialog window.
2. Enter a unique name in the **Report Name** field.
You can also do the following:
 - Enter additional text in the **Brief Report Description** field.
 - Check the **Use same starting point every time** check box to begin at the currently selected point each time you recall the search.
3. Click **OK** or press the enter key.

To recall search criteria:

1.
On the Search module toolbar, click the **Open** button to display the Select Saved Report dialog window.
2. Check the check box for the search, and then click **OK** or press the enter key.
The Basic, Intermediate, or Advanced view displays the search criteria.

To run a report:

1.
In the Search module or the Systems Engineering and Requirements Management module, pull down the **Tools** menu and choose **Run Report** to display the Select Report dialog window.
2. Check the check box for the saved search criteria to use for the report, and then click **OK** or press the enter key.



By default, the starting point for the report is shown in the **Look in** field. However, a different starting point may be specified in the saved search.

A message states that the request is submitted. Depending on server response time, a progress indicator is displayed. Above the indicator, the message changes to show the current stage of the process.

You can continue working in the Architect/Requirements client during this process. When the message closes, the objects that match the search criteria are displayed in the specified output format.



- If you click **Cancel**, the process runs until the server reaches the next checkpoint.
- Your Architect/Requirements system administrator can set the server polling interval through the Web Application Configuration administrative tool. For more information about administrative tools, see the *Systems Architect/Requirements Management System Administrator's Manual*.

Exporting a Report to Microsoft Office Excel

For criteria that you enter in any Search module view, you can export search results directly to a Microsoft Office Excel workbook. As with workbooks exported from other views of the Architect/Requirements client, the report contains property values for objects that meet your search criteria. For more information, see [Exporting Objects to Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.

You can export the search results to a workbook that is not connected to the Architect/Requirements database. Or, you can use the report to create a new live Excel workbook for editing property values interactively. For more information, see [Using the Live Excel Interface](#) in chapter 9, *Working With Object Properties*.

To specify the properties, you can base the report on an Excel template, which contains property columns, tags for property values, and other information. Or, you can base the report on a saved view of property columns. For more information about saved views, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*. For more information about Excel templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about Excel templates or views, consult your project administrator.



Microsoft Office Excel 2013 or Excel 2016 must be installed on your computer.

To export a report to Microsoft Office Excel:

1. Enter your complete search criteria in the Basic Search view, Intermediate Search view, or the Advanced Search view.

For more information, see [Basic Search View](#), [Intermediate Search View](#), or [Advanced Search View](#), later in this chapter.

2. Under **Output Options**, select **Microsoft Excel**.



You can check the **Live?** check box to export the report to a live Excel workbook.

At the bottom of **Output Options**, the list shows the Excel templates and the public views for the project, including the private and pending views for your user name.

3. In the list, select the Excel template or the saved view.



- The Excel template may contain multiple worksheets, and each worksheet may specify different properties and rules. Therefore, data may be separated on worksheets in the export file.
- If the Excel template or the view contains properties that do not apply to an object type specified for export, the inapplicable values are indicated in the related cells.



For a public report, Siemens PLM Software recommends that you select a public view. If the report is imported later with other project data, pending and private views will not be available because they cannot be imported. For more information, see [Customizing Views of Property Columns](#) in chapter 9, *Working With Object Properties*. If you have

questions about reports, views, or importing project data, consult your system administrator.

4. Click the **Search** button.

A message states that the request is submitted. Depending on server response time, a progress indicator is displayed. Above the indicator, the message changes to show the current stage of the process. You can continue working in the Architect/Requirements client during this process.



- If you click **Cancel**, the process runs until the server reaches the next checkpoint.
- Your Architect/Requirements system administrator can set the server polling interval through the Web Application Configuration administrative tool. For more information about administrative tools, see the *Systems Architect/Requirements Management System Administrator's Manual*.

When the message closes, the report opens in a read-only Excel workbook (**.xlsm**). In this temporary file, you can view and print the data and send it by E-mail and fax. This file is deleted from your computer when you exit Architect/Requirements.

You can create a permanent workbook from this report by clicking Excel's **Office Button** and choosing **Save As** to display the **Save As** dialog window. A permanent workbook can serve as record for comparison with future changes, or as a means of editing properties through live Excel. Also, the data in the workbook can be imported to create new objects in any Architect/Requirements project. For more information, see [Importing Objects From Microsoft Office Excel](#) in chapter 4, *Maintaining a Project*.

Procedure Notes

Step 4: If you checked the **Live?** check box in step 2, and if the live Office interface is not installed, a message asks if you want to install live Office. Click **OK** to start the installation. When the installation is complete, repeat steps 2 through 4.

Exporting a Report to Microsoft Office Word

For criteria that you enter in any Search module view, you can export search results directly to a Microsoft Office Word document. As with Word documents generated from other views of the Architect/Requirements client, the report contains data such as property values for objects that meet your search criteria. For more information, see [Exporting Objects to Microsoft Office Word](#) in chapter 4, *Maintaining a Project*.

You can also choose the document template on which you want to base the report. For more information about document templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about document templates, consult your project administrator.



Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

To export a report to Microsoft Office Word:

1. Enter your complete search criteria in the Basic Search view, Intermediate Search view, or the Advanced Search view.

For more information, see [Basic Search View](#), [Intermediate Search View](#), or [Advanced Search View](#), later in this chapter.

2. Under **Output Options**, select **Microsoft Word**.

The field at the bottom of the **Output Options** section displays the default document template for this report. You can choose a different document template in this field.

3. Click the **Search** button.

A message states that the request is submitted. Depending on server response time, a progress indicator is displayed. Above the indicator, the message changes to show the current stage of the process. You can continue working in the Architect/Requirements client during this process.



- If you click **Cancel**, the process runs until the server reaches the next checkpoint.
- Your Architect/Requirements system administrator can set the server polling interval through the Web Application Configuration administrative tool. For more information about administrative tools, see the *Systems Architect/Requirements Management System Administrator's Manual*.

When the message closes, the report opens in a read-only Word document (**.html**). In this temporary file, you can view and print the data and send it by E-mail and fax. This file is deleted from your computer when you exit Architect/Requirements.

You can create a permanent document from this report by clicking Word's **Office Button** and choosing **Save As** to display the **Save As** dialog window. A permanent document can serve as record for comparison with future changes. Also, the data in the document can be imported to create new requirements in any Architect/Requirements project. For more information, see [Creating Requirements and Content by Import From Microsoft Office Word](#) in chapter 5, *Managing Requirements*.

Exporting a Report to Microsoft Office Visio

You can export search results to a Microsoft Office Visio diagram from any Search module view. The diagram contains shapes and other data for objects that meet your search criteria. You can choose the Visio stencil on which you want to base the report. For more information, see [Basic Search View](#), [Intermediate Search View](#), or [Advanced Search View](#), later in this chapter. For more information about Visio stencils, see the *Systems Architect/Requirements Management Project Administrator's Manual*. If you have questions about stencils, consult your project administrator.



- Microsoft Office 2013 or a later version and Office Live must be installed. The Office Live version must match your Architect/Requirements version. For more information, see [Using the Live Office Interface](#) in chapter 3, *Using the Architect/Requirements Main Window*.
- The report is exported to a static diagram, not to a live Visio diagram.

To export a report to Microsoft Office Visio:

1. Enter your search criteria.



- If the criteria include qualifying relationships, for example, with **ADD** statements in the Advanced Search view, the diagram displays results in a multiple level tree. With this structure, you can easily see how system elements work together. For more information, see [FOR EACH and ADD Statements](#), later in this chapter.
- If you enter criteria in the Basic Search view or the Intermediate Search view, all results are displayed at the same level.

2. Under **Output Options**, select **Microsoft Visio**.
3. In the field at the bottom of **Output Options**, select **Default Static Tree Stencil** or a custom static tree stencil for your project.
4. Click the **Search** button.

A message states that the request is submitted. Depending on server response time, a progress indicator is displayed. Above the indicator, the message changes to show the current stage of the process. You can continue working in the Architect/Requirements client during this process.



- If you click **Cancel**, the process runs until the server reaches the next checkpoint.
- Your Architect/Requirements system administrator can set the server polling interval through the Web Application Configuration administrative tool. For more information about administrative tools, see the *Systems Architect/Requirements Management System Administrator's Manual*.

When the message closes, Visio opens a read-only diagram, which you can view, print, and send by E-mail. This file is deleted from your computer when you exit Architect/Requirements.

In a diagram that contains a tree, the superior level shows the objects that match the search criteria, and subordinate levels show the objects that fulfill the qualifying relationships, which are indicated by colored lines. Initially, the tree is displayed vertically with the superior level at the top.



The color for each relationship category is set in the mapping file (.xml) associated with the stencil.

- To change the vertical or horizontal orientation of the tree, right-click a blank area on the Visio page and choose **Display Left to Right / Top to Bottom** from the pop-up menu.
- To hide or show one or more relationships in the entire diagram:



Right-click a blank area on the Visio page and choose **Toggle Relationship** from the pop-up menu.

The Toggle Relationship dialog window is displayed. Initially, the checked relationships are those that are specified in the search criteria.

- Do one or both of the following:
 - Clear the check box for each relationship that you want to hide.
 - Check the check box for each relationship that you want to show.



Relationships not specified in the search criteria are not shown even if you check the check boxes.

By checking **Redraw Tree in Visio**, you can reset the layout of the shapes when you apply your changes. The current layout is retained when this check box is cleared.

- Click **Apply** to see your changes and keep the dialog window open, or click **OK** to apply your changes and close the dialog window.



The Toggle Relationships dialog window does not apply to a diagram that displays all search results at the same level.

- To hide or show all relationships for one object, right-click the shape and choose **Show / Hide Subordinates** from the pop-up menu.



This option is available only in a diagram that displays search results in a tree.

- To view an object's properties, right-click the shape and choose **Teamcenter for systems engineering properties** from the pop-up menu.

- To navigate to an object in the Architect/Requirements client:
 - Right-click the shape and choose **Go To Teamcenter for systems engineering** from the pop-up menu to display the login page.
 - Enter your user name and password, and then click **Login** to see the object highlighted in the client.

You can save the diagram locally with Visio's **Save As** dialog window. When you reopen the file, however, you cannot show or hide relationships or subordinates, nor can you change the orientation.

Procedure Notes

Step 4: If the Office Live interface is not installed, a message asks if you want to install Office Live. Click **OK** to start the installation. When the installation is complete, repeat steps 2 through 4.

Order of Search Results

The search functionality is enhanced for improving the query performance. As a consequence, the order of search results in some cases may be different from the order in the previous Architect/Requirements releases.

Subtype Order: When the search includes multiple subtypes of a base type, the search results include a mix of objects of all subtypes. The mixed order is because the search performs a single query for better performance.

To get the results clustered by subtype, choose **Advanced** search type and add a **SORT** with Subtype clause to the search query. If the search includes only one subtype or when the search is on a base type, the subtypes are automatically clustered in the search results.

View Sort Order: When search results are displayed in a View, the view sort order is used to order the search results if the search query does not contain the **SORT** clause. By default, the search results are ordered by the view's sort column, except when the search query includes a **SORT** clause.



- For any search that does not include a **SORT** clause, the view's sort order may be different than the natural search order. For example, the natural order of the result with a **FOR EACH** clause is the **Number** order. This may be different from the sort order specified by the selected view. In this case, the view sort order is used to order the search results.
- In case of output to Word or output to Excel, the search results are ordered by the natural search order. For example, the natural order of the result with a **FOR EACH** clause is the **Number** order.

Basic Search View

When you first activate the Search module, the Basic search view is displayed by default. This view contains only the Basic search pane. You use this pane to search for objects that meet basic criteria, such as an object name, a string of text, a ROIN of a requirement, an object type or subtype, or a specific location within the project.

The search starts at the object that is selected in the Systems Engineering and Requirements Management module when the Search module is activated. The target and search criteria that you specify determine the search results that you obtain. Selecting an appropriate starting point and the specific object types of interest yields search results more quickly than does a broad search for all object types across an entire project.

At the top of the Basic search view, the **Look in** field displays the container that is the starting point for the search. You can change the starting point by selecting a different container in this field. For more information, see [Activating the Search Module](#), earlier in this chapter.

To the right of the **Look in** field, the **Keep Same Location** check box is cleared by default, allowing you to automatically set the starting location to your last selection in the Systems Engineering and Requirements Management module. You can check the **Keep Same Location** check box to switch to the Systems Engineering and Requirements Management module while maintaining the current starting point in the Search module.



Selecting your Architect/Requirements Recycle Bin in the Systems Engineering and Requirements Management module does not change the starting point in the **Look in** field.

The Basic search view also contains the following:

- **Look in** field
- **Search For Name** field
- **Containing Text** field
- **Matching ROIN** field
- **Types/SubTypes** tree
- **Search** button
- **Search Type** section
- **Search Options** section
- **Output Options** section
- **Keep Same Location** check box

Figure 11-1 shows the Basic search view in the Search module.

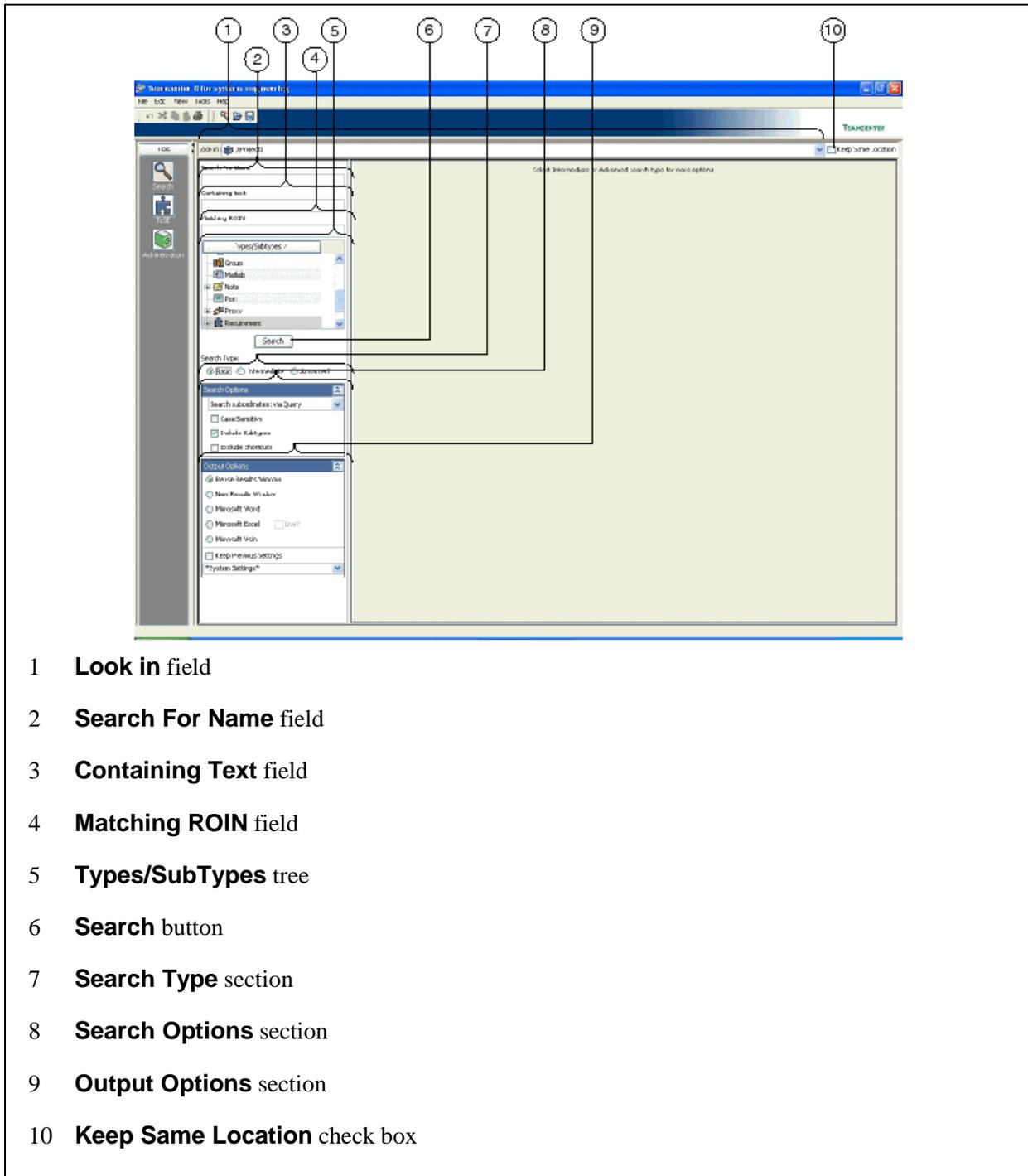


Figure 11-1. Basic Search View in Architect/Requirements Search Module

To use the Basic search view:

1. To specify the search criteria, do one or both of the following:
 - To limit the search to objects that have a specific name, contain specific text, or ROIN, do one or both of the following:

o

In the **Search For Name** field, enter any portion of the name that you want to find.



If you leave this field empty, the search extends to all objects specified in the **Types/SubTypes** tree.

o

In the **Containing text** field, enter any portion of the text that you want to find.

o

In the **Matching ROIN** field, enter any portion of the ROIN of the requirement that you want to find.

These fields are not case sensitive. In all the three fields, you can use wildcard characters to substitute for unknown characters or to cover several possible conditions. Wildcard characters are the asterisk (*) and the question mark (?). For example:

- o The characters **a*b** locate the strings **ab**, **axb**, and **axyzb**.
- o The characters **a?b** locate the strings **axb**, **ayb**, and **azb**.
- o The characters **00*** locate the ROINs **001**, **0002**, and **00345**.



o

In the **Search For Name** field and the **Containing text** field, you do not need to enter wildcard characters for a relative search, which is performed automatically.

o

In the **Matching ROIN** field, enter wildcard characters for a relative search.

If you want to search for text that actually contains an asterisk (*), a question mark (?), an open bracket ([) or closed bracket (]), or a backslash (\), you must precede such characters with a backslash (\).



You can start the search at this point by placing the cursor in either field and pressing the enter key.

●

To extend or limit the search to specific object types or subtypes, do one or both of the following in the **Types/SubTypes** tree:

- o To include an object type or subtype in the search, select the type or subtype in the tree.

You can select multiple types or subtypes by holding down the control key while you click the items. To select adjoining items, click the first item, hold down the shift key, and click the last item.

- o To exclude a currently selected object type or subtype from the search, clear the type or subtype from the tree.

You can clear multiple types or subtypes by holding down the control key while you click the items.



In the **Search Options** section, do one or more of the following:

- o

Select **Search subordinates: via Query** from the dropdown box to perform search using the database query.

- o

Select **Search subordinates: via Relations** from the dropdown box to perform search by traversing over the descendents of the selected object (specified in the **Look in** field).

- o

Select **Search members only** from the dropdown box to limit search on the objects owned by the selected object.

- o

Check the **Case Sensitive** check box to limit the search to the case entered in the text fields.

- o

Clear the **Include Subtypes** check box to exclude subtypes of types selected in the **Types/SubTypes** tree.

- o

Check the **Exclude Shortcuts** check box to exclude shortcuts in the search result.



In advanced searches, the **Exclude Shortcuts** toggle only applies to the initial **SELECT** statement. To remove shortcuts from a **FOREACH ADD**, insert a where clause, toggle **Tcl Script**, and choose the **Not Shortcut** activator.



If shortcuts are included, there is a difference in the behavior between searches depending on relations and searches depending on query. When the search is done depending on relations, both direct shortcuts as displayed with black shortcut icon overlay and shortcut descendents as displayed with red shortcut icon overlay are included in the result. When search is done depending on query, only direct shortcuts are included in the result.



The **Search subordinates: via Query** and **Search subordinates: via Relations** selections do not affect what is included in the search result. They only affect the search performance.

When a project or a high level folder (with a large number of descendents) is selected, the search performance is better with the **Search subordinates: via Query** selection. When a lower level folder (with a relatively small number of descendents) is selected, the search performance is better with the **Search subordinates: via Relations** selection.

For related information on improving the search performance using database indexes, see the *Systems Architect/Requirements Management Server Installation Manual*.



In the **Output Options** section, select the format for the search results. For more information, see [Search Results](#), earlier in this chapter.

2. Click the **Search** button to display the search results in the format selected in the **Output Options** section.

You can clear the current criteria and start a new search by clicking the **New** button.

Intermediate Search View

The Intermediate search view allows you to select properties for specified object types and refine the search by specifying criteria for any of the properties. After activating the Search module, you display the Intermediate search view by clicking the **Intermediate** button in the **Search Type** section in the Basic search pane. For more information, see [Activating the Search Module](#) and [Basic Search View](#), earlier in this chapter.

At the top of the Intermediate search view, the **Look in** field displays the container that is the starting point for the search. You can change the starting point by selecting a different container in this field. For more information, see [Activating the Search Module](#), earlier in this chapter.

To the right of the **Look in** field, the **Keep Same Location** check box is cleared by default, allowing you to automatically set the starting location to your last selection in the Systems Engineering and Requirements Management module. You can check the **Keep Same Location** check box to switch to the Systems Engineering and Requirements Management module while maintaining the current starting point in the Search module.



Selecting your Architect/Requirements Recycle Bin in the Systems Engineering and Requirements Management module does not change the starting point in the **Look in** field.

The Intermediate search view also contains the following:

- The **Select Properties** pane, which lists all available properties for the selected object types in the basic **Search** pane.
- The **Specify Criteria** pane, in which you can specify the selection criteria for an object property. Initially, this pane displays criteria based on the property selected in the **Select Properties** pane. The criteria change according to the type of property that you select. For example, the pane displays different criteria for numeric, date, text, and choice properties.
The **Specify Criteria** pane also displays criteria for the row that you select in the **Edit Search Criteria** pane. You can remove specified criteria by clicking **Delete Row**.
- The **Edit Search Criteria** pane displays each set of criteria specified in the **Specify Criteria** pane.
You can select a row in the **Edit Search Criteria** pane to change that criteria in the **Specify Criteria** pane.

Figure 11-2 shows the Intermediate search view in the Search module.

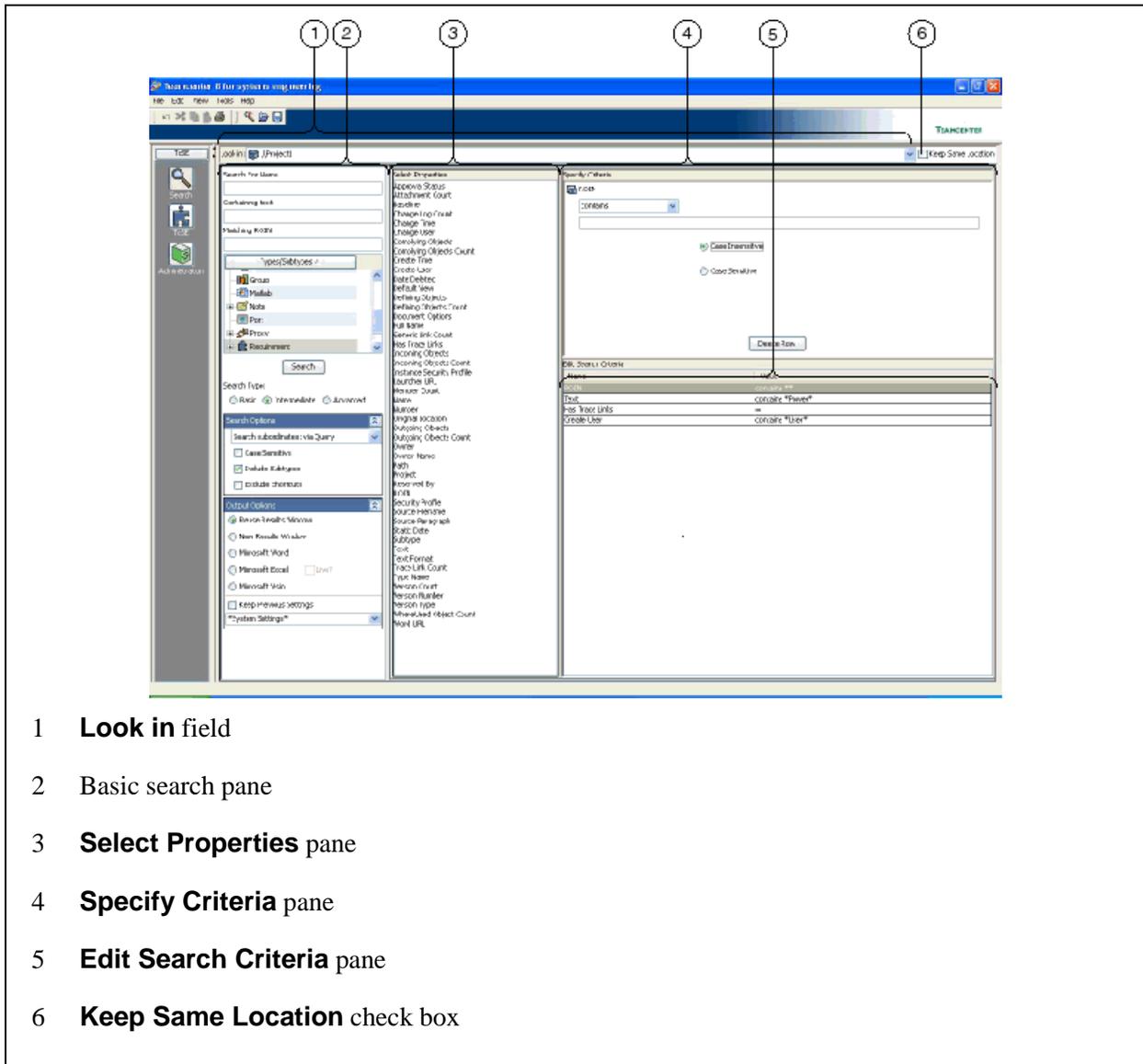


Figure 11-2. Intermediate Search View in Architect/Requirements Search Module

To use the Intermediate search view:

1. In the **Search Type** section of the Basic pane, click the **Intermediate** button.

The **Select Properties** pane displays all available properties for the object types selected in the **Types/SubTypes** tree in the Basic pane.

- 2.

Select a property in the **Select Properties** pane to display the criteria for the property type in the **Specify Criteria** pane.



The selected property does not remain highlighted, allowing you to select that same property a second time. There are times when you may want to add the same property to a search more than once. For example, you may want to search for all change users whose ID starts with *t* but is not equal to *test*. You can accomplish this by including the **Change User** property twice, once for each rule.

- 3.

Click any field in the **Specify Criteria** pane to automatically add criteria to the **Edit Search Criteria** pane.

You can remove criteria from the **Edit Search Criteria** pane by selecting the criteria and pressing the **Delete Row** button.

Depending on the type of property selected in the **Select Properties** pane, one of the following is displayed in the **Specify Criteria** pane:

- The text criteria panel allows you to specify search criteria for text properties. You can specify text that exactly matches the string you enter, and you can specify partial text matches. Also, you can specify text that does not equal or does not contain certain criteria.

You can choose either the **Case Insensitive** or **Case Sensitive** option for any text criteria. Also, you can enter regular expressions directly in a text field.

- The date criteria panel allows you to specify search criteria for date properties. It also shows the format of that property. You can search for dates that occurred in the last or next number of days, weeks, months, or years. Or, you can indicate a specific date and search for items whose property matches that date, that does not equal that date, occurs before that date, or occurs after that date. The final option allows you to specify a date range.

This panel includes a calendar widget. Clicking the calendar icon displays the widget, in which you can select a date in a calendar dialog window. Clicking the **OK** button accepts the selected date. Clicking **Cancel** or pressing the escape key closes the dialog window without changing the selected date in the text field.



You can search for objects that have a property whose value is **TBD**, which marks the value for entry at a later date. You can also exclude these objects from the search.

- o To search for objects with a **TBD** property value, select **TBD** in the **Specify Criteria** pane and choose **is** from the list to the right.

- o To exclude objects with a **TBD** property value, select **TBD** in the **Specify Criteria** pane and choose **is not** from the list to the right.

- The numeric criteria panel allows you to specify search criteria for numeric properties. It also shows the format of that property. You can search for numbers that using any one of several available operands, including *equals*, *not equals*, *less than*, *less than or equal to*, *greater than*, and *greater than or equal to*. You can also indicate a range that the number must fall within.
- The choice criteria panel allows you to specify search criteria for properties with choices. A list displays the choices that you can select. The operands you can use are *equals*, *any*, *not equals*, and *not any*.

The *equals* operand is inclusive and means that you will only find items that match each of the choices you selected. The *any* operand is exclusive and means that you will find items that match any of the choices you selected.

4. To change specified criteria, select the appropriate row in the **Edit Search Criteria** table, and then change the criteria in the **Specify Criteria** pane.
5. To begin the search, click the **Search** button.



- To begin the search without pressing the **Search** button, press the enter key when the cursor is inside a text field.



When you select a property in the **Select Properties** pane, it does not stay selected. Therefore, you can immediately select that same property again and specify additional criteria using the same field.

- Criteria are automatically added to the search when you click any field in a criteria pane.
- To remove criteria from the search, click the **Delete Row** button while editing that criteria in the **Specify Criteria** pane.



Switching from Basic or Advanced search to Intermediate search does not carry over the search criteria. You must enter the criteria again to replicate the search.

Advanced Search View

Although the Advanced search view shares some components with the other views in the Search module, the Advanced view has a fundamental difference. In the Basic and Intermediate views, you can search for objects using limited criteria, and it is not necessary to understand the concept of a query object that is constructed behind the scenes. In the Advanced search view, however, you see and directly construct a query. If you have used a structured query language to query a database, then you are already familiar with this concept.

You can create complex queries through the Advanced search view because it includes additional statements and properties not available in the other two search modes. The specific statements that are available will be discussed later. The Basic and Intermediate views limit the number of properties to a standard, predefined set of the most commonly used system properties, and those user interfaces are less complicated. In the Advanced search view, all properties for the selected object types are available.

At the top of the Advanced search view, the **Look in** field displays the container that is the starting point for the search. You can change the starting point by selecting a different container in this field. For more information, see [Activating the Search Module](#), earlier in this chapter.

To the right of the **Look in** field, the **Keep Same Location** check box is cleared by default, allowing you to automatically set the starting location to your last selection in the Systems Engineering and Requirements Management module. You can check the **Keep Same Location** check box to switch to the Systems Engineering and Requirements Management module while maintaining the current starting point in the Search module.



Selecting your Architect/Requirements Recycle Bin in the Systems Engineering and Requirements Management module does not change the starting point in the **Look in** field.

The Advanced search view also contains the following:

- Basic pane
- **Query View** pane
- **Modify** pane
- **Query Edit** pane
- **Properties** pane
- **Specify Criteria** pane

Figure 11-3 shows the Advanced search view in the Search module.

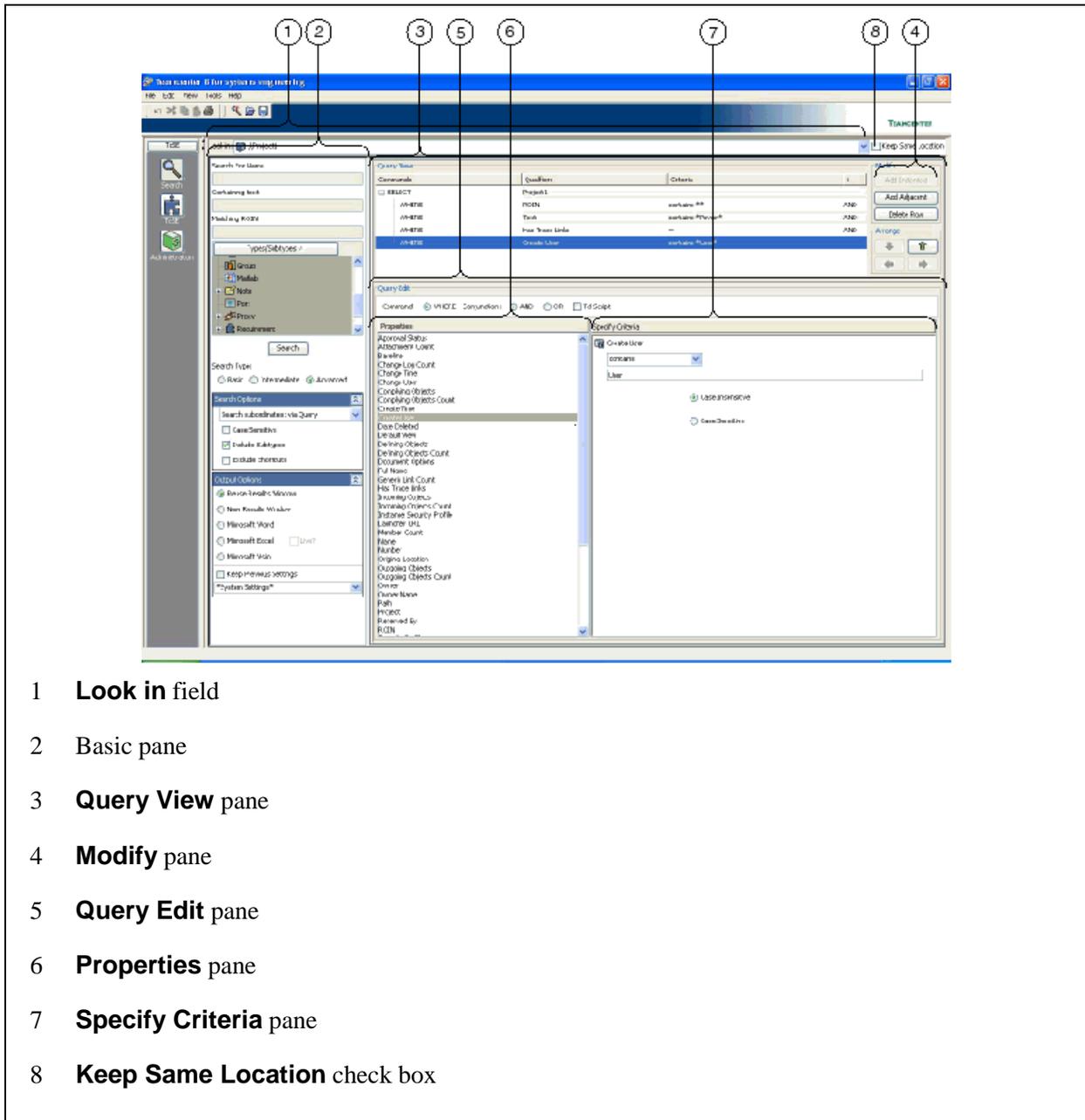


Figure 11-3. Advanced Search View in Architect/Requirements Search Module

Switching to the Advanced Search View

When you first start Architect/Requirements and switch to the Search module, it defaults to the Basic search. You can change the search type to the Intermediate search, which extends the Basic view. Regardless of which of the two search modes you are in, switching to the Advanced Query mode always takes any search criteria you already entered to build the initial query.



Because switching to the advanced mode always builds a new query from whichever search mode you were already in, you should be very careful about switching from the Advanced Query mode down to a lower level search if you want to keep your query information. If you construct a query in the Advanced Query mode, switch to the Basic or Intermediate search, and then switch back to Advanced, your previous advanced query will be replaced by the new one that is built from the criteria you had entered in the lower level search. If you want to keep your advanced query but switch to a lower level query for a new search, it is best to first save your query using the **Save** button.

Basic Pane

The Basic pane of the Advanced search view is the same pane used in the Basic and Intermediate views, with two exceptions:



The **Search for Name** and **Containing text** fields are disabled because any values entered there are displayed as **WHERE** clauses in the query tree view. Although they are disabled, they remain visible so that the other components in the Basic search pane do not move when you switch to the Advanced view.



The **Types/Subtypes** tree is automatically enabled and disabled as appropriate for the type of statement you are editing in the query. Query statements are qualified by either object types or by properties. If a statement uses object types, then the **Types/Subtypes** tree is enabled while you edit that statement. If a statement use properties, then the **Types/Subtypes** tree is disabled, and the **Properties** pane is enabled.

Query View Pane

The **Query View** pane displays the query as it is being constructed. However, this pane is more than a read-only view of the query. It is a hierarchical tree table where rows can be selected, added, updated, and deleted. The query tree table view is the major component that drives the creation and updating of a query in the Advanced search view.

The **Query View** pane contains three columns, **Commands**, **Qualifiers**, and **Criteria**. Depending on the row that is selected, what goes in the columns can vary slightly, but basically you can read the query top-down, left to right.

Modify Pane

The **Modify** pane contains buttons through which you modify the statements contained in the **Query View** pane. Each button is used to change specific rows in the query and is automatically enabled and disabled as appropriate to the selected row. To see a description of a button in a tooltip, you can rest the pointer on the button.

Because a query is a hierarchy, indention is important to understanding the query and its execution. The **Add Indented** button adds a subordinate statement (a child node) indented under the current row. For example, because a **SELECT** statement is the root node for a query, the **Add Indented** button is the only button that is enabled when you select the first row of the query tree. When you click this button, you see a list of the statement types that are valid as child nodes of the selected statement. You use the **Query Edit** pane to specify what statement you want to add. The **Add Indented** button is disabled when you click on a statement that cannot have child nodes, such as a **WHERE** clause.

The **Add Adjacent** button adds a statement, or sibling, after the current row.

The **Delete Row** button deletes the current row and any of its child nodes. You can select multiple rows in the query tree table by clicking a row and then either using shift-click to select a range of rows or control-click to select multiple, nonadjacent rows. If you select multiple rows, the button name changes to **Delete Rows**, and all selected rows and their children are deleted.

You can use the arrow buttons under **Arrange** to change the structure of the query without changing individual statements. All button actions move the selected row and any of its child nodes along with it.

- The down arrow button moves the current row and any of its children down below its next sibling.
- The up arrow button moves the current row and any of its children above its previous sibling.
- The left arrow button promotes a statement up one level to make it a sibling of its current parent.
- The right arrow button demotes a statement down one level to make it a child node of its previous sibling.



Each button is enabled or disabled as is generally appropriate but not every possible combination is covered. It is possible to construct a query that makes no sense or does not execute. If the query does not return results, verify that the query is properly constructed. If an error message is returned, a specific problem may be indicated.

Query Edit Pane

The **Query Edit** pane displays the statements that relate to the row that you select in the **Query View** pane. When you add a new row, the valid types of commands that can be added for the selected row in the query tree table are shown to allow you to choose which type of command you want.

When you choose a statement type, the user interface changes to present you with the options that are valid for that type of statement. Selecting a row in the query tree table allows you to edit an existing statement. When you do this, the command area shows what type of command it is, but it cannot be changed. For more information, see [Statement Types](#), later in this chapter.

The **Properties** pane is for specifying properties for those statements that use properties, such as **WHERE** and **SORT**. Because this is the Advanced Query, all possible properties for the selected object types are listed, unlike the Basic and Intermediate views that have a more limited set of properties.

The **Specify Criteria** pane is for specifying criteria for a given property. When you select an item from the **Properties** pane, the **Specify Criteria** pane displays the appropriate controls for that property type. For example, if you select a date property such as **Create Time**, the **Specify Criteria** pane displays controls that allow you to specify criteria such as a specific date or a date range.

Constructing Advanced Queries

Although the Advanced search view limits your choices to those options that may be valid at any given time, it does not prevent you from creating a query that is illogical or cannot be executed. Queries allow greater flexibility, but a thorough understanding of their construction is required. Therefore, consider the following rules in constructing queries.

- When you switch to the Advanced view from the Basic or Intermediate view, any search information that is already entered from the previous view is used to build an initial query.
- A query is represented by a hierarchical tree structure.
- Each query starts with a **SELECT** statement.
- There are two kinds of statements, those that are qualified by selecting object types, and those that are qualified by specifying property criteria.
- Once a statement is added, you cannot change its statement type.
- The user interface does not require that siblings be in a certain order but the server will process them in a specific order. For more information, see [Sibling Order](#), later in this chapter.
- All **WHERE** clauses are implicitly joined together with an **AND** conjunction. You can specify an explicit **OR** condition between multiple **WHERE** clauses.

Hierarchical Structure

The first thing to consider about a query is that it is hierarchical. The first statement is always a **SELECT** statement, and every other statement below it is subordinate to it. The user interface indicates these levels of a hierarchy by using the indentation of a standard tree table component. This is the same type of tree interface used in many other views in Architect/Requirements, such as the content table of the Systems Engineering and Requirements Management module.

Each item in the tree is referred to as a node. The top level node is called the root node. Nodes are containers that can have ownership of other nodes. A node that contains other nodes is called a parent node. Parent nodes can be collapsed and expanded. A node that is contained within a higher level node is called a child node. Nodes at the same level and with the same parent are called siblings. A node that does not have any children is called a leaf node.



A query is represented by a hierarchical tree table. In order to understand how to properly construct a query and use the interface, it is essential that you understand the concepts of an outline tree structure and how indentation affects the scope of a statement.

Because the query is a hierarchy, it should come as no surprise that operations for a row affect its children. For example, deleting a parent row will also delete all its child nodes. As you would expect, moving a parent row will move its children along with it. And, promote and demote (moving a row left and right in the hierarchy) adjusts the hierarchical level one position for the parent and all its children.

Statement Qualifiers

Each line of a query is a statement that is used to define what you are searching for. Statements have qualifiers and criteria that narrow their scope. Statements are qualified using either an object type or by specifying property criteria. For example, in a **SELECT** statement, you can specify which type of objects you want to iterate over. And, in a **WHERE** clause you can narrow down those objects even further by specify criteria for one or more properties.

It is not necessary to know whether a statement requires object types or properties because the user interface enables the appropriate panel and disable the inappropriate one for that type of statement.

Statement Types

When you select a line in the query tree table and press a button to add a statement to a query, the user interface gives you a choice of valid statements for that point in the query. Once you choose what type of statement you want, for example, **WHERE**, **SORT**, or **REMOVE**, you cannot change it. You can change the qualifiers and criteria for that statement, but you cannot change its statement type.

The reason for this is because each statement has certain rules about where they can be used (what type of clause is their parent) and what must be specified as qualifiers, criteria, and valid subordinate clauses. Changing a type could invalidate the whole query and cause an error. These types of problems are avoided and the user interface is simplified by not allowing statement types to be changed.



Statement types cannot be changed because it could invalidate the query structure. Deleting a query row and adding a new one accomplishes the same thing and allows the user interface to verify that the new statement conforms to the query language syntax.

Sibling Order

In an Architect/Requirements query, the order of siblings is usually important to its functionality. Although the user interface does not require you to put siblings in a specific order, it helps to understand the order that is used to process sibling statements. The server processes query statements in the following order:

1. **SELECT** statement
2. **WHERE** clauses
3. **SORT** clauses
4. All other subcommands

Sometimes the order of siblings is unimportant. For example, if a **SELECT** statement contains multiple **WHERE** clauses and a **SORT** clause, it does not matter which of the three clauses come first because the server still processes them in the order stated above.

In an Architect/Requirements query, the order of siblings is usually unimportant. For example, the following two queries are identical in their results because the two **WHERE** clauses and the one **FOR EACH** clause are all at the same level.

```
SELECT Requirements
  WHERE Name like "Require"
  WHERE Text like "shall"
  FOR EACH Paragraph
    Add Notes
  SORT by NAME

SELECT Requirements
  FOR EACH Paragraph
    Add Notes
  SORT by NAME
  WHERE Text like "shall"
  WHERE Name like "Require"
```

But, most of the time the sibling order does matter. If you have multiple **SORT** clauses at the same level, then the first one indicates the primary sort, the second one indicates the secondary sort, and so on. It does not matter whether the **SORT** clause appears before or after any **WHERE** clauses that are its siblings. For more information, see [SORT Clause](#), later in this chapter.

Also, a **FOR EACH** statement requires that the first child under it be an **ADD** clause or a **WHERE** clause. So, in this case, the order of siblings is important because the **ADD** and the **WHERE** clauses must appear before any other clauses under a **FOR EACH** statement. It may help if you think of the **FOR EACH** and the **ADD** clauses as two parts of one statement.

Additionally, if you have multiple **FOR EACH** statements at the same level, the server will process the first one and its result will be processed by any subsequent **FOR EACH** clauses. Likewise, if you have multiple **REMOVE** clauses that are siblings, the server processes them in the order that you've indicated. Some objects may be removed from the first **REMOVE** statement before it gets to any subsequent statements.

Advanced Query Statements

This section discusses the query statements used in the Advanced search view.

SELECT Statement

Each query starts with a **SELECT** statement and there is only one **SELECT** statement in a query. You cannot create and join multiple **SELECT** statements together. When you enter the Advanced view from either the Basic or Intermediate view, the initial **SELECT** statement is built based on the object types that are currently selected in the **Types/SubTypes** tree of the Basic pane.



You can use **WHERE** clauses to narrow the scope of any **SELECT** statement. For more information, see [WHERE Clause](#), later in this chapter.

A **SELECT** statement has an optional parameter which is indicated in the **Query Edit** pane with a **Use Starting Object** check box. If you check this check box, the object identified in the **Look in** field is used as a starting location. For example, instead of selecting all requirements, the first statement could select a specific requirement as a starting point. Then all subsequent statements, such as a **FOR EACH** and an **ADD** statement, would operate on that object.



The **Look in** field at the top of all the search panels may change depending on how the search module is invoked.

When **Use Starting Object** is selected, the qualifier of the **SELECT** statement in the query tree table changes from one or more objects types to the name of the specific object that you've chosen as the starting point. The only time this might be confusing is if the object name you're starting with is similar to an object type. For example, if you have a requirement object named **Requirement**, then it wouldn't be clear by just looking at the **SELECT** statement row whether you were searching for requirement object types or the specific object named **Requirement**.

Using meaningful names for objects such as folders and requirements is a good practice. It also avoids confusion when searching for object types versus searching with a given object as the starting point.

WHERE Clause

A **WHERE** clause narrows a search and is subordinate to another statement, such as a **SELECT** or a **FOR EACH** statement. In multiple **WHERE** clauses, the conjunctions **AND** and **OR** are explicit and are used in the **Query View** pane as follows:

- *Conjunction statements* are used to group a set of **WHERE** clauses for evaluation, specifying their precedence in the same way as parentheses specify the order of evaluation for mathematical equations. You can specify **AND** and **OR** conjunction statements as qualifying statements. Each conjunction statement must follow a **WHERE** clause and must have at least one subordinate **WHERE** clause. To specify a conjunction statement:
 - For a child of another statement, select the parent, click **Add Indented** in the **Modify** pane, and then click **AND** or **OR** in the **Query Edit** pane.
 - For a sibling of another statement, select the sibling, click **Add Adjacent** in the **Modify** pane, and then click **AND** or **OR** in the **Query Edit** pane.

A subordinate **WHERE** clause is automatically added below the new conjunction statement. You then complete the blank **WHERE** clause. For example:

```
SELECT Requirement
  WHERE Status = "closed"
  AND
    WHERE Version = "5.1.1"
    WHERE Version = "5.1.2"
  OR
    WHERE Create Date > 1/1/04
```

- *Ending conjunctions* are used in **WHERE** clauses, to control how an individual clause is evaluated with the result of its next sibling **WHERE** clause:
 - An **AND** conjunction specifies a condition in which both that **WHERE** clause *and* its next sibling must be true.
 - An **OR** conjunction specifies a condition in which either that **WHERE** clause *or* its next sibling may be true.

The **AND** conjunction ends all **WHERE** clauses by default. Therefore, the **AND** button is selected automatically in the **Query Edit** pane when you add a new **WHERE** clause. To specify an **OR** conjunction, select the **WHERE** clause and click the **OR** button. For example:

```
SELECT Requirement
  WHERE Assigned To = "Frank" AND
  WHERE Status = "closed" AND
  WHERE Version = "5.1.1" OR
  WHERE Version = "5.1.2"
```

For multiple sibling **WHERE** clauses with mixed conjunctions, **AND** conditions are evaluated first. Then **OR** conditions are evaluated. For the last clause in a series, an ending conjunction has no effect and is hidden in the **Query View** pane.

Multichoice fields allow for **OR** conditions because you can specify the operators **ANY** or **NOT ANY**. Using the **ANY** operator, for example, you can specify the following:

```
WHERE Subtype is Any Document or Folder
```

FOR EACH and ADD Statements

A **FOR EACH** statement iterates over the result of the parent statement that it is subordinate to in order to add more objects to the result set. As stated earlier, the first child under a **FOR EACH** statement should be an **ADD** clause.



Because the query language structure requires that a **FOR EACH** statement have an **ADD** clause, it may help to think of the **FOR EACH** statement and the **ADD** clause under it as just one statement written on two lines.

When you create a **FOR EACH** statement, you can choose an object type as a qualifier, but it is not necessary to do so. For example, you may want to iterate over all requirement objects within a folder and add notes for all paragraph objects. Because paragraphs are subtypes of requirements, the start of your query might look something like this:

```
SELECT Requirement
  FOR EACH Paragraph
    Add Notes
```



You can use **WHERE** clauses to narrow the scope of any **FOR EACH** or **ADD** statement. For more information, see [WHERE Clause](#), earlier in this chapter.

Unlike most statements, the qualifier is optional in a **FOR EACH** statement. Omitting a qualifier is equivalent to specifying all of the object types that you have selected in the result set up to that point. If no qualifier is specified, the **FOR EACH** statement iterates over those object types.

Each **ADD** clause must specify a relationship. When you create a new **ADD** clause or select an **ADD** row in the query edit table, the **Query Edit** pane in the middle of the right panel contains a box to specify the relationship between the objects that result from the **FOR EACH** statement and what kind of objects you want to add to that result. For example, you can add all complying objects, defining objects, diagrams, and notes.

An **ADD** clause also contains an optional parameter to specify whether you want its scope to be deep or not. This box is also in the **Query Edit** pane. If the deep box is not checked, only objects at the first level under the **FOR EACH** result set are added to the result. If the deep option is used, objects are added from each level below the starting result set.

```
SELECT Requirement Specification
  FOR EACH
    ADD DEEP Members
      WHERE Release = "2.0"
```

You can use multiple **ADD** clauses with a single **FOR EACH** statement. This allows showing more than one relationship at a particular level in the result. When multiple **ADD** clauses include objects at the same level in the hierarchy, indented under the same parent object, the results are displayed in the same order as the **ADD** clauses. You can rearrange adjacent **ADD** clauses using the up and down arrows to get the desired order in the result. Non-adjacent **ADD** clauses can also include objects at the same hierarchy level.

You can not rearrange the non-adjacent **ADD** clauses. If you need to rearrange non-adjacent **ADD** clauses, use the **Move To Top** check box, to control the result order. Selecting **Move To Top** for an **ADD** clause causes the result objects to be inserted before previous objects at the same hierarchy level.



TOP is appended to the **ADD** command in the query view when **Move To Top** is selected.

Table 11-1 describes the options in the **Relationship** field in the **Query Edit** pane.

Table 11-1. Options in Relationship Field for ADD Statements

Option	Description
Child Diagrams	Specifies all child diagram objects in a parent diagram to which the superior FOR EACH statement applies.
Complying Link List	Specifies complying trace links for the objects to which the superior FOR EACH statement applies.
Complying Objects	Specifies objects that comply with the objects to which the superior FOR EACH statement applies.
Connection List	Specifies objects that are connected to the objects to which the superior FOR EACH statement applies. Both starting and ending connections may be output.
Data Definitions	Specifies all data definitions in the data dictionary to which the superior FOR EACH statement applies.
Defining Link List	Specifies defining trace links for the objects to which the superior FOR EACH statement applies.
Defining Objects	Specifies objects that define the objects to which the superior FOR EACH statement applies.
Diagrams	Specifies diagrams attached to the objects to which the superior FOR EACH statement applies.
From Connections	Specifies connections that start at the objects to which the superior FOR EACH statement applies.
Incoming Link List	Specifies incoming generic links for the objects to which the superior FOR EACH statement applies.
Incoming Objects	Specifies objects that have incoming generic links to which the superior FOR EACH statement applies.
Next Version	Returns the next version plus any variants to which the superior FOR EACH statement applies. Use the deep option to get all later versions.
MatLAB Objects	Specifies all MatLAB objects to which the superior FOR EACH statement applies.
Members	Specifies objects owned by the objects to which the superior FOR EACH statement applies.

Table 11-1. Options in Relationship Field for ADD Statements

Option	Description
Notes	Specifies notes attached to the objects to which the superior FOR EACH statement applies.
Outgoing Link List	Specifies outgoing generic links for the objects to which the superior FOR EACH statement applies.
Outgoing Objects	Specifies objects that have outgoing generic links to which the superior FOR EACH statement applies.
Owner	Specifies the object that owns each object to which the superior FOR EACH statement applies.
Ports	Specifies all port objects that are included in the Visio diagram to which the superior FOR EACH statement applies.
Prior Version	Returns the immediately prior version to which the superior FOR EACH statement applies. Use the deep option to get all prior versions.
Referencing	Specifies objects that are pointed to by reference links in the objects to which the superior FOR EACH statement applies.
Spreadsheet List	Specifies spreadsheets attached to the objects to which the superior FOR EACH statement applies.
To Connections	Specifies connections that end at the objects to which the superior FOR EACH statement applies.
Uses	Specifies objects referenced by the objects to which the superior FOR EACH statement applies.
Version List	Returns all versions and variants for an object (including itself) to which the superior FOR EACH statement applies.
Where Referenced	Specifies objects that contain reference links pointing to the objects to which the superior FOR EACH statement applies.
Where Used	Specifies the groups, diagrams, and remote proxies that reference the objects to which the superior FOR EACH statement applies.

Examples of FOR EACH and ADD Statements

Following are examples for show trace relationships.

The **Defining Objects** and **Complying Objects** keywords are used to follow trace link relationships. The deep option allows you to recursively follow those relationships.

```
SELECT Requirement
  FOR EACH
    ADD DEEP Complying Objects
```

To include the trace link objects in the result, use the **Defining Link List** and **Complying Link List** relationships . Following example shows the defining object and the connecting trace link.

```
SELECT Requirement
  FOR EACH
    ADD Defining Link List
  FOR EACH
    ADD Defining Objects
```

You can use multiple **ADD** statements with a **FOR EACH** to show multiple relationships. For example, to see all the trace relationships for a requirement, use the following statement:

```
SELECT Requirement
  FOR EACH
    ADD Complying Objects
    ADD Defining Objects
```

Different icons are used in the search result window to distinguish the relationship represented. Multiple non-adjacent **ADD** statements can include results at same hierarchy level. By default, the **Members** are displayed before sibling **Defining Objects** because the **ADD Members** clause comes first. Selecting **Move To Top** in the **ADD Defining Objects** clause reverses the order and puts **Defining Objects** before the sibling members. For example:

```
SELECT MyDocument
  FOR EACH
    ADD DEEP Members
  FOR EACH
    ADD TOP Defining Objects
```

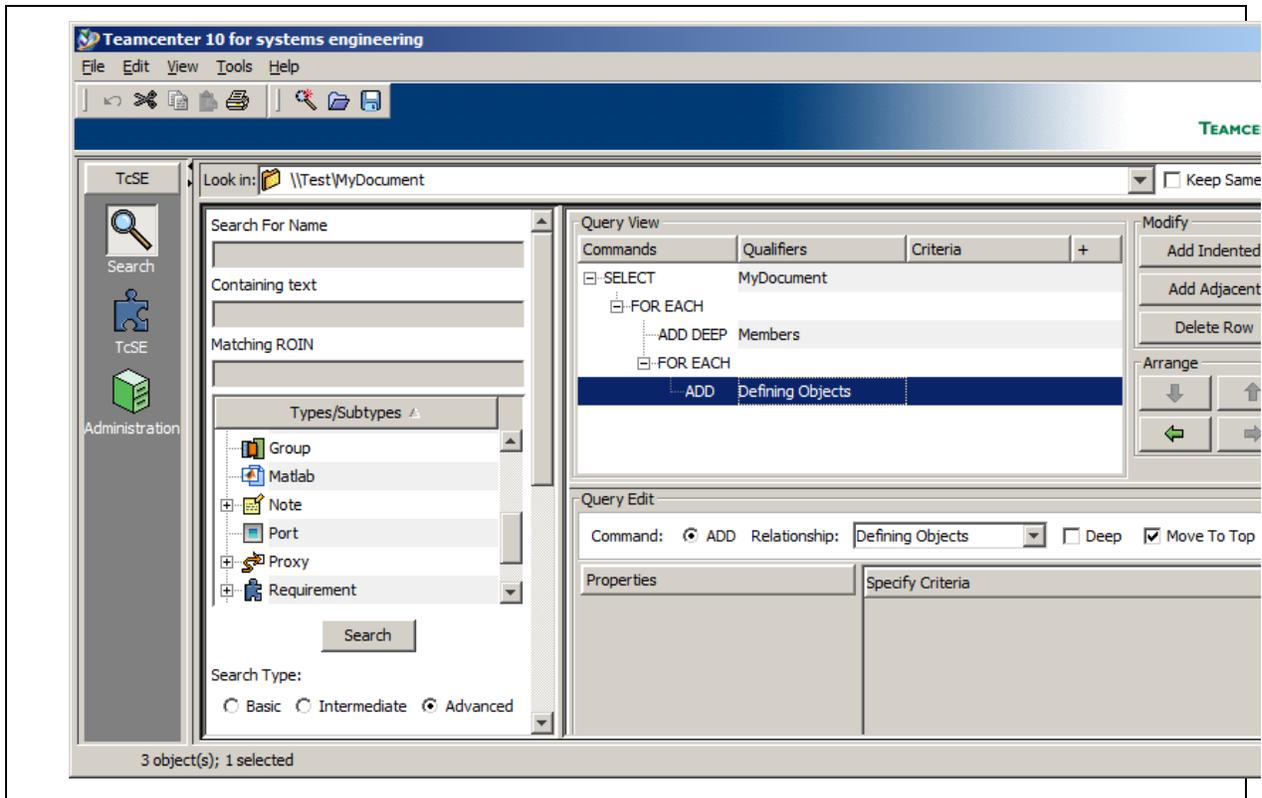


Figure 11-4. Example of the Advanced Search with Move to Top selected

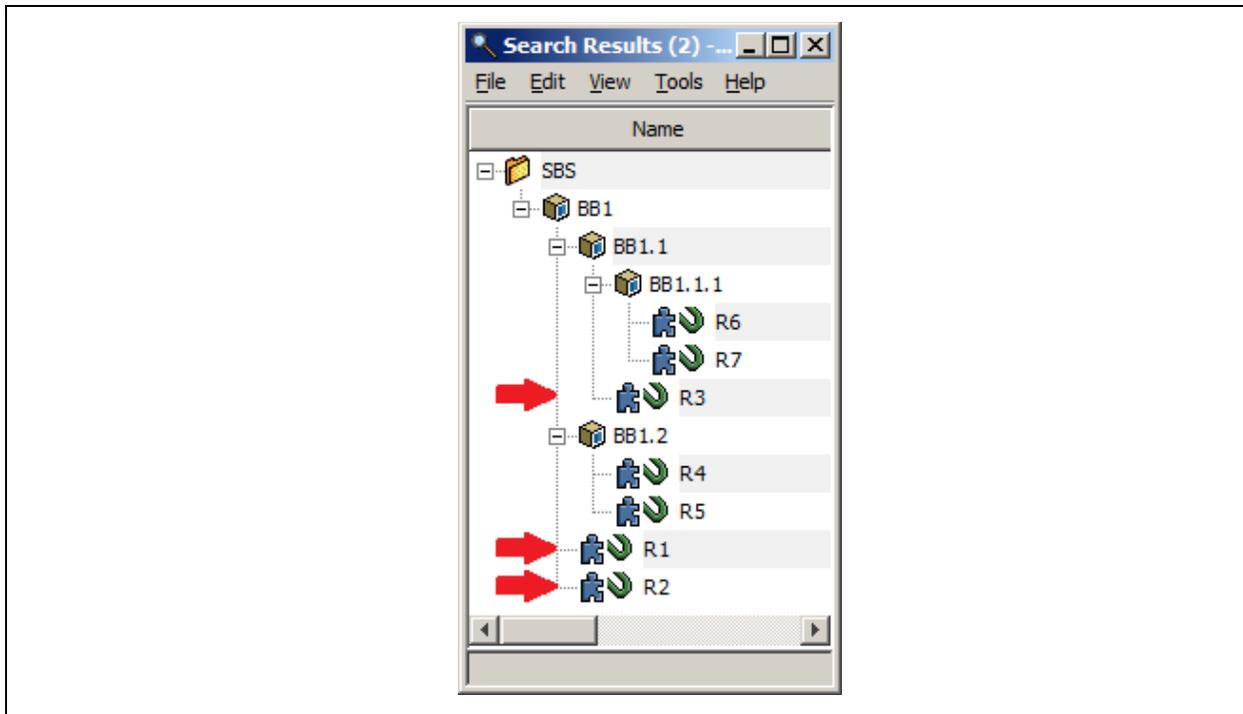


Figure 11-5. Output of the Search when Move To Top is not selected

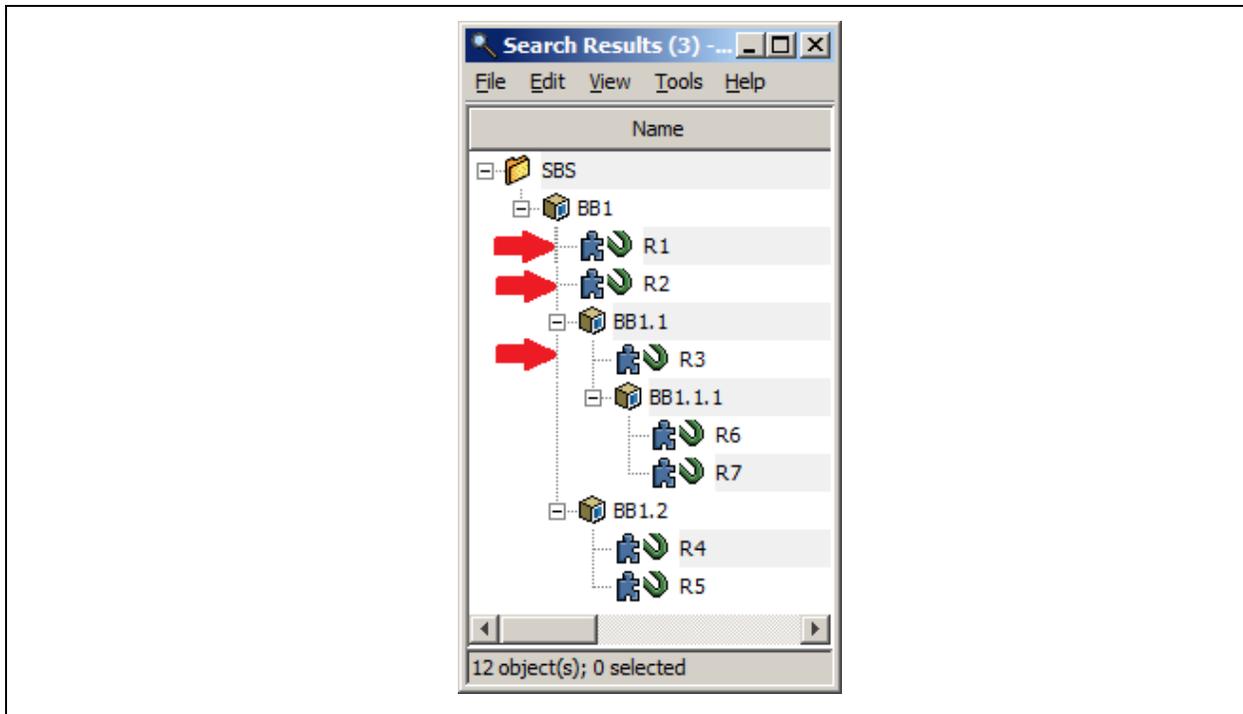


Figure 11-6. Output of the Search after Move To Top is selected

You can search for Trace Links and show the related objects using the following search statement:

```
SELECT Trace Link
FOR EACH
    ADD Complying Objects
    ADD Defining Objects
```

REMOVE Clause

A **REMOVE** clause is used to remove items from the query result. In some queries, you may have to temporarily add objects to the result to get to other objects. **REMOVE** can then get rid of the temporary objects. For example you may want to find all the design elements that may be affected by a requirement change. If the requirement is not directly linked to the design elements, but is indirectly linked through derived requirements, then the derived requirements would have to be added to the result in order to get to the design elements. The derived requirements could then be removed to simplify the result.

```
SELECT Customer Requirement
  FOR EACH
    ADD ComplyingDerived Requirement
  FOR EACH
    ADD Building Block
  REMOVE Derived Requirement
```

A **REMOVE** clause can only be used directly under a **SELECT** or **FOR EACH** statement (and after the **ADD** statement that must be the first child node under a **FOR EACH**). A **REMOVE** statement is qualified by which type of objects you want to remove and can be further defined by adding **WHERE** clauses subordinate to the **REMOVE** clause. In fact, a **WHERE** clause is the only type of child node allowed under a **REMOVE** clause.

SORT Clause

A **SORT** clause is used to arrange the query results in a specific order. Unlike some statements, the order of **SORT** clauses is important. The first **SORT** clause specifies the primary sort, and each sibling **SORT** clause specifies subsequent sort criteria. For example, if the first **SORT** clause is to sort by **Create User** and the next adjacent **SORT** clause is **Create Time**, you see the results sorted primarily by the user that created the object, and within objects created by the same user the results are sorted by create date and time.

Each **SORT** clause is qualified by the property you want to sort on and whether you want to sort ascending or descending. The default is ascending.



The **SORT** clause applies to the **SELECT** and **FOR EACH, ADD** commands that add objects to the search result. Further sorting of objects that are already in the result, such as after a **REMOVE** command, is not supported.

Whenever the Architect/Requirements server sorts the data due to a **SORT** clause in the query, the Architect/Requirements client removes the sorted column indicator (up or down arrow on the sorted column's header). The absence of the sorted column indicator shows the data was sorted on the server and was not altered by the client, regardless of which **Named View** was selected. This feature allows you to use a view to specify the properties you want to display and override the saved sort order (saved view) with the advanced query statements. This is important because an advanced query allows for multiple **SORT** clauses so that there can be a primary sort order followed by subsequent secondary sorts. A **Named View** does not allow for secondary sorts.

Chapter 12: Using the Change Management Package

This chapter provides an overview of the Architect/Requirements change management package and contains instructions for submitting, approving, rejecting, and viewing changes.

Overview of Change Management

Requirements and building blocks can be submitted through the Architect/Requirements change management package for change approval. A submitted object may be either of the following:

- An object with no prior versions is a new submission. When the changes are submitted, the object is frozen for the first time.
- An object with prior versions is a revision to previously submitted content. When the changes are submitted, this revision is frozen.



To use the change management package, the **Version** package must be enabled for the project. For more information, see [Enabling Versions for a Project](#) in chapter 10, *Working With Versions*.

The change management package tracks the following events:

- - A **Submit** event occurs when a requirement or a building block is submitted for approval. The package sends E-mail to recipients in the following categories:
 - *Approvers* have the responsibility to approve or reject the changes. Each of these users receives an E-mail message containing two hyperlinks:
 - A hyperlink to the object in the Architect/Requirements client. The user can click this hyperlink to navigate to the object and review the changes.
 - A hyperlink to the change management package. The user can click this hyperlink to navigate to the approval and rejection page in Microsoft Internet Explorer.

o

Notifiers do not approve or reject but are notified of submitted changes for informational purposes. Each of these users receives an E-mail message containing a hyperlink to the object in the Teamcenter Systems Engineering and Requirements Management client. The user can click the hyperlink to navigate to the object and review the changes.

Each message also contains the submitter's comments, if any. In addition, each approver and notifier receives the object's content in **.mhtml** format as an E-mail attachment. For an object that has prior versions, the previously submitted content is included.

The **Submit** event automatically creates a *change approval object* for the submitted object. Displayed in the **Attachments** tab or window, the change approval object contains a table of information related to the approval process. The initial table row is added for the **Submit** event.

•

A **Response** event occurs each time an approval or a rejection is received.

For each response, a row is added to the change approval object that is attached to the submitted object. The change management package also tracks delinquent responses.

•

An **Approved** event occurs if the changes are approved by all approvers. The approved object remains frozen. An E-mail message is sent to each approver and to the submitter, stating that the changes are approved.

The **Approved** event concludes the approval process, and a final row is added to the change approval object.

•

A **Rejected** event occurs each time the changes are rejected by one approver. When the first **Rejected** event occurs, the rejected object is unfrozen and remains in that state.

For each **Rejected** event:

- o An E-mail message is sent to the submitter only, stating that the changes are rejected by the individual approver.
- o A row is added to the change approval object for this event.

If other **Response** events are received after the first rejection, each later event is added to the change approval object.



All aspects of the change management package can be customized through the Teamcenter Systems Engineering and Requirements Management application programming interface (API). For more information, see the *Systems Architect/Requirements Management API Reference*.

Submitting Changes for Approval

When you submit changes for a requirement or a building block, the change management package sends an E-mail message to the approvers and notifiers. These users must be identified in a security profile, and that security profile must be applied to the requirement or building block. If you have questions about

security profiles, consult your project administrator. For more information about security profiles, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



Each approver automatically has full access to the changes until that user approves or rejects the change request.

You can apply the security profile through the object's **Security Profile** property in the **Properties** tab or window.



- You must have **Modify** permission for the object.
- To submit a building block or a TRAM, you must have **Architect** privilege for the project.

To submit changes for approval:

1. In the hierarchical content table, select the requirement or building block that contains the changes.
- 2.

Pull down the **Tools** menu and choose the **Versions**→**Submit For Approval** options.

The Submit For Approval dialog window is displayed. The **Approvals**, **Notify**, and **Object(s)** fields are read-only, displaying information from the applicable security profile. The following fields are optional:

- In the **Subject** field, you can overwrite the default text with a description of your particular request.
 - In the **Message** field, you can enter plain text for the body of the E-mail message.
3. To close the dialog window and send the request, click **OK**.

A confirmation message states that the request for approval is sent, and the E-mail message is sent to the approvers and notifiers.

If the object is not frozen, a lock symbol appears on the object type indicator in the hierarchical content table. If the changes are approved, the object remains frozen. If the changes are rejected by one approver, the object is unfrozen.

In the **Attachments** tab or window, a change approval object is automatically created for the submitted object. The change approval object contains a table in which the columns display information related to the approval process. Initially, the table contains only a row for your submission. A new row is added for each **Response** event that occurs. If the changes are approved, a final row is added for that event. For more information, see [Overview of Change Management](#), earlier in this chapter, and [Modifying a Change Approval Object](#) and [Viewing a Change Approval Object](#), later in this chapter.

Approving or Rejecting a Change

If you are listed as a change approver for a requirement or a building block, you receive an E-mail message requesting your approval when a change to the object is submitted. You gain access to the change through Microsoft Internet Explorer. You have that access until you approve or reject the request. For more information, see [Submitting Changes for Approval](#), earlier in this chapter.

In the browser, you can review the content of the proposed change with or without running the Architect/Requirements client. Also, you can view the current information for the change approval object that is attached to the object submitted for approval. For more information, see [Viewing a Change Approval Object](#), later in this chapter.

To approve or reject a change:

1.

To gain access to the change in Internet Explorer, do either of the following:

- In the browser, do the following:

- . Open the Architect/Requirements home page, and then click the **Change Approval** hyperlink.



If the Architect/Requirements client is not running, either the Architect/Requirements log in page or the Teamcenter Login page is displayed. Enter your Architect/Requirements or Security Services user name and password, select a language, and click **Log in**.

The browser displays the Changes Waiting for Approval page for your user name.

- . In the **Select from available projects** field, select the project that contains the change.

The page displays a table of changes waiting for your approval.

- . In the row that contains the change, click the **Change Status** button.

The browser displays the Approve / Reject page for the change.

- In the E-mail change approval request, click the hyperlink indicated to approve or reject and provide comments.

The browser displays the Approve / Reject page for the change.

On the Approve / Reject page, you can do the following:

- Click the **Proposed Change** button to open the change content in an **.mhtml** file for review.
- Click the **View Comments** button to open a new browser window containing the current information in the change approval object.
- Enter plain text comments in the **Fill in your comments to approve/reject the change** field.

2. Do one of the following:

- To approve the change, click the **Approved** button.

The browser displays your Changes Waiting for Approval page with the change removed from the table. In the client, a **Response** event indicating your approval is added to the change approval object, which can be viewed in the **Attachments** tab or window.



If yours is the last outstanding approval, an **Approved** event is added to the change approval object. The change approval process is concluded, and the approved object remains frozen.

An E-mail message is sent to each approver and to the submitter, stating that the change is approved by all approvers. Two **.mhtml** files are attached to this message. One attachment contains the approved content, and the other contains the table from the change approval object. The recipient can click the link in the message to navigate to the approved object in the client.

- To reject the change, click the **Rejected** button.

The browser displays your Changes Waiting for Approval page with the change removed from the table. In the client, a **Response** event indicating your rejection is added to the change approval object, which can be viewed in the **Attachments** tab or window.

An E-mail message is sent to the submitter, stating that the change is rejected by your user name. Two **.mhtml** files are attached to this message. One attachment contains the rejected content, and the other contains the table from the change approval object. The submitter can click the link in the message to navigate to the rejected object in the client.



If yours is the first rejection, the rejected object is unfrozen and remains in that state.

Modifying a Change Approval Object

In the **Attachments** tab, a change approval object is automatically created for a requirement or a building block that is submitted for approval. The change approval object contains a table in which the rows show all responses that are currently received. Each row shows the time of response, the responding user name, the user's comments, and the response type.

If you are a change approver for a submitted requirement or building block, you can rename the change approval object, and you can edit the object's content in Microsoft Word as you edit requirement content.



To edit the content of the change approval object, Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

To modify a change approval object:

1. Do one of the following:

- For a requirement or a building block in the hierarchical content table, select the object, and then click the **Attachments** tab to display the change approval object. You can

open the **Attachments** window for this requirement or building block by clicking the **Open tab** button on the notebook pane's toolbar.

- For a requirement or a building block in the **Links** or **Versions** tab:
 - With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
 - In the **Links** or **Versions** tab, select the requirement or building block to display its change approval object in the **Attachments** window.
- 2. In the **Attachments** tab or window, select the change approval object, and then do one or both of the following:
 - To rename the object, pull down the **File** menu and choose **Rename**, enter the new name in the open text field, and press the enter key.

You can also right-click the object and choose **Rename** from the pop-up menu, or press the F2 key.
 - To edit the content:
 - Pull down the **File** menu and choose **Open**.

You can also right-click the object and choose **Open** from the pop-up menu, or double-click the object.

The change approval object opens in Word as an **.mhtml** file, in which you can enter and change content within the table of responses and anywhere outside the table. This file is temporary and is deleted from your computer when you exit Architect/Requirements.
 - To save the content in the database, click Word's **Office Button** and select **Save**.

Viewing a Change Approval Object

The **Attachments** tab and floating window display the change approval object for the requirement or building block selected in the hierarchical content table. For the requirement or building block selected in the **Links** or **Versions** tab, the change approval object is displayed in the **Attachments** window.

For the change approval object selected in the **Attachments** tab or window, you can view the content in two ways:

- In a read-only Microsoft Word file, where you can view and print the content and send it to E-mail and fax recipients. You must have **Project Administrator** privilege to change the content in the database.



Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

- In the **Preview** window, where you can view and print the content without opening the change approval object in Word. You cannot change the content in the database. For more information, see [Preview Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To view a change approval object in Microsoft Word:

1. Do one of the following:
 - For a requirement or a building block in the hierarchical content table, select the object, and then click the **Attachments** tab to display the change approval object for the selected requirement or building block. You can open the **Attachments** window for this requirement or building block by clicking the **Open tab** button on the notebook pane's toolbar.
 - For a requirement or a building block in the **Links** or **Versions** tab:
 - . With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
 - . In the **Links** or **Versions** tab, select the requirement or building block to display its change approval object in the **Attachments** window.
2. In the **Attachments** tab or window, select the change approval object, and then pull down the **File** menu and choose **Open Read-Only**.

You can also right-click the change approval object and choose **Open Read-Only** from the pop-up menu.

The change approval object opens in Word as an **.mhtml** file. This file is deleted from your computer when you exit Architect/Requirements.

To view a change approval object in the Preview window:

For a requirement or a building block in the hierarchical content table:

1. Select the requirement or building block, and then click the **Attachments** tab to display the change approval object.
You can open the **Attachments** window for this requirement or building block by clicking the **Open tab** button on the notebook pane's toolbar.
2. With the **Preview** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Preview** window.
3. In the **Attachments** tab or window, select the change approval object to display its content in the **Preview** window.

For a requirement or a building block in the **Links** or **Versions** tab:

1. With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
2. In the **Links** or **Versions** tab, select the requirement or building block to display its change approval object in the **Attachments** window.
3. With the **Preview** tab on top, click the **Open tab** button to open the **Preview** window.
4. In the **Attachments** window, select the change approval object to display its content in the **Preview** window.

Viewing a Change Log

A change log captures the history of modifications to an object of a given type definition. An object's change log contains a table in which the rows show all change events that are currently recorded. Each row shows the date and time of change, the user who made the change, and the type of change. For changes to object properties, the log also shows the previous and new values.



The previous and new values of requirement and note text are not shown because their text size and rich content can exceed the constraints of the tabular layout of the change log.

You can also compare the content of the requirement. For more information, see [Comparing the Content of Different Requirements](#).

Change logs may apply to folders, requirements, building blocks, groups, notes, trace links, and connections. For each type definition, your project administrator can specify change events that are tracked by a change log. The change log is automatically created for an object of that type in the Systems Engineering and Requirements Management module when a specified change event occurs. A row is added to the change log for each later change event for that object.

The **Attachments** tab and floating window display the change log for the object selected in the hierarchical content table. For the object selected in the **Links** or **Versions** tab, the change log is displayed in the **Attachments** window.

For the change log selected in the **Attachments** tab or window, you can view the content in two ways:

- In a read-only Microsoft Word file, where you can view and print the content and send it to E-mail and fax recipients. You must have **Project Administrator** privilege to change the content in the database.



Microsoft Office Word 2013 or Microsoft Office Word 2016 must be installed on your computer.

- In the **Preview** window, where you can view and print the content without opening the change log in Word. You cannot change the content in the database. For more information, see [Preview Tab](#) in chapter 3, *Using the Architect/Requirements Main Window*.

To view a change log in Microsoft Word:

1. Do one of the following:
 - For a requirement or a building block in the hierarchical content table, select the object, and then click the **Attachments** tab to display the change log for the selected requirement or building block. You can open the **Attachments** window for this requirement or building block by clicking the **Open tab** button on the notebook pane's toolbar.
 - For a requirement or a building block in the **Links** or **Versions** tab:
 - . With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
 - . In the **Links** or **Versions** tab, select the requirement or building block to display its change log in the **Attachments** window.

2. In the **Attachments** tab or window, select the change log, and then pull down the **File** menu and choose **Open Read-Only**.

You can also right-click the change log and choose **Open Read-Only** from the pop-up menu.

The change log opens in Word as an **.mhtml** file. This file is deleted from your computer when you exit Architect/Requirements.

To view a change log in the Preview window:

For a requirement or a building block in the hierarchical content table:

1. Select the requirement or building block, and then click the **Attachments** tab to display the change log.
You can open the **Attachments** window for this requirement or building block by clicking the **Open tab** button on the notebook pane's toolbar.
2. With the **Preview** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Preview** window.
3. In the **Attachments** tab or window, select the change log to display its content in the **Preview** window.

For a requirement or a building block in the **Links** or **Versions** tab:

1. With the **Attachments** tab on top in the notebook pane, click the **Open tab** button on the toolbar to open the **Attachments** window.
2. In the **Links** or **Versions** tab, select the requirement or building block to display its change log in the **Attachments** window.
3. With the **Preview** tab on top, click the **Open tab** button to open the **Preview** window.
4. In the **Attachments** window, select the change log to display its content in the **Preview** window.

Appendix A: Glossary

This appendix defines Architect/Requirements terms.

B

Building Block

Object that can represent any element of a hierarchy, such as a function or a component of a product, a task in a work breakdown structure, or a job function in an organizational chart. Building blocks allow you to construct hierarchies that decompose systems and illustrate relationships among system elements.

Below each top level building block, subordinate building blocks can be organized in multiple levels of parents, children, and siblings. A system-defined building block subtype, **TRAM**, can be used for transitional mapping, a method of interrelating system views for comparison and analysis. To illustrate these relationships graphically, diagrams can be used in conjunction with building blocks. See also *Diagram*, *Complying Object*, *Defining Object*, *Full Block Number*, and *TRAM*.

C

Child

Object that is subordinate to a higher level object in a folder hierarchy. Compare with *Parent* and *Sibling*. See also *Direct Child* and *Member*.

Circular Reference

Defining trace link from a complying object back to its defining object. Circular references are allowed only if the defining and complying trace links are of different subtypes. See also *Complying Object*, *Defining Object*, and *Trace Link*.

Complying Object

Object that partially or completely fulfills a condition specified by a defining object. Trace links from complying objects can be traced up through the hierarchy to one or more originating sources. Compare with *Defining Object*.

D

Defining Object

Object that specifies a condition that a product or component must fulfill. Trace links from defining objects can be traced down through the hierarchy to the lowest level requirement that meets the original condition. Compare with *Complying Object*.

Derived Requirement

Requirement that has been translated into parameters suitable for lower level analysis and design. The inputs to the derivation process are higher level requirements. The outputs are new requirements.

Diagram

Object that graphically illustrates relationships for the object to which the diagram is attached. Typically, diagrams are attached to building blocks, which can represent superior and subordinate elements of a system view. Diagrams also can be attached to folders, requirements, and groups.

Diagrams are created and edited through live Visio, the Architect/Requirements interface with Microsoft Office Visio. Each live Visio diagram is interactively synchronized with the Architect/Requirements database, and is associated with a special stencil that contains shapes representing the Architect/Requirements object types. Objects can be created, modified, and deleted in the database by adding, modifying, and deleting the corresponding shapes in the diagram. The diagram is updated dynamically when members of the diagram owner are created, modified, and deleted in the database. See also *Building Block*. Compare with *Note* and *Spreadsheet*.

Direct Child

Child that occupies the level directly below the parent in a folder hierarchy. Compare with *Parent* and *Sibling*. See also *Child* and *Member*.

E

Editable Properties

System-defined or user-defined properties whose values can be changed directly by the user, subject to the permissions of individual users. Compare with *Read-Only Properties*.

F

Folder

Object that occupies the top level of organization within a project hierarchy. As folders contain files and other folders in Microsoft Windows, folders contain requirements, building blocks, groups, and other folders in Architect/Requirements.

Full Block Number

System-defined property that Architect/Requirements assigns to each building block. The full block number indicates the building block's level within a hierarchy of building blocks in a folder. The current level and full block number can be changed by promoting or demoting the building block, or by manually changing its **Number** property.

G

Group

Collection of references to existing objects that reside elsewhere in an Architect/Requirements project. Each reference associates one object as a *member* of the group. The referenced objects remain in their present locations, and can be maintained directly from the group without the need to switch to other folders or views. See also *Member*.

I

Indicator

Special graphic that symbolizes certain information about an object. There are four types of indicators: object type indicators, attachments indicators, links indicators, and versions indicators.

M

Member

Subordinate object occupying the next level below the immediate superior. Members of a folder, a requirement, or a building block reside in the superior object's location as direct children. Members of a group reside elsewhere in an Architect/Requirements project, retaining existing relationships to parents in those locations without becoming children of the group. The **Member Count** property shows the number of members for a selected object. See also *Direct Child* and *Parent*.

N

Note

Object that can be attached to folders, requirements, building blocks, and groups. Notes are useful for recording information that relates only to certain objects, rather than to all objects of a type. The content is created and edited in Microsoft Word. Compare with *Diagram* and *Spreadsheet*.

O

Object

Data element in an Architect/Requirements project. The objects types in the Systems Engineering and Requirements Management module are folders, requirements, building blocks, groups, notes, diagrams, and trace links.

P

Paragraph Number

System-defined property that Architect/Requirements assigns to each requirement. The paragraph number indicates the requirement's level within a hierarchy of requirements in a folder. The current level and paragraph number can be changed by promoting or demoting the requirement, or by manually changing its **Number** property.

Parent

Object to which one or more other objects are subordinate at lower levels of a folder hierarchy. Compare with *Child* and *Sibling*. See also *Direct Child* and *Member*.

Project

Object at the highest level of organization in Architect/Requirements, superior to all other objects. Projects define boundaries for user access, user customization of object types and properties, and organization of the other objects.

R

Read-Only Properties

System-defined properties whose values cannot be changed directly by the user. Compare with *Editable Properties*.

Requirement

Object whose content specifies one or more conditions that must be fulfilled by a system or a component. Requirement content also specifies the method by which the requirement is fulfilled. The content is created and edited in Microsoft Word. See also *Paragraph Number*, *Complying Object* and *Defining Object*.

Root Node

Topmost node in the navigation tree. Whether the root node is enabled or disabled, it always occupies the highest level in the tree. Below the root node, the navigation tree contains a node for each project to which you have access.

S

Sibling

Object that is subordinate to the same parent, and occupies the same level, as one or more other objects in a folder hierarchy. Compare with *Child* and *Parent*. See also *Direct Child* and *Member*.

Spreadsheet

Object that stores equations in the Architect/Requirements database. Spreadsheet objects can be created from worksheets in existing Microsoft Excel files. Equations in worksheet cells are preserved when the spreadsheet is opened, rather than being overwritten by updated property values from the database. Spreadsheets can be attached to folders, requirements, building blocks, and groups. Equations and other content can be edited in Excel. Compare with *Diagram* and *Note*.

Style Sheet

Formatting information specified when objects in an Architect/Requirements project are exported to a Microsoft Word document. For each exported object type, Architect/Requirements references the specified style sheet and applies that formatting in the export document.

System-Defined Properties

Properties that Architect/Requirements assigns automatically to built-in object types. Some system-defined properties receive a value that cannot be changed directly by the user, such as the object's creation date and time. Other system-defined properties, such as the object's name, receive a default value, which can be changed directly by the user. Compare with *User-Defined Properties*.

T

Trace Link

Object that establishes a directional relationship between two other objects, and indicates which object precedes, or defines, the other in the relationship. Defining and complying trace links can be created between objects that reside within the same Architect/Requirements project, and also can be created between objects that reside in different projects. Defining trace links can be created from objects in Architect/Requirements to complying objects in other Teamcenter products.

Trace links cannot be created from one trace link to another. See also *Complying Object* and *Defining Object*.

TRAM

System-defined building block subtype that can be used for transitional mapping, a method of interrelating system views for comparison and analysis. Through trace links, a TRAM can be associated with building block hierarchies to create a flow of information among source views and destination views, with the TRAM as the focal point. See also *Building Block* and *Trace Link*.

U

Uniform Resource Locator (URL)

Address that locates a specific resource on the Internet or on an intranet.

User-Defined Properties

Properties that an Architect/Requirements project administrator assigns to built-in object types and to custom subtypes. All user-defined properties are editable properties. That is, their values can be changed directly by the user, subject to the permissions of individual users. Compare with *System-Defined Properties*.

Appendix B: System-Defined Properties in the Systems Engineering and Requirements Management Module

This chapter describes the system-defined properties of the object types used in the Systems Engineering and Requirements Management module.

Overview of System-Defined Properties

Architect/Requirements automatically assigns system-defined properties to all built-in object types used in the Systems Engineering and Requirements Management module:

- *Folders*
- *Requirements*
- *Building blocks*
- *Notes*
- *Trace links*
- *Groups*

Each system-defined property falls into one of two categories:

- A *read-only* property has a value that cannot be changed, except as an indirect result of certain user actions on an object.
- An *editable* property has a default value that can be changed directly by the user.

For more information, see [Working With Object Properties](#).

Table of System-Defined Properties

The **Properties** tab and floating window display all viewable system-defined properties for the object selected in the content table. Other views in the Systems Engineering and Requirements Management module display certain system-defined properties by default. In these views, default properties can be removed and other system-defined properties can be added. For more information, see [Properties Tab](#) and [Using Tabs in Floating Windows](#) in chapter 3, *Using the Architect/Requirements Main Window*, and [Adding and Removing Columns](#) in chapter 9, *Working With Object Properties*.

Table B-1 lists each system-defined property, including the object types to which it applies and its description.

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
Approval Status	Requirement	<p>Current stage of the change approval process for this requirement. This property is read-only. Values are:</p> <ul style="list-style-type: none"> • Change Approved - the changes are approved by all approvers. • Change Rejected - the changes are rejected by at least one approver. • None - no changes are submitted. • Pending - changes are submitted and at least one approver response is outstanding.
Attachment Count	All object types	<p>In the Properties tab and window and the Edit Properties dialog window, the number of notes, diagrams, spreadsheets, and change approval objects attached to the selected object. This property is read-only.</p> <p>In other client views, contains a graphic indicator if the object has one or more notes, diagrams, spreadsheets, or change approval objects. To see the number of attachments, rest the pointer on the indicator to display a tooltip. You can click an indicator to see these objects in the Attachments tab or window.</p>
Baseline	Building block, Requirement	Names of the baselines associated with the object. This property is read-only.
Change Approval Pending	Change Approval	Indicates if a change approval is pending for the object to which the change approval object is attached. This property is read-only.
Change Approvers	Change Approval	List of users who have the responsibility to approve or reject the changes. This property is read-only.

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
Change Notifiers	Change Approval	List of users who are notified of submitted changes for informational purposes. Change notifiers do not approve or reject the changes. This property is read-only.
Change Rejected	Change Approval	Indicates if a change is rejected for the object to which the change approval object is attached. This property is read-only.
Change Time	All object types	<p>Date and time when the object was last modified. This property is read-only. For any object type, the property is updated when:</p> <ul style="list-style-type: none"> ● A defining or complying trace link is created on the object or is deleted from the object. ● The object is deleted, thus moving to the Recycle Bin of the deleting user. ● The object is restored from the Recycle Bin. <p>Additional change events depend on the object type:</p> <ul style="list-style-type: none"> ● For a requirement, the property is updated when the requirement content is edited, when the requirement is frozen, and when it is unfrozen. ● For a note, the property is updated when the note content is edited. ● For a building block, the property is updated when the building block is frozen and when it is unfrozen. ● For a folder, the property is updated when a member is added or deleted. <p>For additional details on when the Change Time property is updated, see the <i>Updates to Change Time and Change User Properties</i> topic in chapter 2 of the <i>Systems Architect/Requirements Management Project Administrator's Manual</i>.</p>
Change User	All object types	Login name of the user who last modified the object. Change events for this property are the same as those described above for the Change Time property. This property is read-only.
Complying Objects	Defining objects of all types	ROIN of each requirement, name of each building block, and OID of each object in another Teamcenter product that complies with the selected defining object. This property is read-only.

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
Copy Snapshot	Diagram	<p>Controls the behavior of the diagram image. Values are:</p> <ul style="list-style-type: none"> • No - The GIF image in the diagram image note is not updated even if the diagram is changed and saved. This is the default value. • Yes - The GIF image in the diagram image note is updated when the diagram is saved. <p>This property is editable.</p>
Create Time	All object types	Date and time when the object was created. This property is read-only.
Create User	All object types	Login name of the user who created the object. This property is read-only.
Date Deleted	Deleted objects of all types in Recycle Bin	Date and time when the object was deleted. This property is read-only.
Defining Objects	Complying objects of all types	ROIN of each requirement, and name of each object of all other types, that defines the selected complying object. This property is read-only.
Diagram Content	Diagram	<p>Controls the behavior of the shapes that represent the members of the live Visio diagram owner. This property is editable. Values are the following:</p> <ul style="list-style-type: none"> • Members fully synchronizes the set of shapes with the member set. As members are added to the owner and moved to other owners, shapes are added and removed. <p>A Members diagram is changed to Static when a shape is added for a non-member object or when a member shape is removed.</p> <ul style="list-style-type: none"> • Static prevents the automatic addition and removal of shapes as the member set changes. When members are added to the owner and moved to other owners, the set of shapes is unchanged. <p>The diagram remains connected to the database. Shapes for new members can be manually added to the diagram. Shapes automatically reflect changes to corresponding object properties.</p>

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
		Any shape can be manually removed. The corresponding object is not deleted. For both values, shapes are automatically deleted when corresponding objects are deleted in the client.
Direction	Connection	Direction of the connection. Values are Non-Directional , Uni-Directional , and Bi-Directional . This property is editable.
Document Template	Folder	Document template used when objects in the folder are exported to Microsoft Word. The document template controls the export process through object templates, which determine the data exported for the objects, and a style sheet, which determines the Word formatting applied to the data. This property is editable.
Expiry Date	Change Approval	Date after which the changes are automatically approved if one or more change approvers have not responded. This property is editable.
Full Name	All object types	Complete path to the object, beginning with the project name. This property is read-only.
GUID	Folder, Requirement, Building block, Group	Global Unique Identifier of the Teamcenter product containing an object that is linked to the selected object. This property is read-only.
Image Height	Note (Image Note subtype)	Height, in points, of the GIF image in the diagram image note. This property is editable.
Image Scale	Note (Image Note subtype)	Indicates whether the GIF image in the diagram image note is scaled from the original size. Values are No , the default, and Yes . If Yes , the Image Height and Image Width properties show the dimensions. This property is editable.
Image Width	Note (Image Note subtype)	Width, in points, of the GIF image in the diagram image note. This property is editable.
Member Count	Folder, Requirement, Building block, Group	Number of subordinate objects occupying the next lower level below the selected object. This property is read-only.
Name	All object types	Name of the object. This property is editable.

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
Number	Requirement, Building block	Current paragraph number of the requirement, or current full block number of the building block. Conforming to the numbered outline style, the value indicates the object's level in the hierarchy and its relationships to parent, sibling, and child objects. This property is editable.
OID	Folder, Requirement, Building block, Group	Object Identifier of an object (in another Teamcenter product) that is linked to the selected object. This property is read-only.
Original Location	Deleted objects of all types in Recycle Bin	Complete path from which the object was deleted. This property is read-only.
Project	All object types	Name of the project that contains the object. This property is read-only.
Reserved By	All object types	Login name of the user who is currently modifying the object. No other user can modify the object until the current change user releases the reservation by completing modifications. This property is read-only.
ROIN	Requirement	Requirement Object Identification Number of the requirement. Each ROIN is unique within the project This property is read-only.
Security Profile	All object types	Access rules that specify which users can view, modify, and delete the object. This property is editable.
Source Filename	Requirement	Name of the imported document from which the requirement was generated. This property is editable.
Source Paragraph	Requirement	In the imported document, the number of the paragraph that generated the requirement. This property is read-only.
Static Date	Building Block, Requirement	Date and time when the object was frozen. This property is read-only.
Stencils	Diagram	Live Visio stencils associated with the diagram. This property is editable.
Subtype	All object types	User-defined subtype assigned to the object. If no subtype is assigned, this property has the same value as

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
		the Type property for the object. This property is editable.
Synergy Check-In Comment	Requirement	Plain text information about a requirement that is checked in to IBM Synergy. The default comment is Check In from TeSE . This property is editable.
SynergyID	Requirement	Full path to a requirement in a IBM Synergy project. This property is read-only.
SynergyTask	Requirement	IBM Synergy task to which a checked out requirement is assigned. This property is editable.
Text	Requirement, Note	Plain text contained in the requirement or note. This property is editable.
Text Format	Requirement, Note	Format of the requirement or note content. This property is editable. Values are the following: <ul style="list-style-type: none"> ● Text, for plain text content. ● HTML, for rich text content. ● MHTML, for content entered in Microsoft Word.
Trace Link Count	Folder, Requirement, Building block, Group	In the Properties tab and window and the Edit Properties dialog window, the number of objects connected to the selected object by a trace link. This property is read-only. In other client views, contains a graphic indicator if the object has one or more trace links to other objects. To see the number of trace links, rest the pointer on the indicator to display a tooltip. You can click an indicator to see the linked objects and trace links in the Links tab or window.
Type Name	All object types	Built-in object type. This property is read-only.
Version Count	Building block, Requirement	In the Properties tab and window and the Edit Properties dialog window, the number of versions of the selected object. This property is read-only. In other client views, contains a graphic indicator if the object has one or more versions or variants. To see the number of versions or variants, rest the pointer on the indicator to display a tooltip. You can click an indicator

Table B-1. System-Defined Properties in Systems Engineering and Requirements Management Module

Property	Applies To	Description
Version Number	Building block, Requirement	to see the versions and variants in the Versions tab or window. Version number of this object. This property is read-only.
Version Type	Building block, Requirement	Indicates whether the object is frozen or a variant. Values are Static , for a frozen object, and Variant . An object can have both values. This property is read-only.
WhereUsed Object Count	All	<p>In the Properties tab and window and the Edit Properties dialog window, the number of times the object is referenced by another object. This property is read-only.</p> <p> -1 is displayed for certain objects, where calculating the correct value would require too much time. However, the Where Used tab or window always displays the correct information for these objects.</p> <p>In other client views, contains a graphic indicator if the object is referenced by one or more other objects. To see the number of referencing objects, rest the pointer on the indicator to display a tooltip. You can click an indicator to see the referencing objects in the Where Used tab or window.</p>

Appendix C: Live Office Interface Frequently Asked Questions

This chapter contains a list of Live Office Interface FAQs.

General

What are the prerequisites to use the Live Office Interface?

- Microsoft Office Office 2013 or Office 2016.
- Microsoft Office Visio 2013 or Visio 2016.
- The Microsoft .NET Framework versions 4.0 CLR.

Mixed installation of the Microsoft Excel 2013 and Microsoft Excel 2016 is not supported. When more than one version is installed, there is no reliable way for the Architect/Requirements client to launch one specific version. If you need to install Office 2016, you must install Excel 2013 on a different computer. You can also use a virtual machine for installing other versions of Microsoft Office.

For more information, see [Prerequisites for Using the Live Office Interface](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

Can I install Microsoft Office 2013, and Office 2016 on the same machine?

Yes. However, the live Office interface may become unstable. Microsoft recommends that you should install only one version of Office on your machine.

Mixed installation of the Microsoft Office Excel 2013 and Microsoft Office Excel 2016 is not supported. When more than one version is installed, there is no reliable way for the Architect/Requirements client to launch a specific version. If you need to install Microsoft Office 2016, then Excel 2013 must be available on a separate computer, or you can use the virtual machine for the installations.

What are the Architect/Requirements default addins?

Following addins are installed when you install Architect/Requirements.

- **TcROffice2K3Addin.Connect Friendly Name**
- **TcRExcelImport2K3Addin.Connect Friendly Name**
Active only in Microsoft Excel.

- **TcRVisioAddin.Connect Friendly Name**

Active only in Microsoft Visio.



Errors have been observed when these addins conflict with other custom addins. You must disable these addins to resolve the errors.

How do I disable Architect/Requirements addins in Microsoft Office application?

1. Launch the Microsoft Office application for which you want to disable the addin. For example, Microsoft Word, Microsoft Excel, or Microsoft Visio.
2. Click **File**.
3. Click **Options**.
4. In the **Options** dialog, click **Add-Ins** in the left pane.
5. Select **COM Add-ins** from the **Manage** drop down list and click **Go**.
6. In the **COM Add-Ins** dialog, clear the check box for the Architect/Requirements addin.
For information on the default Architect/Requirements addins, see [What are the Architect/Requirements default addins?](#)
7. Click **OK**.

What are the registry entries created by Architect/Requirements for the Live Office Interface?

- Entry for the Microsoft Office Excel on either 32-bit or 64-bit Windows:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\Excel\Addins\
TcROffice2K3Addin.Connect
LoadBehavior = 3
```

```
HKEY_CURRENT_USER\Software\Microsoft\Office\Excel\Addins\
TcRExcelImport2K3Addin.Connect
LoadBehavior =3
```

- Entry for the Microsoft Office Word on either 32-bit or 64-bit Windows:

```
HKEY_CURRENT_USER\Software\Microsoft\Office\Word\Addins\
TcROffice2K3Addin.Connect
LoadBehavior =3
```

- Entry for the Microsoft Office Visio on 32-bit Windows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Visio\Addins\
TcRVisioAddin.Connect
LoadBehavior =3
```

- Entry for the Microsoft Office Visio on 64-bit Windows:

```
HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Microsoft\Visio\Addins\  
TcRVisioAddin.Connect  
LoadBehavior =3
```



The value of the **LoadBehavior** registry entry must be set to 3 for the live office interface to work properly.

What should I do if the Architect/Requirements Live Office Interface does not work on my machine after I install the latest version of the Architect/Requirements client?

OR

How to fix the communication error message when I export to live Excel from Architect/Requirements?

Run the Architect/Requirements **Live Office Interface Diagnostic Tool** to diagnose any issues.

For more information, see [Using the Live Office Diagnosis](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

If the **Live Office Interface Diagnostic Tool** reports that the Office add-ins are not registered on your system, then go to your client installation folder and run the **register_office.bat** and **register_visio.bat** batch files.



To run the **register_office.bat** and **register_visio.bat** batch files, the supported version of Microsoft Office and Microsoft Office Visio must be installed on your system.

What should be the security settings in the Microsoft Office Excel, Microsoft Office Word, and Microsoft Office Visio in order to use the Architect/Requirements Live Office Interface?

- Configure Microsoft Word to use the following settings:

Macro Settings: Choose the **Disable all macros except digitally signed macros** option.

Add-ins: Check the **Require Application Add-ins to be signed by Trusted Publisher** check box. The unsigned add-ins present on the system will not work in this case.

For more information, see [Configuring Microsoft Office for the Live Office Interface](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

Where can I find the log file for Live Office Interface?

The log file for the live Office interface captures errors encountered while using any of the live Office interface features. It helps in identifying and troubleshooting any problems.

The log file can also capture trace, which enables logging every Architect/Requirements communication in addition to errors. You can enable logging by setting **Trace = True** in the **Trace.txt** file available at *C:\Documents and Settings\%User Name%\Application Data\TcRequirements*.

- The live Office interface log file is located at:

C:\Users\%User Name%\AppData\Roaming\TcRequirements\OfficeInterface_log.txt.

Microsoft Office Word Troubleshooting

What do I do if the formatting is lost while exporting requirements to Microsoft Word?

Bullets and Numbering formatting is lost when a requirement is exported from Architect/Requirements to Microsoft Word. The problem occurs when you format the text directly instead of formatting the text using the styles from the style sheet attached to the document. The direct formatting is also lost when you apply the **AutoNumbering** and **AutoBullet** styles or on saving the objects as Microsoft Word tags the bullets and the numbers as same. To preserve the bullets and numbering, apply styles containing the **Bullet** or **Number** format. This allows preserving the formatting in the style sheet.

Bullets format applied to a document are converted to numbers if both of the following conditions are true:

- The text of the requirement includes automatic bullets.
- The style sheet uses automatic numbering for the headings.



Note that the bullets are displayed correctly in the **Preview** tab in the **Notebook** pane when the folder's document template is set to the one used for export.

How to fix problems Importing Microsoft Word Document?

If you have moved from Microsoft Office 2010 to Microsoft Office 2013 or a later version and are getting errors in importing Word documents using keyword parsing, you must clear the Microsoft Form cache by running the diagnostic tool for Word.

To clear the Microsoft Form cache

1. Open the Architect/Requirements application URL.
2. Click the **Administrative Tools** link.
3. Click the **Diagnostic Tools** link.
4. Click the **Live Office Diagnosis** link.
5. In the **TcSE Office Interface Diagnostic Tool**, unselect all check boxes and select **Word** only.
6. Click **Diagnose**.
7. Click **No** to prevent launching Word.

The **Word Diagnostic Information** in the **TcSE Office Interface Diagnostic Tool** window displays a log of the diagnosis performed.

8. Click the **Clear cache** link.
9. Click **Close**.

Microsoft Office Excel Troubleshooting

What should I do if the live Excel functionality does not work on my system? The diagnostic tool does not suggest any errors. The security and registry settings are correct.

Verify that the supported versions of Microsoft Office and Microsoft Office Visio are installed on your system.

Mixed installation of the Microsoft Office Excel 2013 and Excel 2016 is not supported. When more than one version is installed, there is no reliable way for the Architect/Requirements client to launch one specific version. If you are required to install the Microsoft Office 2016, then Excel 2013 should be available on a separate system. You can also use a virtual machine for the other versions.

How to fix the missing Architect/Requirements toolbar in Microsoft Office Excel?

OR

How to view the Architect/Requirements menu when I open a saved live Excel workbook?

OR

What should I do if the Microsoft Office Excel Add-ins do not load after I open a saved live Excel workbook?

To fix any of the above issues, follow the steps below:

1. Launch Microsoft Excel and open a live Excel file.
2. Check the security settings of Microsoft Office Excel.

Configure Microsoft Excel to use the following settings:

Macro Settings: Choose the **Disable all macros except digitally signed macros** option.

Add-ins: Check the **Require Application Add-ins to be signed by Trusted Publisher** check box. The unsigned add-ins present on the system will not work in this case.

For more information, see [Configuring Microsoft Office for the Live Office Interface](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

3. Kill any Microsoft Office Excel processes running in the background using the **Task Manager**. Reopen the work book as mentioned in step 1.

To diagnose any further issues, you may run the Architect/Requirements **Live Office Interface Diagnostic Tool**.

For more information, see [Using the Live Office Diagnosis](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

I made changes in the saved independent live Excel workbook after I connected to the Architect/Requirements server. I cannot see these changes in the Architect/Requirements client. What could be the reason?

Refresh the Architect/Requirements client to synchronize and view the changes from the Microsoft Office Excel workbook.

In what format should I save my live Excel workbook to retain the live behavior after I open the saved live Excel sheet?

- Save in the **XLSM** format.

I edited a requirement (a Microsoft Office Word document) from a live Excel workbook using the Architect/Requirements Open menu in the live Excel sheet. However, the changes are not reflecting in the Architect/Requirements client. What could be the issue?

Run the Architect/Requirements **Live Office Interface Diagnostic Tool** to diagnose the live Word interface. Ensure you are connected to Architect/Requirements.

For more information, see [Using the Live Office Diagnosis](#) in the chapter 2, *Installing the Architect/Requirements Client with Office Integration*.

Microsoft Office Visio Troubleshooting

I open a saved live Visio diagram from my local disk, connected to Architect/Requirements and changed the property of object, but I cannot see the property value updated in Architect/Requirements client. What could be the issue?

Refresh the Architect/Requirements client to synchronize and view the changes.

What could be the issue if adding a master shape in live Visio diagram does not create type or subtype in Architect/Requirements?

- Check the mapping file for the type or subtype mapping.
For more information on customizing the interface with Microsoft Office, see the *Systems Architect/Requirements Management Project Administrator's Manual*.
- Run the **Stencil Diagnostic Utility** available as a command when you right-click on a live Visio diagram. You can identify the subshape, diagnose the connection point, verify the property mapping file, and verify the stencil.

For more information, see [Stencil Diagnostic Utility](#) in the chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

What should I do if I cannot open at least one stencil object when I try to create a new live Visio diagram?

You must register the type library on your system again:

1. Locate the **regtlib.exe** file on your machine. Usually, it is located at *C:\WINDOWS\system32\URTTemp*.
2. Locate the **vislib.dll** file on your system.
The **vislib.dll** file is usually located at *C:\Program Files\Microsoft Office\Office12(OR 11)\vislib.dll*.
3. Open a command prompt and change the directory to the path of the **regtlib.exe** file.
4. Run the **regtlib C:\Program Files\Microsoft Office\Office12(OR 11)\vislib.dll** command. Change the path of the file if it is different.

How can I map more than one Architect/Requirements property to a Microsoft Office Visio master shape?

You must create a group master shape that has as many shapes as the number of properties you want to map. You must know the sequence in which you have added the shapes to the group shape. In property mapping file, you can map various properties with each of the sub shape of the group shape.



You can use the **Identify Subshape** command to identify the subshape sequence.

For more information, see [Stencil Diagnostic Utility](#) in the chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

How can I identify the sequence of the sub shapes in a group shape?

On a live Visio diagram, right-click and run the **Identify Subshape** command to identify the sequence of the subshapes in a group shape.

For more information, see [Stencil Diagnostic Utility](#) in the chapter 6, *Constructing System Views With Building Blocks and Diagrams*.

While creating or opening Visio diagram, some of the shapes appear outside the Visio page. What should I do?

1. Click on the Visio page and choose the **File→Page Setup** command.
2. Click on **Page Size** tab.
3. Select the **Same as printer paper size** option, and then the **Size to fit drawing contents** option.
4. Click the **Apply** or the **OK** button.

Troubleshooting for the Architect/Requirements Client

The client installation of Systems Architect/Requirements Management is failing. What could be the possible cause?

The installation of Systems Architect/Requirements Management fails.

The error occurs when you click **Launch Teamcenter systems engineering** after opening the Systems Architect/Requirements Management page. On a client machine that does not have Systems Architect/Requirements Management installed, clicking the **Launch Teamcenter systems engineering** link installs the Systems Architect/Requirements Management client.

To resolve the error, check and modify the following Web application configuration:

1. On the Systems Architect/Requirements Management home page, click **Administrative Tools**.
2. On the **Administrative Tools** page, click **Web Application Configuration**.
3. Provide the logon credentials and click **Log In**.
4. Locate the **WOLF.AppIP** parameter. If it is set to **localhost**, change it to the machine name on which Systems Architect/Requirements Management is installed.
5. Click **Update** to save the changes.

How do I unregister and register the Architect/Requirements Client and Microsoft Office?

For the Architect/Requirements Microsoft Office interface functionality, the required libraries must be registered. The Microsoft Office interface libraries get registered each time client is launched. When the Microsoft Office interface is invoked, Architect/Requirements checks the registry for the installed version of Microsoft Office. It compares the version with the version of the client. If the versions do not match, Architect/Requirements attempts to unregister the old version and register the version installed on the computer. If the process of unregistering and registering the client, users need to manually unregister and register the Architect/Requirements client.

You require Power User or Administrator privileges for registering or unregistering the Architect/Requirements client, Microsoft Office, Microsoft Visio.

To unregister the Architect/Requirements client, run the following batch files from the client installation directory:

unregister_client.bat

To register the Architect/Requirements Client, run the following batch files from the client installation directory:

register_client.bat

To unregister Microsoft Office, run the following batch files from the client installation directory:

unregister_office.bat

To register Microsoft Office, run the following batch files from the client installation directory:

register_office.bat

To unregister Microsoft Visio, run the following batch files from the client installation directory:

unregister_visio.bat

To register Microsoft Visio, run the following batch files from the client installation directory:

register_visio.bat

What should I do if I cannot create a live Visio diagram from the Architect/Requirements client and the Visio Live—>Create Diagram menu is disabled?

You need the **Architect** privilege to create a Visio diagram. Please contact your project administrator for setting up the privilege.

I am not able to open a Microsoft Office Word document from the Architect/Requirements client. I get several errors related to com.inzoom.comjni.ComJniException.eComError. How I can get rid of those?

Un-register the **jacozoom** library by:

```
$Directory$ regsvr32 IzmJniComAx.dll /u
```

Re-register the **jacozoom** library by:

```
$Directory$ regsvr32 IzmJniComAx.dll /s
```

In the above commands, *\$Directory\$* is the folder where **IzmJniComAx.dll** file is located. For Architect/Requirements 2007.x release, the file location is the **System32** folder on your machine. For Architect/Requirements 11.1 release, the file location is the Architect/Requirements client installation folder.

If the above actions do not resolve the issue, verify if you have more than one version of the Architect/Requirements client installed on the same machine. The versions could be the Architect/Requirements 7.1.4 or earlier and the Architect/Requirements 2007.2 or later. Remove all of them and reinstall the version you intend to work with.

How to fix the several pop up messages I get related to vtable call when I launch the Architect/Requirements client?

One of the error messages you get is **Something has gone wrong with vtable call. Stack is corrupted. Please report to infoZoom.** If you have installed the Architect/Requirements 2007.1.4 and the Architect/Requirements TcSE 2007.2 or later on the same machine, you might get these error messages. The reason is the **jacozoom** DLL files are not backward compatible.

To avoid getting these errors do following:

1. Close the Architect/Requirements clients and go to the installation folder of the Architect/Requirements 2007.2 or later version.
2. Copy the **izmcomjni.jar**, **izmcomtlb.jar**, and **izmjnicom.dll** files to the Architect/Requirements 2007.1.4 client installation folder.

When you launch the Architect/Requirements client, you should not get this error.

Where can I find the log file for the Architect/Requirements client?

The log file for the Architect/Requirements client captures errors encountered while using any of the features. It helps in identifying and troubleshooting any problems.

- The Architect/Requirements client log file is located at:

C:\Users\%User Name%\AppData\Roaming\TcRequirements\tcr_log.txt.

What are the log files that I can use for troubleshooting?

You can use Systems Architect/Requirements Management log files for debugging purposes.

To view the location of the client log file:

1. Select **Tools**→**System Information**.
2. Click **Client Log**.

The client log tab displays the path for the log file. You can navigate to the log path from Windows Explorer. The following log files are present in the log path location:

- **tcr_log.txt**

This is the only file with up to date client log information and the only file that you must use for debugging.

- **tcr_log_old.txt**

This is a backup of the log file. You can use this file for debugging a problem that happened earlier than the first item shown in **tcr_log.txt**.

- **temp_tcr_log.txt**

This is a snapshot of the log file created when you click the **Open In Notepad** button. This file is not kept up to date and you must not use it for collecting client log information.

Appendix D: Changing the maximum memory available for rich client

The Architect/Requirements rich client is a 32-bit Java application. By default, the maximum memory available to the rich client is set to 1 GB. This is adequate most of the time, but you can change it using the procedure below if it is insufficient. One of the conditions where you need to increase the maximum memory is when you get an out of memory condition from a large search result or import of a large document.



The client systems running the rich client must have sufficient physical memory installed to accommodate the maximum memory setting that you choose. Additionally, your client systems may need memory for other applications, so verify the feasibility of the maximum memory setting with your IT department before making a change.

To modify the client memory allocation:

1. Create a temporary folder and extract the **launch.jsp** file from the **tcr.war** file.

Open the command prompt, change to the directory location of the **tcr.war** file, and run the following command:

```
jar xvf tcr.war ugs/tc/req/launch.jsp
```



You must have the JDK installed to run the **jar** command.

You can download JDK from

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

2. Open the **ugs/tc/req/launch.jsp** file in a text editor like **Textpad** or **Notepad++**.
3. Search for the following string:

```
<PARAM name="vm_args" value="-Xms64m -Xmx1024m"> <!-- vm args for launching client -->
```

Xmx1024m in the string denotes the maximum heap space. In this case, the heap space is set to **1024 MB** or **1 GB**. Change it to the desired value.

For information about the optimum heap space, see the Java documentation at <http://docs.oracle.com/>.

4. Add the updated **launch.jsp** file to the **tcr.war** file. Run the following command at the command prompt:

```
jar uvf tcr.war ugs/tc/req/launch.jsp
```

5. Deploy the updated **tcr.war** file on your Web server.

Index

- .gif images, diagrams202
- .html file name extension..... See Excel or Word
- .mhtml file name extension.....See Word
- .stp file name extension..... 138
 - Importing objects 152
 - Updating properties.....290
- .xml file name extension 142, 156
- Action column, Edit Properties dialog window
.....284
- Activating the Systems Architect/Requirements
Management Search module.....340
- ADD statement, queries372
- Adding
 - Columns272
 - Group members.....128
- Address bar
 - Disabling or enabling58
 - Hiding or showing.....59
 - Overview.....58
 - Resizing59
- Advanced search view362
- AP 233 data format
 - Updating property values290
- AP233 data format
 - Exporting project data 138
 - Importing project data 152
- AP233 STEP File
 - Exporting project data 138
- AP233 STP File
 - Exporting project data 138
- AP233 XML file
 - Importing objects 154
- AP233 XML File
 - Exporting project data 140
- Applying a column view
 - Content table267
 - Notebook pane268
- Applying views to folders265
- Approval Status property396
- Approved, change management event380
- Approver, change management user379, 381
- Approving a change..... 381
- Attaching
 - Data definitions to connections 228
 - Data dictionaries to diagrams 216
 - Diagrams..... 201
 - Notes 246
 - Spreadsheets 308
- Attachment Count property 64, 91, 396
- Attachments indicator 64, 68, 91
- Attachments tab
 - Columns
 - Adding and removing 272
 - Rearranging 274
 - Resizing 275
 - Sorting information..... 275
 - Description..... 75
 - Exporting to Excel 143
 - Using in floating window 90
- Auto Refresh check box, Data Definitions dialog
window 218
- Baseline Name dialog window 336
- Baseline property396
- Baseline with in-work, effectivity rule .. 328, 337
- Baseline, effectivity rule 328, 337
- Baselines
 - Creating 335
 - Viewing 337
- Basic search view 353
- Behavior
 - Reference links 310
- Browser
 - Installing Systems Architect/Requirements
Management client..... 33
 - Starting Systems Architect/Requirements
Management
 - Home page 45
 - Object URL..... 122
- Browser and dialog window examples 17
- Building blocks
 - Adding to groups 128
 - Baseline

Creating.....	335	Search results.....	341
Viewing.....	337	Shortcuts	
Connections		Creating	133
Creating, Architect/Requirements client	221	Hiding and showing children of master	
Creating, editing, and deleting	223	objects.....	134
Viewing.....	231	Navigating to master objects	134
Copying.....	119	Overview	131
Creating.....	192	Replacing with copies of master objects	135
In live Excel	305	Spreadsheets, attaching.....	308
Defining and complying objects, viewing .	238	Subtypes.....	192
Deleting.....	162	Trace links	
Demoting	195	Creating	234, 237
Diagrams		Deleting	243
Attaching.....	201	Navigating to linked objects	242
Editing.....	204	Viewing	241
Viewing, if Visio not installed.....	214	Trace Links	
E-mail, sending object data.....	135	Creating, Copy to Diagram, and Paste to	
Exporting		TcSE	231
AP233 data file	138	TRAM subtype	
Word	146	Adding to groups	128
XML data file.....	142	Creating	192
Freezing	329	Overview	191
Importing		Removing from groups.....	129
AP233 STP File	152	Unfreezing	330
AP233 XML File	154	URLs, copying.....	122
Excel	158	Versions and variants	
XML data file.....	156	Creating a variant.....	332
Linking		Creating a version.....	331
To other Teamcenter products	237	Deleting	334
Within Systems Architect/Requirements		Viewing	333
Management.....	234	Buttons, Systems Architect/Requirements	
Members, filtering.....	97	Management toolbar	51
Members, viewing.....	95	Calculating numeric property values with	
Moving.....	123	formulas.....	288
Notes, attaching.....	246	Calendar, date property value	280, 285, 287
Number property.....	193, 194, 195	Case Sensitive check box, Search module.....	356
Overview.....	191	Change approval object	
Promoting.....	195	Modifying.....	383
Properties		Viewing	384
Calculating numeric values with formulas		Change Approval Pending property	396
.....	288	Change Approvers property.....	396
Editing.....	279, 283, 286, 292	Change log indicator.....	66
System-defined, descriptions	396	Change Log property	66
Updating, from AP 233 data file.....	290	Change log, viewing.....	386
Removing from groups	129	Change management	
Renaming	125	Approving a change.....	381
Restoring.....	163	Change approval object	
Search module		Modifying	383
Advanced search view	362	Viewing	384
Basic search view.....	353	Change log, viewing	386
Intermediate search view	358	Events	

Approved.....	380	Columns	
Rejected.....	380	Adding and removing	272
Response	380	Overview	272
Submit	379	Rearranging	274
Overview	379	Resizing	275
Rejecting a change	381	Sorting information.....	275
Submitting changes for approval	380	System default view.....	266, 267
Users		Views, applying to folders.....	265
Approver	379, 381	Comparing	
Changes waiting for approval, gaining		Requirements in two folders	189
access	382	Two requirements	188
Notifier.....	380	Comparing the content of different requirements	
Change Notifiers property.....	397	187
Change Password dialog window	110	Comparison mode, requirements	187
Change Rejected property	397	Complying Object Traceability window	
Change Time property	397	Navigating to linked objects	242
Change User property	397	Viewing complying objects	238
Changing		Complying objects.....	233
Diagram type, live Visio	211	Complying Objects property.....	397
Levels, object hierarchy	184, 195	Complying Trace column, description	77
Password, Systems Architect/Requirements		Connections	
Management.....	110	Connectivity tab.....	83
Property values		Creating	
AP 233 data file	290	Architect/Requirements client	221
Edit Properties dialog window	279	Creating, editing, and deleting.....	223
live Excel	292	Data definitions	
Multiple object selections	283	Attaching	228
Properties tab	279, 283	Deleting	226
Table view cells	286	Names, hiding and showing.....	227
Saved views	272	Viewing	231
Saved views, hierarchical content table	276	Connectivity tab	
Child diagrams	203	Description.....	83
Child Object, Deleting	163	Constructing Advanced Queries	367
Child objects		Containing text field, Search module	355, 364
Creating		Containing text field, wildcard characters	355
Building blocks	194	Content elements, requirements.....	175
Folders.....	116	Content table	
Requirements	170	Column view, conditions	266
Filtering.....	97	Columns	
Viewing.....	95	Adding and removing	272
Choice property		Hierarchical view.....	64
Definition	263	Rearranging	274
Value, changing		Resizing	275
Edit Properties dialog window	279	Sorting information.....	275
Live Excel interface	298, 303	Systems Architect/Requirements	
Properties tab	279	Management Recycle Bin.....	68
Table view cells	286	Default columns.....	267
Choosing an effectivity rule.....	328	Disabling or enabling.....	70
Column Settings dialog window		Exporting to Excel	143
Adding and removing columns	272	Filtering objects in a hierarchy	97
Rearranging columns	274	Hiding or showing	70

Hierarchical.....	64	Rich text format	248
Overview.....	63	Word.....	250
Plus signs	64, 95, 113	Ports.....	220
Resizing	71	Reference links	
Switching to.....	70	Full content.....	319
Systems Architect/Requirements Management		Plain text.....	318
Recycle Bin.....	68	Requirements	
Viewing objects in a hierarchy.....	95	By importing from Word.....	171
Views, applying to folders	265	Child of existing requirement	170
Conventions		In Architect/Requirements client views.....	168
Browser and dialog window examples	17	Sibling of existing requirement	169
Names	18	Subtypes.....	168
Revisions.....	17	Top level of folder	169
Values	18	Saved views, content table.....	269
Copy Connection, live Visio menu option.....	199	Shortcuts	133
Live Visio diagram	225	Spreadsheets	308
Copy Port, live Visio menu option.....	199	Trace links	
Copy SnapShot menu.....	309	To other Teamcenter products	237
Copy Snapshot property.....	202, 398	Within Systems Architect/Requirements	
Copy to Diagram		Management	234
Trace Links		Trace Links	
Live Visio diagram	232	Live Visio diagram	232
Copy to Diagram, live Visio menu option		Variants.....	332
Relations subtab	225, 232	Versions.....	331
Copying		Views, folder	269
Copy SnapShot to requirement	309	Current Frozen Version, effectivity rule.....	328
Objects	119	Current Version, effectivity rule.....	328
Reference links, in copied objects.....	313	Data definitions	
URLs of objects	122	Attaching to connections	228
Visio diagram to live Visio.....	209	Creating	
Create Note On dialog window		In Architect/Requirements client.....	217
MHTML note, Word.....	250	In diagram.....	218
Plain text note	247	Shortcuts	
Rich text note	248	Creating	133
Create Time property	398	Hiding and showing children of master	
Create User property	398	objects.....	134
Creating		Navigating to master objects	134
Baselines	335	Overview	131
Building blocks	192	Replacing with copies of master objects.....	135
Connections		Data Definitions dialog window.....	218, 228
Architect/Requirements client.....	221	Data dictionaries	
Live Visio diagram	224	Attaching to diagrams.....	216
Data definitions		Creating	215
In Architect/Requirements client	217	Shortcuts	
In diagram	218	Creating	133
Data dictionaries	215	Hiding and showing children of master	
Diagrams.....	201	objects.....	134
Folders	115	Navigating to master objects	134
Groups.....	127	Overview	131
Notes	246	Replacing with copies of master objects.....	135
Plain text format.....	247	Date Deleted property.....	398

Date property	
Definition	263
Value, changing	
Edit Properties dialog window	279
Live Excel interface	298, 303
Properties tab	279
Table view cells	286
Debugging options, Architect/Requirements	
client	107
Default	
Property columns, content table	266, 267
View	
Setting for folder	270
Setting for user name	271
Default View option, View list	267
Defining Object Traceability window	
Navigating to linked objects	242
Viewing defining objects	238
Defining objects	233
Defining Objects property	398
Defining Trace column, description	77
Delete in TcSE, live Visio menu option	199
Delete in Teamcenter for systems engineering,	
live Visio menu option	226
Deleting	
Connections	226
Folders, requirements, building blocks,	
groups, notes	162
Reference links	314, 323
Trace links	243
Versions, variants	334
Demoting	
Building blocks	195
Requirements	185
Diagnose Connection Point, live Visio menu	
option	200
Diagram Content property	208, 211, 398
Diagram Image note	202
Diagram Inputs dialog window	201
Diagrams	
Adding to groups	128
Connections	
Creating, editing, and deleting	223
Viewing	231
Copy Snapshot property	202
Copying from Visio to live Visio	209
Creating	201
Data definitions	
Creating	218
Data dictionaries	
Attaching	216
Creating	215
Disconnecting from Systems	
Architect/Requirements Management	
database	213
Editing	204
E-mail, sending object data	135
Embedding, Word	319
Exporting	
AP233 data file	138
Word	146
XML data file	142
Images (.gif)	202
Importing	
AP233 STP File	152
AP233 XML File	154
Excel	158
XML data file	156
Overview	191
Ports, creating	220
Removing from groups	129
Shapes, default arrangement	202
Trace Links	
Creating, Copy to Diagram, and Paste to	
TcSE	231
Types, changing	211
URLs, copying	122
Viewing, if Visio not installed	214
Dialog Window examples	17
Dialog windows	
Baseline Name	336
Change Password	110
Column Settings	
Adding and removing columns	272
Rearranging columns	274
Create Note On	
MHTML note, Word	250
Plain text note	247
Rich text note	248
Data Definitions	218, 228
Diagram Inputs	201
Document Import	174
Edit	253
Edit Properties	
Data dictionaries, attaching to diagrams	216
Diagram type, changing	211
Multiple object selections	283
Properties, editing	279
Properties, filtering	277
Text Format property	182, 255
Export To Excel	145, 295
Import AP233	

Importing objects	153	Document Import dialog window	174
Updating properties	290	Document Options property	
Multi-Choice	279, 283, 286, 326	Exporting objects to Word	148
Open	153, 155, 157, 159, 205, 290, 308	Importing requirements from Word	174
Print	102	Document template	146
Proxy Server Settings		Document Template property	173, 399
Starting new Architect/Requirements		Document Template Rules property	316, 321
session	46	Document, folder subtype	
Save	139, 141, 142	Adding to groups	128
Save Properties	103	Applying views	265
Save View	269	Creating	115
Saved Report Information	345	Removing from groups	129
Search Results	341, 342	Edit dialog window	253
Select a Baseline	337	Edit Properties dialog window	
Select Document Template	149	Data dictionaries, attaching to diagrams	216
Comparing requirements in two folders	189	Diagram type, changing	211
Comparing two requirements	188	Properties	
Select Object Chooser	237	Editing	279
Select Report	345	Filtering, property tables	277
Select Saved Report	345	Multiple object selections	283
Select Subtype	207	Text Format property	182, 255
Building blocks	193	Edit Search Criteria pane, Search module	358, 360
Data definitions	217	Editable properties	
Data dictionaries	215	Definition	261
Folders	115, 116, 117	System-defined, descriptions	396
Groups	128	Values, changing	279, 283, 286, 292
Notes	250	Values, numeric, calculating with formulas	
Requirements	169, 170	288
Trace links	235	Values, updating from AP 233 data file	290
Select User	136	Viewing	272
Send Email	136	Editing	
Single-Choice	149, 279, 283, 286	Change approval object	383
Comparing requirements in two folders	189	Connections	
Comparing two requirements	188	Live Visio diagram	224
Data dictionaries, attaching to diagrams	216	Diagram type, live Visio	211
Diagram Content property	211	Diagrams	204
Text Format property	182, 255	Groups	128, 129
Submit For Approval	381	Notes	
System Information	103	Plain text format	252
Client Log tab	105	Rich text format	252
Logging Options tab	107	Text property	257
Teamcenter for systems engineering		Word	252
properties	206	Properties	
Toggle Relationship	351	Calculating numeric values with formulas	
Direction property	399	288
Disable Numbering value, Document Options		Edit Properties dialog window	279
property		From AP 233 data file	290
Exporting objects to Word	148	Live Excel	292
Importing requirements from Word	174	Multiple object selections	283
Disconnecting a live Excel spreadsheet	307	Properties tab	279, 283
Disconnecting a live Visio diagram	213		

Table view cells	286	PLM XML	150
Requirement content		Word	146
Content elements.....	175	XML data file	142
Formatting.....	175	Reference links, in exported objects	315
Page setup	177	Reports, Search module	
Text property cells	179	Excel	347
Word	177	Visio.....	350
Requirements		Word.....	349
Word	175	System information to Excel	103
Effectivity		Filter operators.....	101
Definition	325	Filter toolbar	98
Rule, filtering versions.....	328	Filtering data.....	101
Elements of Systems Architect/Requirements		Filtering objects	97
Management main window.....	48	Filtering objects in a hierarchy	97
E-mail, sending from Systems		Filtering properties	97
Architect/Requirements Management client		Filtering property tables.....	277
.....	135	Filtering versions by effectivity rule.....	328
Embedding, diagrams, Word	319	First, button, Data Definitions dialog window	
Emptying Systems Architect/Requirements		228
Management Recycle Bin.....	165	Floating windows	
Enabling Text property editing		Notebook pane tabs	90
Notes	255	Toolbar buttons.....	55
Requirements	182	Folders	
Enabling versions for a project	326	Adding to groups	128
End a connection subtype, toolbar button 54, 221		Applying views.....	265
End a connection, toolbar button	54, 221	Baseline	
End Link Using Subtype, live Visio menu option		Creating	335
.....	198	Viewing	337
End Link, live Visio menu option.....	198	Column Settings	272
Equations, using in Excel spreadsheet	308	Comparing requirements	189
Excel		Connections	
Editing properties, live Excel.....	292	Creating, editing, and deleting.....	223
Exporting objects	143	Viewing	231
Exporting reports, Search module.....	347	Copying	119
Exporting system information.....	103	Creating	115
Exporting table views	343	In live Excel.....	305
Importing objects	158	Default view, setting for folder.....	270
Live interface, Excel template	347	Defining and complying objects, viewing .	238
Search results output option.....	342	Deleting	162
Spreadsheets, attaching to objects.....	308	Diagrams	
Template	143, 293, 347	Attaching	201
Exclude Shortcuts check box, Search module	356	Editing	204
Exiting Systems Architect/Requirements		Viewing, if Visio not installed.....	214
Management.....	110	Document subtype	115
Expanding projects and folders.....	60	Adding to groups	128
Expiry Date property.....	399	Removing from groups.....	129
Export To Excel dialog window	145, 295	Document Template property	146, 175
Exporting		Document Template Rules property ..	316, 321
Objects		E-mail, sending object data.....	135
AP233 data file	138	Expanding.....	60
Excel	143	Exporting	

AP233 data file	138	Views	
PLM XML	150	Creating	269
Word	146	FOR EACH statement, queries.....	372
XML data file.....	142	Formatting, requirement content	175
Importing		Formulas, calculating numeric property values	
AP233 STP File	152	288
AP233 XML File	154	Freezing objects.....	329
Excel	158	Frozen As Of Date, effectivity rule	328
XML data file.....	156	Full Name property.....	129, 399
Linking		Generic links.....	244
To other Teamcenter products	237	Behavior.....	244
Within Systems Architect/Requirements		Creating, Modifying	244
Management.....	234	Get connection from Visio diagram, live Visio	
Members, filtering.....	97	menu option	225
Members, viewing.....	95	Get Connection From Visio Diagram, live Visio	
Moving.....	123	menu option	198
Notes, attaching.....	246	Go To Parent Diagram, live Visio menu option	
Opening.....	113	198
Properties		Go To TcSE, live Visio menu option ...	198, 204, 223
Calculating numeric values with formulas		Groups	
.....	288	Adding members.....	128
Editing.....	279, 283, 286, 292	Connections	
System-defined, descriptions	396	Creating, editing, and deleting.....	223
Updating, from AP 233 data file.....	290	Viewing	231
Removing from groups	129	Copying	119
Renaming	125	Creating	127
Restoring.....	163	In live Excel.....	305
Search module		Defining and complying objects, viewing .	238
Advanced search view	362	Deleting	162
Basic search view.....	353	Diagrams	
Intermediate search view	358	Attaching	201
Search results	341	Editing	204
Shortcuts		Viewing, if Visio not installed.....	214
Creating.....	133	E-mail, sending object data.....	135
Hiding and showing children of master		Exporting to Word.....	146
objects	134	Importing from Excel	158
Navigating to master objects.....	134	Linking	
Overview.....	131	To other Teamcenter products	237
Replacing with copies of master objects	135	Within Systems Architect/Requirements	
Spreadsheets, creating.....	308	Management	234
Style Sheet Source property	173	Members	
Subtypes.....	115	Filtering	97
Trace links		Viewing	95
Creating.....	234, 237	Moving.....	123
Deleting.....	243	Notes, attaching	246
Navigating to linked objects	242	Overview	126
Viewing.....	241	Properties	
Trace Links		Calculating numeric values with formulas	
Creating, Copy to Diagram, and Paste to		288
TcSE.....	231	Editing	279, 283, 286, 292
URLs, copying	122		

System-defined, descriptions	396
Updating, from AP 233 data file	290
Removing members	129
Renaming	125
Restoring	163
Search module	
Advanced search view	362
Basic search view	353
Intermediate search view	358
Search results	341
Spreadsheets, attaching	308
Subtypes	127
Trace links	
Creating	234, 237
Deleting	243
Navigating to linked objects	242
Viewing	241
Trace Links	
Creating, Copy to Diagram, and Paste to TcSE	231
URLs, copying	122
GUID property	399
Hide/Unhide Connection Name, live Visio option	198
Hide/Unhide Connection Name, live Visio option	227
Hide/Unhide Connection Names, live Visio option	227
Hide/Unhide Marked Connection Name, live Visio menu option	198
Hide/Unhide Marked Connection Names, live Visio option	227
Hide/Unhide Port Name, live Visio menu option	198
Hide/Unhide Port, live Visio menu option	198
Hiding	
Shortcuts, children of master objects	134
Hierarchical content table	64
Home page, Systems Architect/Requirements Management	33, 45
HTML note	
Creating	248
Editing	252
HTML, value of Text Format property	246
Identify Subshape, live Visio menu option	200
Image Height property	399
Image Scale property	399
Image Width property	399
Images, diagrams	202
Import AP233 dialog window	
Importing objects	153
Updating properties	290
Importing	
Keywords, import document, Word	171
Objects	
AP233 STP File	152
AP233 XML File	154
Excel	158
XML data file	156
Outline levels, import document, Word	171
Reference links, in imported objects	315
Requirements from Word	173
Include Subtypes check box, Search module.	356
Indicators	63, 90
Installing	
Systems Architect/Requirements Management client	33
Systems Architect/Requirements Management MATLAB interface	43
Installing Architect/Requirements Client with Office Integration	
Components	28
Introduction	27
Prerequisites	28
Privileges	29
Registry	29
Intermediate button, Search module	358
Intermediate search view	358
Internet Explorer	
Installing Systems Architect/Requirements Management client	33
Starting Systems Architect/Requirements Management	45
ISO 10303	138, 152, 290
Java, properties	103
Keep Previous Settings	
Search results output option	342
Keep Same Location check box, Search module	353, 358, 362
Keyboard operations	
Switching to content table	70
Switching to navigation tree	61
Switching to notebook pane	93
Keywords, import document, Word	171
Last, button, Data Definitions dialog window	228
Levels, object hierarchy	
Changing	184, 195
Creating	171
Building blocks	193
Requirements	169
Filtering	97
Viewing	95

Links	See Trace links
Links indicator	65, 91
Links tab	
Columns	
Adding and removing	272
Rearranging	274
Resizing.....	275
Sorting information.....	275
Description.....	77
Exporting to Excel	143
Hiding or showing panes	93
Navigating to linked objects	242
Relations subtab.....	80
Resizing panes	94
Trace subtab.....	77
Using in floating window.....	90
Viewing	
Defining and complying objects	238
Trace links.....	241
Live Excel interface	
Creating	
Objects	305
Trace links.....	306
Editing properties, independent session.....	293
Editing properties, session with Systems	
Architect/Requirements Management client	
.....	292
Excel template.....	143, 293
Overview.....	292
Spreadsheet	
Creating.....	295
Disconnecting from Systems	
Architect/Requirements Management	
database.....	307
Live Office Diagnosis	
Diagnose	42
Diagnostic Information	42
Email Report	42
Generate Report	42
Help.....	42
Live Office Interface Diagnostic Tool	42
Run.....	42
Suggested Fix.....	42
Live Office Interface Diagnostic Tool	42
Diagnose	42
Diagnostic Information	42
Email Report	42
Generate Report	42
Help.....	42
Run.....	42
Suggested Fix.....	42
Live Visio	
Hide/Unhide Connection Name option	227
Hide/Unhide Connection Names option.....	227
Hide/Unhide Marked Connection Names	
option.....	227
Reset Marked Connection Names option ..	227
Live Visio diagram	
Connectors	
Deleting	226
Live Visio interface	
Diagram type, changing.....	211
Diagrams	
Copying from Visio	209
Creating	201
Creating Trace Links, Copy to Diagram,	
and Paste to TcSE.....	231
Creating, editing, and deleting connections	
.....	223
Disconnecting from Systems	
Architect/Requirements Management	
database	213
Editing	204
Ports, creating	220
Viewing connections	231
Viewing, if Visio not installed.....	214
Overview	197
Stencil Diagnostic Utility	200
Log file, Architect/Requirements client	105
Logging in, Systems Architect/Requirements	
Management	33, 45, 292
Long, client debugging option.....	107
Look in field, Search module.....	353, 358, 362
Main window, Systems Architect/Requirements	
Management	48
Make Visio Diagram Non Live, live Visio menu	
option.....	198
Matching ROIN field, Search module	355
Matching ROIN field, wildcard characters....	355
MATLAB interface, installing.....	43
Member Count property . 95, 128, 129, 205, 206,	
399	
Members	
Adding to groups	128
E-mail, sending object data.....	135
Filtering in a hierarchy	97
Removing from groups.....	129
Restoring.....	163
Viewing in a hierarchy	95
Members diagram type, live Visio.....	211
MHTML Library:	38
MHTML note, Word	

Creating.....	250	Networking symbol	48
Editing.....	252	Next, button, Data Definitions dialog window	228
MHTML, value of Text Format property	246	Non-Live button, disconnecting live Excel file	307
Microsoft Internet ExplorerSee Internet Explorer		Notebook pane	
Microsoft Office Excel	See Excel	Applying a column view.....	268
Microsoft Office Visio.....	See Visio	Attachments tab	75
Microsoft Office Word	See Word	Columns	
Microsoft Word		Adding and removing	272
Styles		Rearranging	274
Direct formatting.....	176, 406	Resizing	275
Modify pane, Search module	365	Sorting information.....	275
Modifying		Connectivity tab.....	83
Change approval object.....	383	Disabling or enabling.....	93
Module bar		Hiding or showing	93
Disabling or enabling	57	Links tab	
Hiding or showing.....	57	Description.....	77
Overview.....	57	Relations subtab.....	80
Resizing	57	Trace subtab.....	77
Moving objects.....	123	Overview	72
MUI Pack for Office	38	Preview tab	85
Multi-Choice dialog window .279, 283, 286, 326		Properties tab	74
Multilingual User Interface (MUI) Pack.....	38	Resizing	94
Name column		Switching to.....	93
Content table	64	System Settings option	268
Versions tab	88	Using tabs in floating windows	90
Name conventions.....	18	Versions tab	88
Name property	399	Where Used tab	86
Navigating		Notes	
Linked objects		Adding to groups	128
In other Teamcenter products	242	Attaching notes to notes	246
In Systems Architect/Requirements		Attaching to objects	246
Management project.....	242	Copying	119
Modules, Systems Architect/Requirements		Creating	
Management.....	57	Plain text format	247
Objects, Architect/Requirements		Rich text format	248
Live Visio interface.....	204	Word.....	250
Objects, Systems Architect/Requirements		Deleting	162
Management		Diagram images (.gif).....	202
Excel	144, 300	Editing	
Live Visio interface.....	223	Plain text format	252
Search module.....	342	Rich text format	252
Shortcuts, master objects	134	Text property	257
Navigation tree		Word.....	252
Disabling or enabling	61	E-mail, sending object data.....	135
Expanding projects and folders.....	60	Enabling Text property editing.....	255
Hiding or showing.....	61	Exporting	
Overview.....	60	AP233 data file	138
Plus signs	60, 113	Word.....	146
Resizing	62	XML data file	142
Root node, disabling or enabling	61		
Switching to	61		

Importing		Ports	220
AP233 STP File	152	Requirements, by importing from Word	171
AP233 XML File	154	Requirements, in Architect/Requirements	
Excel	158	client views	168
XML data file	156	Requirements, subtypes	168
Overview	245	Requirements, top level of folder	169
Printing content	85, 254, 260	Shortcuts	133
Properties		Sibling of existing requirement	169
Editing	279, 283	Spreadsheets	308
System-defined, descriptions	396	Trace links	234, 237
Removing from groups	129	Data definitions	
Renaming	125	Attaching to connections	228
Restoring	163	Creating, in Architect/Requirements client	
Search module		217
Advanced search view	362	Creating, in diagram	218
Basic search view	353	Data dictionaries	
Intermediate search view	358	Attaching to diagrams	216
Search results	341	Creating	215
Subtypes	246	Defining	233
URLs, copying	122	Deleting	
Viewing content	259	Folders, requirements, building blocks,	
Notifier, change management user	380	groups, notes	162
Number property	400	Trace links	243
Building blocks	193, 194, 195	Diagrams	
Requirements	169, 170, 171, 185	Copying from Visio to live Visio	209
Numeric property		Creating	201
Definition	263	Editing	204
Value, changing		Editing	
Edit Properties dialog window	279	Notes, plain text format	252
Live Excel interface	298, 303	Notes, rich text format	252
Properties tab	279	Notes, Word	252
Table view cells	286	Exporting	150
Object template	146	AP233 data file	138
Object type indicator	64, 68, 75, 77, 86, 88	Excel	143
Objects		Word	146
Adding to groups	128	XML data file	142
Complying	233	Importing	
Connections		AP233 STP File	152
Creating, Architect/Requirements client	221	AP233 XML File	154
Creating, editing, and deleting	223	Excel	158
Viewing	231	XML data file	156
Copying	119	Linking	
Creating		To other Teamcenter products	237
Building blocks	192	With Generic links	244
Child of existing requirement	170	Within Systems Architect/Requirements	
Folders	115	Management	234
Groups	127	Moving	123
In live Excel	305	Properties	
Notes, plain text format	247	Calculating numeric values with formulas	
Notes, rich text format	248	288
Notes, Word	250	Definition	261

Editable	261
Editing	279, 283, 286, 292
Read-only	261
Reference links	309
Reference links, target objects	309
Reference links, types	309
System-defined, descriptions	396
Types	261
Updating, from AP 233 data file	290
Viewing	272
Release reservation	264
Removing from groups	129
Renaming	125
Restoring	163
Search module	
Advanced search view	362
Basic search view	353
Intermediate search view	358
Search results	341
Sending data by E-mail	135
Shortcuts	
Creating	133
Hiding and showing children of master objects	134
Navigating to master objects	134
Overview	131
Replacing with copies of master objects	135
Shortcuts, replacing with copies of master objects	135
Subtypes	174
Building blocks	192
Folders	115
Generic links	244
Groups	127
Notes	246
Requirements	168
Trace links	234
Trace Links	
Creating, Copy to Diagram, and Paste to TcSE	231
Types	23
Off, client debugging option	107
OID property	400
OLE content, exported requirements	147
On, client debugging option	107
Open button, live Excel interface	300
Open Child Diagram w/Stencil, live Visio menu option	199
Open Child Diagram, live Visio menu option	199
Open dialog window	153, 155, 157, 159, 205, 290, 308
Open in Word button, Create Note On dialog window	250
Open Read-only button, live Excel interface.	300
Opening	
Diagrams, if Visio not installed	214
Folders	113
Notebook pane tabs in floating windows	90
Notes	252, 259
Projects	112
Requirements	177, 186
Order	
Search Results	352
Organizing requirements in a hierarchy	184
Original Location property	400
Outline levels, import document, Word	171
Output Options section, Search module	357
Output options, Search module	341
Page setup, requirement content	177
Paragraph, requirement subtype	
Adding to groups	128
Creating	168, 174
Removing from groups	129
Parsing, import document, Word	
By outline levels and keywords	171
By outline levels only	171
Password	
Security Services	34, 46
Systems Architect/Requirements Management Changing	110
Logging in	34, 46
Paste and Create In TcSE, live Visio menu option	198
Paste as Tunnel Port, live Visio menu option	198
Paste From TcSE, live Visio menu option	198
Paste From Teamcenter for systems engineering, live Visio menu option	225, 232
Paste to TcSE	
Trace Links	
Live Visio diagram	232
Pending, Shared State property value	
Reports, Search module	344
Plain text note	
Creating	247
Editing	252
PLM XML	
Exporting objects	150
Plus signs	
Content table	64, 95, 113
Navigation tree	60, 113
Ports	
Connectivity tab	83

Creating.....	220	Read-only	
preamble, note, document import	171	Definition.....	261
Prerequisites for Using the Live Office Interface		System-defined, descriptions.....	396
.....	37	Reference links	
MHTML Library.....	38	Copying	313
Multilingual User Interface (MUI) Pack.....	38	Creating, full content	319
Preview tab		Creating, plain text.....	318
Description.....	85	Deleting	314, 323
Notes, viewing content.....	259	Exporting	315
Reference links, deleting.....	323	Importing	315
Reference links, viewing.....	321	Overview	309
Requirements, viewing content.....	186	Removing referenced property from type	
Using in floating window.....	90	definition.....	314
Previous, button, Data Definitions dialog		Target objects	309
window.....	228	Types	309
Printing		Versioning	315
Note content	85, 254, 260	Viewing	321
Requirement content	85, 149, 177, 186	System-defined	
Selected Panel	102	Definition.....	261
Private, Shared State property value		Descriptions	396
Reports, Search module	344	Systems Architect/Requirements Management	
Project property.....	400	system.....	103
Projects		Types	261
Expanding	60	User-defined, definition.....	263
Opening.....	112	Viewing	272
Overview.....	111	Properties tab	
Promoting		Columns	
Building blocks	195	Resizing	275
Requirements	185	Sorting information.....	275
Properties		Description.....	74
Columns		Exporting to Excel	143
Adding and removing	272	Filtering properties	277
Rearranging.....	274	Using in floating window	90
Resizing.....	275	Property columns	
Sorting.....	275	Views	
System default view	266, 267	Applying, content table.....	267
Views, applying to folders	265	Applying, notebook pane.....	268
Definition	261	Content table, conditions	266
Editable		Property editing, multiple object selections... 282	
Definition	261	Proxy Server Settings dialog window	
System-defined, descriptions	396	Starting new Architect/Requirements session	
Editing		46
Calculating numeric values with formulas		Public, Shared State property value	
.....	288	Reports, Search module.....	344
Edit Properties dialog window.....	279	Queries, Advanced search view.....	367
From AP 233 data file.....	290	Queries, statements.....	370
live Excel	292	Query Edit pane, Search module	366
Multiple object selections	283	Query View pane, Search module	364
Properties tab	279, 283	Read-only properties	
Table view cells	286	Definition.....	261
Filtering.....	277	System-defined, descriptions.....	396

Rearranging	
Columns	274
Toolbar buttons	55
Recalling	
Saved reports, Search module	345
Saved views	
Content table	267
Notebook pane	268
Recycle Bin	
Description	68
Emptying	165
Exporting to Excel	143
Restoring objects	163
Reference links	
Behavior	310
Copying	313
Creating	
Full content	319
Plain text	318
Deleting	314, 323
Exporting	315
Importing	315
Overview	309
Promote References	310
Reference Settings	310
Removing referenced property from type definition	314
Target objects	309
Types	309
Versioning	315
Viewing	321
Refresh button, Data Definitions dialog window	218, 228
Refresh button, live Excel interface	300
Refresh Entire Visio Diagram, live Visio menu option	198
Refresh With Connection, live Visio menu option	198
Registering	
Architect/Requirements Client and Microsoft Office	410
Rejected, change management event	380
Rejecting a change	381
Relations subtab, Links tab	
Deleting	
Reference links	323
Overview	80
Viewing	
Generic links	244
Reference links	321
Trace links	241
Relative searches, wildcard characters	355
REMOVE clause, queries	378
Removing	
Columns	272
Group members	129
Renaming objects	125
Replacing a shortcut with a copy of the master object	135
Report generation	See Advanced search view
Reports, Search module	
Exporting	
Excel	347
Visio	350
Word	349
Progress messages	345, 348, 349, 350
Running from Search module	345
Running from URL	344
Saving and recalling	344
Requirements	
Adding to groups	128
Baseline	
Creating	335
Viewing	337
Comparing content	
Comparison mode	187
Overview	187
Requirements in two folders	189
Two requirements	188
Connections	
Creating, editing, and deleting	223
Viewing	231
Copying	119
Creating	
By importing from Word	171
Child of existing requirement	170
In Architect/Requirements client views	168
In live Excel	305
Sibling of existing requirement	169
Subtypes	168
Top level of folder	169
Defining and complying objects, viewing	238
Deleting	162
Demoted, positions	184
Demoting	185
Diagrams	
Attaching	201
Editing	204
Viewing, if Visio not installed	214
Editing	
Word	175
Editing content	

Content elements.....	175	Creating	133
Formatting.....	175	Hiding and showing children of master	
Page setup	177	objects.....	134
Text property cells	179	Navigating to master objects	134
Word	177	Overview	131
E-mail, sending object data.....	135	Replacing with copies of master objects	135
Enabling Text property editing	182	Spreadsheets, attaching.....	308
Exporting		Subtypes.....	168, 174
AP233 data file	138	Trace links	
PLM XML	150	Creating	234, 237
Word	146	Deleting	243
XML data file.....	142	Navigating to linked objects	242
Freezing	329	Viewing	241
Importing		Trace Links	
AP233 STP File	152	Creating, Copy to Diagram, and Paste to	
AP233 XML File	154	TcSE	231
Excel	158	Unfreezing	330
Word	171	URLs, copying.....	122
XML data file.....	156	Versions and variants	
Keywords, import document, Word.....	171	Creating a variant.....	332
Linking		Creating a version.....	331
To other Teamcenter products	237	Deleting	334
Within Systems Architect/Requirements		Viewing	333
Management.....	234	Viewing content.....	186
Members, filtering.....	97	Requirements, ROIN	
Members, viewing.....	95	Search module	
Moving.....	123	Basic search view	353
Notes, attaching.....	246	Reserved By property	400
Number property.....	169, 170, 171, 185	Reset Marked Connection Names, live Visio	
Outline levels, import document, Word.....	171	option.....	227
Overview.....	167	Reset Marked Connection, live Visio menu	
Paragraph subtype.....	168, 174	option.....	198
Adding to groups.....	128	Resizing	
Removing from groups	129	Address bar	59
Printing content.....	85, 149, 177, 186	Columns.....	275
Promoted, positions.....	184	Content table.....	71
Promoting.....	185	Module bar.....	57
Properties		Navigation tree	62
Calculating numeric values with formulas		Notebook pane.....	94
.....	288	Response Time, client debugging option.....	107
Editing.....	279, 283, 286, 292	Response, change management event.....	380
System-defined, descriptions	396	Restoring objects	163
Updating, from AP 233 data file.....	290	Reverse Direction, live Visio menu option... 199,	
Removing from groups	129	224	
Renaming	125	Revision conventions.....	17
Restoring.....	163	Rich text note	
Search module		Creating	248
Advanced search view	362	Editing	252
Intermediate search view	358	ROIN property.....	400
Search results	341	Running	
Shortcuts		Saved reports, Search module.....	345

Search report, from URL	344	Activating	340
Save dialog window	139, 141, 142	Advanced search view	362
Save Properties dialog window	103	Basic search view	353
Save View dialog window	269	Intermediate search view	358
Saved Report Information dialog window	345	Order of Search Results	352
Saved views	102	Output Options	341
Applying		Overview	339
Content table	267	Progress messages	345, 348, 349, 350
Notebook pane	268	Reports	
Content table, conditions	266	Exporting to Excel	347
Modifying	272	Exporting to Visio	350
Saved views, content table		Exporting to Word	349
Creating	269	Running from Search module	345
Saved views, hierarchical content table		Running from URL	344
Modifying	276	Saving and recalling reports	344
Search results output option	341	Search results	341
Saving and recalling		Search Options section, Search module	356
Content table views	269	Search Results	
Search module reports	344	Order	352
Search feature, Data Definitions dialog window		Search Results dialog window	341, 342
.....	228	Search subordinates via query, Search module	
Search for Name field, Search module	364	356
Search For Name field, Search module	355	Search subordinates via relations, Search	
Search For Name field, wildcard characters	355	module	356
Search members only, Search module	356	Security Profile property	400
Search module		Security profile, applying to object	381
Case Sensitive check box	356	Security Services ...34, 45, See Security Services	
Containing text field	355, 364	Select a Baseline dialog window	337
Edit Search Criteria pane	358, 360	Select Document Template dialog window ...	149
Exclude Shortcuts check box	356	Comparing requirements in two folders ...	189
Include Subtypes check box	356	Comparing two requirements	188
Intermediate button	358	Select Non TcSE Shapes, live Visio menu	
Keep Same Location check box	353, 358, 362	option	198
Look in field	353, 358, 362	Select Object Chooser dialog window	237
Matching RION field	355	Select Properties pane, Search module	358, 360,
Modify pane	365	361	
Output Options section	357	Select Report dialog window	345
Query Edit pane	366	Select Saved Report dialog window	345
Query View pane	364	SELECT statement, queries	370
Search for Name field	364	Select Subtype dialog window	207
Search For Name field	355	Building blocks	193
Search members only	356	Data definitions	217
Search Options section	356	Data dictionaries	215
Search subordinates via query	356	Folders	115, 116, 117
Search subordinates via relations	356	Groups	128
Select Properties pane	358, 360, 361	Notes	250
Specify Criteria pane	358, 360	Requirements	169, 170
Types/Subtypes tree	364	Trace links	235
Types/SubTypes tree	355	Select User dialog window	136
Search module, Systems Architect/Requirements		Send Email dialog window	136
Management			

Sending E-mail from Systems	
Architect/Requirements Management client	
.....	135
Set Data Definition, live Visio menu option..	199
Set View As Default, View menu option.....	271
Setting	
Default view	
For folder	270
For user name.....	271
Sort column.....	275
Shapes, default arrangement in live Visio	
diagrams.....	202
Shapes, live Visio diagrams, adding and deleting	
.....	204
Shared State property	
Reports, Search module	344
Short, client debugging option	107
Shortcut Objects.....	163
Shortcuts	
Creating.....	133
Hiding and showing children of master	
objects	134
Navigating to master objects.....	134
Overview.....	131
Replacing	135
Showing	
Shortcuts, children of master objects	134
Showing and hiding filters	97
Single Sign-On (SSO)..... See Security Services	
Single-Choice dialog window 149, 279, 283, 286	
Comparing requirements in two folders.....	189
Comparing two requirements.....	188
Data dictionaries, attaching to diagrams ...	216
Diagram Content property	211
Text Format property	182, 255
SORT clause, queries.....	378
Sorting, property columns.....	275
Source Filename property	400
Source Paragraph property.....	400
Specify Criteria pane, Search module....	358, 360
Spreadsheets	
Attaching to objects	308
Creating.....	295
Disconnecting from Systems	
Architect/Requirements Management	
database.....	307
Start a connection, toolbar button	54, 221
Start Link, live Visio menu option.....	198
Starting Systems Architect/Requirements	
Management	
Excel	
Exported table view	144
Live interface.....	292
Home page	45
Microsoft Outlook	122
Object URL.....	122
Startup folder, Word	35
Static Date property	400
Static diagram type, live Visio.....	211
Stencil Diagnostic Utility	200
Stencils property	400
STEP (ISO 10303).....	138, 152, 290
Style sheet.....	146
Styles Vs Direct formatting	176, 406
Styles, Word	146
Imported requirements.....	173
Requirement content.....	175
Submit For Approval dialog window	381
Submit, change management event	379
Submitting changes for approval.....	380
Subtype property.....	400
Subtypes	
Building blocks.....	192
Folders	115
Groups	127
Notes.....	246
Requirements	168, 174
Search module	
Advanced search view	362
Basic search view	353
Intermediate search view	358
Search results	341
Trace links	234
Synergy Check-In Comment property	401
SynergyID property	401
SynergyTask property.....	401
System Information dialog window.....	103
Client Log tab	105
Logging Options tab	107
System Settings option, notebook pane	268
System Settings option, View list.....	267
System-defined properties	
Definition.....	261
Descriptions	396
Values, changing	279, 283, 286, 292
Values, updating from AP 233 data file ...	290
Viewing	272
Systems Architect/Requirements Management	
E-mail, sending from client	135
Exiting	110
Home page	33, 45
Installing client component.....	33

Live Excel interface	
Editing properties, independent session	293
Editing properties, session with Systems Architect/Requirements Management client	292
Objects, creating	305
Overview	292
Trace links, creating	306
Logging in	33, 45, 292
Login page	34, 45
Main window	48
MATLAB interface, installing	43
Password, changing	110
Recycle Bin	
Description	68
Emptying	165
Exporting to Excel	143
Restoring objects	163
Search module	
Activating	340
Advanced search view	362
Basic search view	353
Intermediate search view	358
Order of Search Results	352
Output Options	341
Overview	339
Reports, exporting to Excel	347
Reports, exporting to Visio	350
Reports, exporting to Word	349
Saving and recalling reports	344
Search results	341
Starting	
Excel, exported table view	144
Excel, live interface	292
Home page	45
Microsoft Outlook	122
Object URL	122
System information	103
Toolbar	
Displaying button groups in floating windows	55
Overview	51
Rearranging button groups	55
Tabs, notebook pane	
Attachments tab	75
Connectivity tab	83
Links tab	
Description	77
Relations subtab	80
Trace subtab	77
Preview tab	85
Properties tab	74
Using tabs in floating windows	90
Versions tab	88
Where Used tab	86
TBD button, date property calendar	280, 285, 287
TBD search criteria, Intermediate search view	360
TcSE Properties, live Visio menu option	198
TcSEPreprocessor.DOT file	173
TCSEPreprocessor.dot file	35
Teamcenter for systems engineering properties, dialog window	206
Template	
Document	146
Excel	143, 293, 347
Object	146
Text Format property	246, 401
Text property	401
Definition	263
Notes, editing	257
Notes, enabling for cell editing	255
Requirements	
Editing content	179
Requirements, enabling for cell editing	182
Value, changing	
Edit Properties dialog window	279
Live Excel interface	298, 303
Properties tab	279
Table view cells	286
Text, value of Text Format property	246
Today button, date property calendar	280, 285, 287
Toggle Column Filters, View menu option	97
Toggle Relationship dialog window	351
Toolbar buttons	
End a connection	54, 221
End a connection subtype	54, 221
Start a connection	54, 221
Toolbar, Systems Architect/Requirements Management	
Displaying button groups in floating windows	55
Overview	51
Rearranging button groups	55
Tools	
Release reservation	264
Tooltips	
Attachments tab	75
Complying Object Traceability window	240
Connectivity tab	83

Defining Object Traceability window.....	239	Search reports, running.....	344
Hierarchical content table	64	Sending by E-mail	136
Overlapping Preview tab.....	85	Usage	
Search Results dialog window	341, 342	Child Object.....	163
Trace subtab, Links tab	78	Shortcut Objects	163
Versions tab	88	Use Rich Text button, Create Note On dialog	
Trace Link Count property.....	65, 91, 401	window	248
Trace links		User name	
Creating		Default view, setting.....	271
In live Excel	306	Security Services	34, 46
To other Teamcenter products	237	Systems Architect/Requirements Management	
Within Systems Architect/Requirements		34, 46
Management.....	234	User-defined properties	
Deleting.....	243	Definition.....	263
Navigating to linked objects	242	Values, changing	279, 283, 286, 292
Overview.....	233	Values, numeric, calculating with formulas	
Properties		288
System-defined, descriptions	396	Values, updating from AP 233 data file	290
Restoring.....	163	Viewing	272
Subtypes.....	234	Using groups to maintain objects	126
Trace Links		Using OLE Objects in Word.....	178
Creating, Copy to Diagram, and Paste to		Using tabs in floating windows	90
TcSE.....	231	Using the live Visio interface	197
Trace subtab, Links tab		Value conventions	18
Overview.....	77	Values, object properties	
Viewing		Calculating numeric values with formulas	288
Defining and complying objects	238	Changing.....	279, 283, 286, 292
Trace, client debugging option.....	107	System-defined, descriptions.....	396
TRAM, building block subtype		Updating from AP 233 data file.....	290
Adding to groups.....	128	Variants	
Creating.....	192	Baseline	
Overview.....	191	Creating	335
Removing from groups	129	Viewing	337
Transitional mapping See TRAM, building block		Creating	332
subtype		Deleting	334
Type Name property	401	Renaming.....	125
Types		Viewing	333
Object.....	23	Verify Prop Map XML And Stencil, live Visio	
Property.....	261	menu option	200
Types/Subtypes tree, Search module	364	Version As Of Date, effectivity rule.....	328
Types/SubTypes tree, Search module.....	355	Version Count property	66, 91, 401
Unassign Data Definition, live Visio menu		Version Number property	402
option	199	Version Type property	402
Unfreezing objects	330	Versions	
Unregistering		Baseline	
Architect/Requirements Client and Microsoft		Creating	335
Office	410	Viewing	337
Updating properties from an AP 233 data file	290	Choosing an effectivity rule.....	328
Upgrade Diagram, live Visio menu option	198	Creating	331
URLs of objects		Deleting	334
Copying.....	122	Enabling for project	326

Freezing requirements, building blocks	329	Search results output option.....	341
Overview	325	Visio	
Reference links in	315	Copying diagrams to live Visio	209
Renaming	125	Diagrams, live Visio	
Unfreezing requirements, building blocks	330	Connections, viewing	231
Viewing.....	333	Diagram type, changing.....	211
Versions indicator	66, 91	Editing	204
Versions tab		Viewing, if Visio not installed.....	214
Columns		Diagrams, live Visio interface	
Adding and removing	272	Connections, creating, editing, and deleting	
Rearranging.....	274	223
Resizing.....	275	Creating	201
Sorting information.....	275	Shapes, default arrangement.....	202
Description.....	88	Trace Links, Creating, Copy to Diagram,	
Exporting to Excel	143	and Paste to TcSE.....	231
Using in floating window.....	90	Exporting reports, Search module	350
View list		Search results output option.....	342
Applying a column view	267	Using the live Visio interface	197
Default View option.....	267	Web browser	
System Settings option.....	267	Installing Systems Architect/Requirements	
Viewing		Management client.....	33
Baselines	337	Starting Systems Architect/Requirements	
Change approval object.....	384	Management	
Change log	386	Home page	45
Connections.....	231	Object URL.....	122
Defining and complying objects	238	WHERE clause, queries.....	371
Diagrams, if Visio not installed	214	Where Used tab	
Notes, content	259	Columns	
Objects in a hierarchy	95	Adding and removing	272
Properties	272	Rearranging	274
Reference links.....	321	Resizing	275
Requirement content	186	Sorting information.....	275
System information, Systems		Description.....	86
Architect/Requirements Management....	103	Exporting to Excel	143
Trace links.....	241	Using in floating window	90
Variants	333	WhereUsed Object Count property	402
Versions	333	Wildcard characters, relative searches.....	355
Views		Word	
Applying		Exporting objects	146
Content table	267	Exporting reports, Search module	349
Notebook pane	268	Importing requirements	173
Content table, conditions	266	Keywords, import document	171
Creating.....	269	Notes	
Default		Creating	250
Setting for folder	270	Editing	252
Setting for user name	271	OLE content, exported requirements	147
Saved.....	102	Outline levels, import document.....	171
Views, applying to folders	265	Requirements	
Views, content table		Content elements	175
Saving and recalling.....	269	Creating	171
Views, hierarchical content table		Editing	175

Editing content.....	177	Requirement data.....	146
Formatting.....	175	TCSEPreprocessor.dot file	35
Page setup	177	Viewing	
Using OLE Objects in Word.....	178	Notes, content	259
Search results output option.....	341	Requirement content.....	186
Startup folder	35	XML data format	
Styles		Exporting project data	142
Imported requirements	173	Importing project data	156
Requirement content	175		

Teamcenter 11.1 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management Project Administrator's Manual

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	3.0	March 2003
B	3.0.1	May 2003
C	4.0	December 2003
D	4.1	February 2004
E	5.0	July 2004
F	6.0	March 2005

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
G	2005	September 2005
H	2005 SR1	June 2006
I	2007	December 2006
J	2007.1	April 2007
K	2007.2	September 2007
L	2007.3	January 2008
M	8	January 2009
N	8.0.1	June 2009
O	8.1	October 2009
P	8.2	October 2010
Q	9	July 2011
R	9.1	May 2012
R1	9.1.5	January 2014

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	10.0	January 2015
T	10.1	September 2016
U	11.1	March 2018

This edition obsoletes all previous editions.

Project Administrator's Manual Contents

Manual History	4
Contents	7
Preface.....	11
Audience	11
Conventions	11
Revision Marks	11
Browser and Dialog Window Examples.....	11
Names and Values	12
Submitting Comments.....	12
Proprietary and Restricted Rights Notice.....	12
Chapter 1: Introduction to Project Administration	15
Overview of Projects.....	15
Schema Objects.....	16
Administration Folders	21
Custom Menus	26
Notebook Pane.....	30
Creating a Project.....	34
Importing a Project.....	35
Exporting Project Data.....	37
Importing Schema Objects.....	39
Running an Import From the Command Line.....	41
Running an Export From the Command Line.....	43
Managing Rule File to Export or Import Objects	45
Using the Rule File	45
Identifying the Rule for Objects	45
Identifying the Rule for Properties of Objects	46
Updating Objects During Import	47
Architect/Requirements XML Format	49
Types of XML Files.....	49
Types of Data Elements.....	50
Schema XML Format	52
Order of Objects.....	52
Removing References to a Schema Object	52
Deleting a Schema Object.....	58

Deleting a Project.....	60
Chapter 2: Customizing Object Properties	61
Overview of Property Definitions.....	61
Creating a Property Definition.....	63
Modifying a Choice Property Definition	64
Setting a Dynamic Choice List for a Choice Property Definition.....	69
Modifying a Date Property Definition	73
Modifying a Numeric Property Definition.....	74
Modifying a Text Property Definition	76
Updates to Change Time and Change User Properties	79
Chapter 3: Customizing Object Types.....	81
Overview of Type Definitions	81
Creating a Subtype.....	83
Modifying the Properties of a Type Definition.....	84
Customizing the Object Type Indicator for a Type Definition	86
Setting the Default View for a Folder Type Definition	87
Customizing the ROIN for a Requirement Type Definition	88
Chapter 4: Customizing the Interface With Microsoft Office	91
Templates and Style Sheets for Export to Microsoft Office Word.....	91
Document Templates	91
Object Templates	96
Style Sheets.....	100
Templates for Export to Microsoft Office Excel	103
Creating an Excel Template Object	103
Modifying an Excel Template	104
Packing Multiple Objects on One Row	106
Excel Template Modification Concepts.....	107
Activator Tag Results	108
Rule Table Concepts.....	110
Date Style Support for Export to Microsoft Office Excel.....	115
Stencils for Microsoft Office Visio.....	115
Configuring a Mapping File	116
Creating a Stencil Object	125
Modifying a Stencil Object.....	126
Viewing Diagram Links.....	128
Guidelines for Working In Live Visio Stencils	129
Chapter 5: Managing Users	133
Overview of Users	133
Project Access.....	133
Special Privileges.....	134
Power Users	135
Licensing.....	137
Creating a New User	144

Granting or Revoking Project Access	145
Modifying a User Profile	146
Resetting a User Password.....	149
Creating a User Group	150
Modifying a User Group	151
Deactivating a User.....	153
Emptying a User's Recycle Bin.....	154
Chapter 6: Maintaining Project Security	155
Introduction.....	155
Security Profiles and Access Control.....	155
Inherited Security.....	157
Security Profile Support for Schema Objects	158
Access Control for Object Properties	158
Creating a Security Profile.....	159
Setting User Permissions	160
Setting Access Control Inheritance.....	162
Assigning a Default Security Profile to New Objects of a Type	163
Applying a Security Profile to a Schema Object	164
Chapter 7: Configuring the Change Management Package	165
Change Approval	165
Enabling the Versions Package for the Project.....	169
Enabling Effectivity-Sensitive References for a Project.....	170
Setting Up the Change Approval Process.....	171
Change Logs	174
Enabling Change Logs for the Project.....	175
Setting Up Change Logs for a Type Definition.....	175
Change Event Flags	176
Transaction Management.....	177
Appendix A: Glossary.....	181
Appendix B: System-Defined Properties in the Administration Module.....	185
Overview of System-Defined Properties	185
Table of System-Defined Properties	186
Appendix C: Date Style Generator Utility.....	197
Appendix D: Project Package Property.....	199
Appendix E: Controlling Requirement Content Changes Through IBM Synergy	201
Overview of the Architect/Requirements Interface With Synergy	201
Synergy Package Additions to Architect/Requirements Project Schema	202
Enabling the Synergy Package for an Architect/Requirements Project.....	204
Adding Requirements to a Synergy Project.....	205
Checking In Requirements to Synergy	207
Checking Out Requirements From Synergy	208
Synergy Interface Customization.....	210

Program Flow	210
Customization Points	210
New Applications and Interfaces	211
createAction RunJava	211
ClientJavaAPI Class	211
Icon Overlays	213
Java Development Environment	214
Index.....	216

Preface

This manual is a project administrator's reference for Teamcenter Systems Architect/Requirements Management 11.1. Systems Architect/Requirements Management belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual is for Systems Architect/Requirements Management (Architect/Requirements) project administrators. The manual provides both conceptual information and step-by-step instructions for specific tasks.

This manual assumes that you are familiar with your project and your product development process, that you understand general computer terminology and the Microsoft Windows operating system, and that you have experience with Microsoft Word, Microsoft Excel, and Microsoft Office Visio.



Project Administrator privilege is required for all procedures in this manual.

Conventions

This manual uses the conventions described in the following sections:

Revision Marks

Technical changes are marked by a bar adjacent to the changed text.

Browser and Dialog Window Examples

The examples of browsers and dialog windows in this manual may appear different from those you see on your screen:

- The examples reflect Systems Architect/Requirements Management as initially installed at your site. Your enterprise may customize the browsers and dialog windows such that they appear different from those in the examples.
- The examples reflect individual Systems Architect/Requirements Management modules. If you install additional modules, your dialog windows and browsers reflect the additional modules.

- The examples reflect Systems Architect/Requirements Management installed on a Windows platform.

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **initsid.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
<i>Italic</i>	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In initsid.ora , <i>sid</i> identifies a varying portion of the name (a unique system ID). For example, the name of the file might be: initBlue5.ora
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

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

© 2018 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Chapter 1: Introduction to Project Administration

This chapter presents an overview of projects in Architect/Requirements, describes the types of schema objects used in the Administration module, and contains instructions for creating and importing projects, for exporting and importing project data, and for deleting schema objects and entire projects.



Enterprise Administrator privilege is required for creating a project and importing a project.
Project Administrator privilege is required for all other procedures in this manual.

Overview of Projects

In Architect/Requirements, a project defines logical boundaries for the following:

- Access privileges of individual users

Access privileges determine which users can modify the information in that project, and which can only view the information. Each user can be granted access to any number of projects, with different permissions for each project.

Users can be granted project access privileges at several broad levels, with corresponding restrictions on all user actions within each project. This is in addition to the permissions imposed through the security profile settings on individual objects within a project.

- Customization of object properties and object types

At project scope, an Architect/Requirements project administrator defines custom object properties, property values, and object subtypes. Those properties, values, and subtypes apply only within a single project. Therefore, each project can be customized for a particular purpose, process, or product.

Schema Objects

In the Architect/Requirements Administration module, project administrators work with the following types of schema objects:

- *Property definitions*
- *Type definitions*
- *Templates and style sheets*
- *Reports*
- *Views*
- *Users and user groups*
- *Security profiles*
- *Activators*

Property Definitions

A project administrator can create custom property definitions to tailor Architect/Requirements for an organization's requirements management process, product types, and business needs. Each property is defined once, and can then be applied to as many different object types as desired.

For each property, the property definition determines the following:

- The property type (*choice, date, numeric, or text*)
- The default value for that property to be assigned to newly created objects.

For choice properties, the property definition also determines the list of valid choices, and whether the list should have single-choice or multiple-choice behavior.

Property definitions are defined within the scope of a project, and can be applied only to objects within that project. Different projects may have properties with the same names, but the properties are entirely independent.

Type Definitions

A type definition specifies the properties that apply to all objects of a given type. In turn, these properties determine the nature and behavior of the related objects.

Type definitions also allow the project administrator to define custom object types, which are subtypes based on the built-in object types. For example, it may be useful to define several subtypes of requirements, each with unique identifying properties:

- Customer Requirement — CustomerID property
- Mil Spec — Requirements extracted from DoD specifications, with a DoD Spec Number property
- Company Policy — Standard Procedure Document Number property

Type definitions are defined within the scope of a project and can only be applied to objects within that project. Different projects may have object types with the same names, but they are entirely independent of each other.

Templates and Style Sheets

When users export object data from the Systems Engineering and Requirements Management module to Microsoft Word, Architect/Requirements references a *document template* to generate the export document. In turn, the document template supplies instructions through the *object templates* and the *style sheet* that are associated with the document template.

A document template, two object templates, and a style sheet are created automatically for every new project. In the object templates, *tags* represent the object properties whose values are extracted to the export document for each object that the user specifies. The style sheet determines the Microsoft Word formatting that is applied to the data in the export document.

The project administrator can modify the built-in document template, object templates, and style sheet to customize the export process. In addition, the project administrator can create custom document templates, object templates, and style sheets according to specific project needs.

Reports

Architect/Requirements provides built-in reports through the optional **Standard Reports** package, which the project administrator can add to any project. In addition, users can create and save reports based on search criteria entered in the Search module.

Standard and custom reports are project specific. All reports for a project are stored in the **Reports and Formatting** folder in the Administration module. For more information about using the Search module and reports, see the *Systems Architect/Requirements Management User's Manual*.

The following reports are available in the **Standard Reports** package:

- - **Orphan Requirements** shows requirements that have no defining trace links. The search starts at the selected project node, folder, or requirement. All lower level requirements that match the search criteria are shown in the Search Results dialog window.
 - **Requirement Approval Status** shows requirements whose **Approval Status** property value is **Change Approved**. The search starts at the selected project node, folder, or requirement. All lower level requirements that match the search criteria are shown in the Search Results dialog window.

The **Change Approved** value is set when the change approval package receives an **Approved** event for the submitted requirement. For more information, see [Change Approval](#), in chapter *Configuring the Change Management Package*.
 - **Show Impact Downstream** shows all objects in the complying path from the selected object. These complying objects are affected if the selected object is changed. This report is output to the Search Results dialog window. If the project node is selected, the report shows only the project node.
 - **Show Impact Upstream** shows all objects in the defining path to the selected object. These defining objects are affected if the selected object is changed. This report is output to the Search Results dialog window. If the project node is selected, the report shows only the project node.
 -

Trace Report Deep shows complying trace links for all objects at all levels in the selected folder. This report is output to the Search Results dialog window.

•

Trace Report Deep Table Format shows complying trace links for all objects at all levels in the selected folder. This report is output to a Microsoft Excel spreadsheet.

•

Unallocated Requirements shows requirements that have no complying trace links. The search starts at the selected project node, folder, or requirement. All lower level requirements that match the search criteria are shown in the Search Results dialog window.

To add the standard reports to a project:

1. In the navigation tree, select the **Projects** node.
2. In the content table, select the project.
3. In the **Properties** tab or window, double-click the **Packages** property value to display the Multi-Choice dialog window.
4. Check the **Standard Reports** checkbox and click **OK** to close the dialog window.

For each standard and custom report, the **Shared State** property value determines the report's visibility among the users in the project:

Private The report is visible only to the creator, and only that user can modify the report. The report is not available to any other user. **Private** is the default value for each new report.



Private also allows the report to be disabled temporarily, for example, during modifications.

Pending The report is marked to be made visible to all users. The creator can set the **Pending** value. If appropriate, a project administrator can make the report visible by setting the **Public** value.

Public The report is visible to all users. The **Public** value can be set by any user who has **Project Administrator** privilege for the project.

A public report can be modified or deleted only by a project administrator.

Views

In the Systems Engineering and Requirements Management module, users can customize the display of property columns in the hierarchical content table. A unique set of columns, or *view*, can be applied to each project node or folder selected in the navigation tree. Views can be saved in the database and can be applied at any time.

All views are project specific and are stored in the **Reports and Formatting** folder. For each view, the **Shared State** property value determines the view's visibility among the users in the project:

Private The view is visible only to the creator, and only that user can modify the view. The view is not available to any other user. **Private** is the default value for each new view.



Private also allows the view to be disabled temporarily, for example, during modifications.

Pending The view is marked to be made visible to all users. The creator can set the **Pending** value. If appropriate, a project administrator can make the view visible by setting the **Public** value.

Public The view is visible to all users. The **Public** value can be set by any user who has **Project Administrator** privilege for the project.

A public view can be modified or deleted only by a project administrator. If a deleted public view is the default view for one or more users, the **Default View** property for each user is automatically set to a blank value.

Private, public, and pending views can be exported from the Administration module as can other schema objects. However, only public views can be imported. For more information, see [Exporting Project Data](#), [Importing a Project](#), and [Importing Schema Objects](#), later in this chapter.

Saved views are also available in the Basic pane of the Search module. For each search that you output to the Search Results dialog window, you can specify the column settings by selecting a view in the field at the bottom of the **Output Options** section. For more information about search results, see the *Systems Architect/Requirements Management User's Manual*.

Users and User Groups

Every person who uses Architect/Requirements must be represented by a *user* object in the database. A *maximum privilege level* assigned to each user limits that user's access to all projects. No user can be granted access to any project at a level that is higher than that user's maximum privilege level. The maximum privilege level also determines the type of Architect/Requirements license that is consumed for the user.

The project administrator can create *user groups* consisting of shortcuts to existing user objects in the project. Users represented by the shortcuts can be managed directly from the user group and changes are applied to the actual user objects. User groups can be added to a security profile to control user access for objects to which the security profile applies.

Security Profiles

A security profile is a set of access rules that control user actions on specific objects in a project. Every object can be associated with a security profile, and a given security profile can be applied to any number of objects.

Each security profile lists the users and user groups that have access to the objects to which the security profile applies. Within that list, individual users and user groups are assigned *permission* to view, modify, or delete the objects.

Activators

Activators are used to program automated processes that execute when certain events occur as a result of user actions in the Architect/Requirements client. Activators contain executable code written in Tool Command Language (Tcl). When an activator is triggered, that Tcl script is executed. For more information about using activators, see the *Systems Architect/Requirements Management API Reference* manual.



To open an activator script for editing or viewing, a user must have one of the following:

- The maximum privilege level of **Enterprise Administrator**.
- **Project Administrator** access privilege for the project that contains the activator.
- **Script Authoring** privilege for the project that contains the activator.

The same conditions must be met to view the **Script** property value of an activator. For more information, see [Project Access](#) and [Special Privileges](#) in chapter 5, *Managing Users*.

Administration Folders

In the Administration module, certain system defined folders are generated automatically for every new project. These system defined folders reside at the project's primary level, directly below the project node, and contain the schema objects for the project. The primary folders are **Activators**, **Property Definitions**, **Reports and Formatting**, **Security Profiles**, **Type Definitions**, and **Users**. For more information, see [Schema Objects](#), earlier in this chapter.

The primary folders cannot be renamed, moved, or deleted. However, you can use subordinate folders to arrange schema objects according to your preference. Within any primary folder except the **Type Definitions** folder, you can do the following:

- Create subfolders, in which you can organize schema objects in hierarchies. For more information, see [Creating a Subfolder](#), later in this chapter.
 -  You cannot create subfolders in the **Type Definitions** folder because the object types and subtypes are structured in a system defined hierarchy.
- Modify folders by doing the following:
 - Move existing schema objects from their primary folder to one or more subfolders. You can also move objects from one subfolder to another and from subfolders to the primary folder.
 - Move one or more subfolders within their containing primary folder.
 - Rename subfolders.

For more information, see [Modifying Administration Folders](#), later in this chapter.

- Control access to primary folders and subfolders through security profiles. For more information, see [Applying a Security Profile to an Administration Folder](#), later in this chapter.
- Export and import project data from and to subfolders. For more information, see [Exporting Project Data](#) and [Importing Schema Objects](#), later in this chapter.
- Delete subfolders. For more information, see [Deleting Subfolders](#), later in this chapter.

Creating a Subfolder

You can create subfolders at the top level of any primary folder except the **Type Definitions** folder. Also, you can create subfolders in other subfolders to extend the hierarchy of schema objects to lower levels. You cannot create subfolders directly below the project node.

After you create a subfolder, you can rearrange existing schema objects by moving them to the new subfolder. You can also create new schema objects in the new subfolder, of the types appropriate to the primary folder. For more information, see [Modifying Administration Folders](#), later in this chapter; [Creating a Property Definition](#) in chapter 3, *Customizing Object Types*; [Creating a Document Template](#), [Creating an Object Template](#), [Creating a Style Sheet](#), [Creating an Excel Template Object](#), and [Creating a Stencil Object](#) in chapter 4, *Customizing the Interface With Microsoft Office*; [Creating a New User](#) and [Creating a User Group](#) in chapter 5, *Managing Users*; and [Creating a Security Profile](#) in chapter 6, *Maintaining Project Security*.



All subfolders receive the **Admin Folder** object type. You cannot change that type, nor can you create subtypes for administration folders.

To create a subfolder:

1. Select the parent folder in the navigation tree or in the content table.
Do not select the project in the navigation tree or a primary folder in the content table.
2. Do one of the following:
 - In the navigation tree, pull down the **File** menu and choose the **New→Folder** options.
 - In the content table:
 - To create a sibling of an existing subfolder within the parent folder, select the sibling, and then pull down the **File** menu and choose the **New→Folder** options.
 - To create a child of the parent folder, pull down the **File** menu and choose the **New→Child Folder** options.

The content table displays the new folder with the default name in an open text field.

3. Enter the folder name in the text field, and then press the enter key.
You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Procedure Notes

Step 1: To see lower level folders, click the plus signs, or double-click a folder with a plus sign. In the navigation tree, you can also select a folder and then pull down the **View** menu and choose **Expand**, or right-click the folder and choose **Expand** from the popup menu. In the content table, you can also select a folder and then pull down the **View** menu and choose **Expand All**, or right-click the folder and choose **Expand All** from the popup menu.

Step 2: You can also right-click the parent or sibling and choose the options from the popup menu. Or, click the **Create New Folder** or **Create New Child** button on the toolbar. In the navigation tree, you can also press control-L. In the content table, you can also press control-L for a new sibling or control-K for a new child.

Modifying Administration Folders

Administration folders allow you to create hierarchical structures for the schema objects in a project. In every primary folder except the **Type Definitions** folder, you can distribute schema objects among subfolders at multiple levels.

Schema objects can be moved from their primary folder to subfolders, from one subfolder to another, and from subfolders to the primary folder. In addition, subfolders can be moved and renamed.



- Schema objects and subfolders cannot be moved between primary folders or to the project's primary level, directly below the project node.
- Primary folders cannot be moved or renamed.
- Administration folders cannot be copied. Schema objects cannot be copied, except that user objects can be added to user groups through the copy function.

To move schema objects or subfolders:

1. Select the objects or subfolders, and then pull down the **Edit** menu and choose **Cut**.



You can also use the mouse to drop the selection on the destination folder and complete this procedure.

2. Select the destination folder in the navigation tree or the content table, and then pull down the **Edit** menu and choose **Paste**.



If the destination is a primary folder, you must select it in the navigation tree.

Procedure Notes

Step 1: In the navigation tree, you can select only one subfolder. In the content table, you can select multiple objects, including subfolders. You can also right-click the selection and choose **Cut** from the popup menu, click the **Cut** button on the toolbar, or press control-X.

Step 2: You can also right-click the selection and choose **Paste** from the popup menu. Or, click the **Paste** button on the toolbar or press control-V. To reverse this action, you can pull down the **Edit** menu and choose **Undo Move**, click the **Undo** button on the toolbar, or press control-Z.

To rename a subfolder:

1. Select the subfolder in the navigation tree or the content table, and then pull down the **File** menu and choose **Rename**.
2. Enter the new name in the open text field, and then press the enter key.

Procedure Notes

Step 1: You can also right-click the subfolder and choose **Rename** from the popup menu. Or, press the F2 key.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Applying a Security Profile to an Administration Folder

By default, users must have **Project Administrator** privilege for a project to modify the contents of an administration folder. However, all users in the project can view the contents of a folder.

Without defining additional project administrators, you can use security profiles to allow certain users to modify the objects in administration folders. You can also specify the users who can view the contents of a folder. For more information, see chapter 6, [Maintaining Project Security](#).

You can apply a security profile to any administration folder, including a primary folder.

To apply a security profile to an administration folder:

1. Select the folder in the content table.
The **Properties** tab or window displays all viewable properties for the folder.
2.
In the **Value** column, double-click the **Security Profile** property value to display the Single-Choice dialog window.
3. Check the checkbox for the security profile, and then click **OK** to close the dialog window and set the property value.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Deleting Subfolders

A subfolder cannot be deleted unless it is empty. First, you must move or delete all objects from each subfolder that you intend to delete. For more information, see [Modifying Administration Folders](#), earlier in this chapter, and [Deleting a Schema Object](#), later in this chapter.

To delete subfolders:

1. Select the subfolders.

In the navigation tree, you can select only one subfolder. In the content table, you can select one subfolder or a group of nonadjacent or adjoining subfolders.

2. Pull down the **File** menu and choose **Delete**.

A confirmation message asks if you are sure you want to delete the selected objects.

3. Click **Yes** to remove the selection from the Architect/Requirements database.



This action clears the queue for the **Undo** option and cannot be reversed.

Procedure Notes

Step 2: You can also right-click the folder and choose **Delete** from the popup menu. Or, click the **Delete** button on the toolbar, or press the delete key.

Custom Menus

A system defined administration folder subtype, **Menu Folder**, lets you define project-specific menus and submenus. Using a system defined activator subtype, **Menu Item**, you can add options to main menus and submenus.

A given menu can be set for display in the Systems Engineering and Requirements Management module, the Search module, and the Administration module. Custom main menus are displayed between the **Tools** menu and the **Help** menu, in alphabetical order from left to right. Menu items and submenus are arranged alphabetically from top to bottom.



Custom menus and menu options cannot be internationalized.

Creating a Menu

Menu folders for main custom menus reside at the top level of the **Activators** folder. Submenus can be added by nesting other menu folders within a menu folder.

To create a menu:

1. Do one of the following:
 - For a main menu, select the **Activators** folder in the navigation tree.
 - For a submenu, select the parent menu folder in the navigation tree.
2. Pull down the **File** menu and choose the **New**→**Menu Folder** options.

The content table displays the menu folder with the default name in an open text field.

3. Enter the menu folder name in the text field, and then press the enter key.



The displayed menu name is set by the **Menu Text** property of the menu folder object. If this value is blank, the name of the menu folder object is used for the menu. For more information, see [Assigning a Display Name to a Menu or Menu Item](#), later in this chapter.

Procedure Notes

Step 2: You can also right-click the **Activators** folder or parent menu folder and choose the **New**→**Menu Folder** options from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Creating a Menu Item

You use a system defined activator subtype, **Menu Item**, to add individual options to a custom menu. For each menu item, you use Tool Command Language (Tcl) to write the script that executes the command. For information about editing activators, see the *Systems Architect/Requirements Management API Reference* manual.



To create menu items and edit their activator scripts, you must have **Script Authoring** privilege for the project. For more information, see [Project Access](#), [Special Privileges](#), and [Licensing](#) in chapter 5, [Managing Users](#).

To create a menu item:

1. In the navigation tree or the content table, select the parent menu folder.
2. Pull down the **File** menu and choose the **New**→**Activator Subtype**→**Menu Item** options.
The content table displays the item with the default name in an open text field.
3. Enter the menu item name in the text field, and then press the enter key.



The item name displayed on the menu is set by the **Menu Text** property of the menu item object. If this value is blank, the name of the menu item object is used for the item on the menu. For more information, see [Assigning a Display Name to a Menu or Menu Item](#), later in this chapter.

Procedure Notes

Step 2: In the navigation tree, you can also right-click the parent menu folder and choose the **New**→**Activator Subtype**→**Menu Item** options from the popup menu. In the content table, you can also right-click the parent menu folder and choose the **New**→**Menu Item** options from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Assigning a Display Name to a Menu or Menu Item

For each menu folder object and menu item object, the **Menu Text** property value determines the text that is displayed as the name of the menu or item. If the **Menu Text** value is blank, the object name is used for the name of the menu or item. You can assign a display name that is different than the object name.

To assign a display name to a menu or menu item:

1. Within the **Activators** folder, select the menu folder or menu item.
You can select a menu folder in the navigation tree or the content table. You can select a menu item only in the content table.
2. To display the Edit Properties dialog window, pull down the **File** menu and choose **Properties**.
You can use the **Properties** tab or window if your selection is in the content table.
3. In the **Value** column, double-click the **Menu Text** value to open a text field.
4. Enter the text to display as the name, and then press the enter key.

Setting the Visibility of a Menu or Menu Item

A system defined property, **Shared State**, determines whether a menu or menu item is visible to the users in the project.

To set the visibility of a menu or menu item:

1. Within the **Activators** folder, select the menu folder or menu item.
You can select a menu folder in the navigation tree or the content table. You can select a menu item only in the content table.
2. To display the Edit Properties dialog window, pull down the **File** menu and choose **Properties**.
You can use the **Properties** tab or window if your selection is in the content table.
3.
In the **Value** column, double-click the **Shared State** property value to display the Single-Choice dialog window.
4. Check one of the following checkboxes:

Private The menu or item is visible only to the creator. For any other user, the menu or item is not displayed. **Private** is the default value for each new menu and item.



Private allows the user to disable the menu or item temporarily, for example, while completing the script that runs a menu item.

Pending The menu or item is marked to be made public by a project administrator. The creator can set the value to **Pending**.

Public The menu or item is visible to all users in the project. Any user who has **Project Administrator** privilege can change the value to **Public**.



If a menu contains no public menu items, the entire menu is visible only to its creator. A menu is not displayed if it does not contain any menu items.

Filtering Menus by Privilege

A menu command may not work for a particular user if the user does not have the privilege for the object that the command accesses. For example, a user who does not have Architect privilege cannot use a command that modifies a building block. To avoid confusion, these commands can be removed from the user's custom menu structure. Setting the **Required Privilege** property on a menu folder restricts that menu to users that have the specified privileges. The choices in **Required Privilege** match the choices in the **Addition Privilege** property. When **Required Privilege** is set on a menu folder, that menu does not appear for users that do not have the corresponding privileges set in their user object's **Additional Privilege** property. If no privilege is set then all users have access to the menu.



The **Required Privilege** property only exists on menu folders. It is only possible to filter out an entire menu or sub-menu; individual commands cannot be filtered.

Filtering Menus with a Security Profile

It is possible to control which users see a particular menu or menu command. This is done by setting the **Security Profile** property on the corresponding menu folder or menu item. If a security profile is set, a user must have at least read access in the security profile, otherwise the menu or command does not appear. If no security profile is set, then all users have access to the menu or command. When filtering menus, security profiles are processed differently than when checking for other types of access. Project administrators have access to objects in a project even if the object has a security profile where they are not included. When filtering menus, you must include a user explicitly in the security profile else the user does not see the menu. This means menu filtering applies to administrators.



To grant all users access to a menu or command that has a security profile, set the **Everyone** choice in the security profile's **Read Access** property.

Setting the Module Display for a Menu

A given custom menu can be displayed in or removed from the Systems Engineering and Requirements Management module, the Search module, and the Administration module.

To set the module display for a menu:

1. Within the **Activators** folder, select the menu folder in the navigation tree or the content table.
2. To display the Edit Properties dialog window, pull down the **File** menu and choose **Properties**.
You can use the **Properties** tab or window if your selection is in the content table.
3. In the **Value** column, double-click the **Module** property value to display the Multi-Choice dialog window.
4. Do one or both of the following:
 - Check the checkbox for each module where you want display the menu.
 - Clear the checkbox for each module where you want to remove the menu.

You can check or clear the checkboxes in any combination.

Rearranging Menus and Items

Custom main menus are displayed between the **Tools** and **Help** menus, from left to right in alphabetical order of the top level menu folders in the **Activators** folder. Submenus and menu items are displayed in alphabetical order of object names within the menu folder hierarchy.



Any display name assigned to a menu folder object or a menu item object appears in alphabetical order of the object name. For more information, see [Assigning a Display Name to a Menu or Menu Item](#), earlier in this chapter.

To rearrange a menu or menu item:

- Within the **Activators** folder, rename the object according to the new position.

Notebook Pane

As in the Systems Engineering and Requirements Management module, the notebook pane displays information for the object selected in the navigation tree or the content table. In the Administration module, the notebook pane contains the **Properties**, **Attachments**, **Preview**, and **Where Used** tabs. Each tab can be opened in a separate floating window.

Properties Tab

The **Properties** tab displays all viewable properties that apply to the selected schema object. Values that you cannot change are dimmed in the **Value** column. To change an editable property value, you can double-click the value to open the Single-Choice dialog window, the Multi-Choice dialog window, or a text field, depending on the property type.

The **Properties** tab has controls that filter the properties displayed in the table. You can use these controls to view properties according to certain categories. For more information, see [Filtering the Properties Tab](#), later in this chapter, and chapter 2, [Customizing Object Properties](#).

Attachments Tab

The **Attachments** tab displays the notes attached to the selected schema object. You can attach notes to schema objects in the same way as you do to objects in the Systems Engineering and Requirements Management module. You can also attach notes to other notes.



You cannot attach notes to individual user objects. However, you can attach notes to user groups.

When you create a note, you choose one of the following content formats:

- Plain text, shown by the **Text** value of the note's **Text Format** property.
- Rich text, shown by the **HTML** value of the note's **Text Format** property.
- Microsoft Word, shown by the **OpenXML** value of the note's **Text Format** property.

You can also assign a user defined subtype to the note. For more information, see [Creating a Subtype](#) in chapter 3, *Customizing Object Types*.

This tab displays the notes hierarchically in the **Attachment** column. This column contains a plus sign (+) for each note to which at least one other note is attached. You can click the plus sign to see the notes at the next lower level. Plus signs may be displayed also for lower level notes.

For more information about working with notes, see the *Systems Architect/Requirements Management User's Manual*.

Preview Tab

The **Preview** tab displays the content of the style sheet, template, activator, or macro selected in the content table. Also, you can open the **Preview** floating window to view the content of the note selected in the **Attachments** tab or window. For more information, see [Floating Tab Windows](#), later in this chapter.

Where Used Tab

The **Where Used** tab displays all objects that reference the selected schema object. Referencing objects shown in the tab depend on the selected object type:

Object Type	Referencing Objects
Activator	Type definitions
Property definition	Type definitions
Document template	Folder instances, saved searches
Object template	Type definitions, document templates
Style sheet	Document templates
View	Folder type definitions, folder instances, user objects
User	User groups, security profiles, projects
User group	Security profiles
Security profile	Objects to which the security profile applies

If a referencing object resides in the Systems Engineering and Requirements Management module, you can navigate to that object by selecting it in the **Where Used** tab, pulling down the **View** menu, and choosing the **Go To**→**Go To Object** options. Or, right-click the object in the tab and choose **Go To Object** from the popup menu. You cannot automatically navigate to referencing objects in the Administration module.

Floating Tab Windows

You can open the top tab in a separate floating window by clicking the **Open tab** button on the **Notebook** pane toolbar. To open all tab windows simultaneously, click the **Open each tab** button. Also on the toolbar, you can:

- Click the **Close all** button to close each open tab window in a single action.
- Click the **Minimize** button to reduce the **Notebook** pane to its minimum height.
- Click the **Maximize** button to enlarge the **Notebook** pane to its maximum height.

You can see a description of each button by resting the pointer on the button to display a tooltip.

Filtering the Properties Tab

In the **Properties** tab and floating window, you can set the properties that you want to see in the property table. By default, the table displays all viewable properties for the selected object or objects.

You can limit the properties to a specific subset through filtering options on the **View** and popup menus and through filter buttons in each view. For more information, see appendix [System-Defined Properties in the Administration Module](#), and [Overview of Property Definitions](#) in chapter *Unsatisfied xref reference*, *Unsatisfied xref title*.



The filter buttons in the **Properties** tab are hidden each time you start a new Architect/Requirements session. In the floating window, the buttons are hidden each time you open the window.

To show the filter buttons:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Show Controls**.
- Right-click the table and choose **Filter**→**Show Controls** from the popup menu.
- Click the down arrow on the horizontal split bar above the table.



You can also point to the split bar until the pointer becomes a double vertical arrow. Then, click the split bar to size the filter bar automatically, or drag the split bar to set the size that you want.

To hide the filter buttons:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Hide Controls**.
- Right-click the table and choose **Filter**→**Hide Controls** from the popup menu.
- Click the up arrow on the horizontal split bar above the table.

To display only properties whose values you can change:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Editable**.
- Right-click the table and choose **Filter**→**Editable** from the popup menu.
- Click the **Editable** filter button.

To display only properties whose values cannot be changed:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**Read-only**.
- Right-click the table and choose **Filter**→**Read-only** from the popup menu.
- Click the **Read Only** filter button.

To display only the custom properties for the project:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**User Defined**.
- Right-click the table and choose **Filter**→**User Defined** from the popup menu.
- Click the **User Defined** filter button.

To display only system defined properties:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**System**.
- Right-click the table and choose **Filter**→**System** from the popup menu.
- Click the **System** filter button.

To display all viewable properties:

Do one of the following:

- Pull down the **View** menu and choose **Filter Properties**→**All**.
- Right-click the table and choose **Filter**→**All** from the popup menu.
- Click the **All** filter button.

Creating a Project

Only enterprise administrators can create projects. By default, the creator becomes the original project administrator, with full access to perform any action in the project and to delete the project. In addition, all other enterprise administrators are added to the **Users** folder in the new project and have full access automatically.



You must have **Enterprise Administrator** privilege to perform this procedure.

For any other user, access must be granted specifically after the project's creation. Access can be granted by the creator, by another enterprise administrator, or by a user who is subsequently granted **Project Administrator** access privilege. For more information, see [Project Access](#) and [Granting or Revoking Project Access](#) in chapter 5, *Managing Users*.



- To automatically create a full project with objects in both the Systems Engineering and Requirements Management and Administration modules, you can import object data from an XML file. For more information, see [Importing a Project](#), later in this chapter.
- The new project can be deleted by users who are subsequently granted **Project Administrator** access privilege. For more information, see [Deleting a Project](#), later in this chapter, and [Modifying a User Profile](#) in chapter 5, *Managing Users*.

To create a project:

1. In the navigation tree, select the root node.

The content table displays all projects in the database.

2. Pull down the **File** menu and choose the **New**→**Project** options.

The content table displays the project with a default name in an open text field.

3. Enter the project name, and then press the enter key.

To refresh the view, pull down the **View** menu and choose **Refresh**. You can also right-click the root node and choose **Refresh** from the popup menu. Or, click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the root node and choose the **New**→**Project** options from the popup menu. Or, right-click in the content table and choose **New Project** from the popup menu.

Importing a Project

An enterprise administrator can create an entire project in one action by importing data from an XML file. The new project automatically contains objects in both the Systems Engineering and Requirements Management module and the Administration module.



You must have **Enterprise Administrator** privilege to perform this procedure.

The import file can be one containing data that was previously exported from the Administration module. For more information, see [Exporting Project Data](#), later in this chapter. Or, you can use the data in an XML file exported from a product other than Architect/Requirements.



The file being imported must have been exported from either the same Architect/Requirements version or the prior major version. Exporting from a newer version and importing into an older version is not supported because it may not be possible to map new data into an older database.

For the Systems Engineering and Requirements Management module, the data represents folders, requirements, building blocks, groups, notes, and diagrams, complete with the following:

- Properties and values, both system defined and user defined.
- Parent, child, and sibling relationships among imported objects.
- Trace links between objects that reside within the project from which the data was exported. Interproject trace links are not imported.
- Versions and variants of requirements and building blocks.
- **Create User** and **Create Time** system defined properties of the objects. If the created user does not exist in the database, then the current user is set as the created user. For more information, see the [Table of System-Defined Properties](#) in the appendix [System-Defined Properties in the Administration Module](#).



In the export file, a diagram may contain shapes representing objects that reside outside the diagram's parent folder. These cross-folder or cross-project references are not preserved in the imported diagram, and the corresponding shapes are marked in red.



User groups cannot be imported into projects.

For the Administration module, the file contains data that defines the following schema objects:

- Activators and macros in the **Activators** folder.
- Property definitions in the **Property Definitions** folder.
- Document templates, object templates, style sheets, and reports in the **Reports and Formatting** folder.
- Security profiles in the **Security Profiles** folder.
- Object types and subtypes in the **Type Definitions** folder.



Data is not imported for user objects, proxy objects, or links that are external to the project, such as Teamcenter Interface links.

Schema objects defined in the file are created if they do not exist in the database. Existing schema objects are updated in the database as defined in the file.



- Before importing, ensure that you know the name and location of the XML file containing the data that you want.
- Although the import file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the import file.

To import a project:

1.

Select the root node in the navigation tree, and then pull down the **File** menu and choose the **Import**→**Project** options.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder.

2. Select the import file in the list, or enter the name in the **File name** field.

The **Look in** field displays the current drive or folder. If the import file is not in the list, you can use this field to change the drive or folder.

3. Click **Open** or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

The dialog window closes, and a message indicates that the import is in progress. When the import is complete, a confirmation message is displayed, and the new project node is displayed in the navigation tree and the content table.

4. Select the project node, and then pull down the **File** menu and choose **Rename**.

5. Enter the project name in the open text field, and then press the enter key.

Procedure Notes

Step 1: You can also right-click the root and choose the options from the popup menu.

Step 4: You can also right-click the project and choose **Rename** from the popup menu. Or, press the F2 key. To reverse this action, pull down the **Edit** menu and choose **Undo Rename**.



If you are importing one or more projects from Architect/Requirements versions prior to 8.2, perform the upgrade steps mentioned in the topic *Converting Excel Templates* described in Chapter 3 of the *Architect/Requirements Management Server Installation Manual*. You can also use the bulk conversion method for converting the templates. The bulk conversion method ignores templates from existing projects that are already in the Open XML format.

Exporting Project Data

Data representing objects in a project can be exported from the Administration module to an XML file. The export file is generated by Architect/Requirements and is saved in the drive or folder that you specify. Then the data can be imported to another project, where Architect/Requirements reproduces the objects. Thus, objects can be shared among projects in the same Architect/Requirements installation and in installations at other sites. For more information, see [Importing a Project](#) and [Importing Schema Objects](#), later in this chapter.

Depending on your purpose, you can:

- Export all data for the entire project. From the Systems Engineering and Requirements Management module, the data represents all of the project's folders, requirements, building blocks, groups, notes, and diagrams. The object data is complete with the following:
 - Properties and values, both system defined and user defined.
 - Parent, child, and sibling relationships among the objects.
 - Trace links between objects that reside within the project. Interproject trace links are not exported.
 - Versions and variants of requirements and building blocks.



A diagram may contain shapes representing objects that reside outside the diagram's parent folder. Although these cross-folder or cross-project references are exported to the XML file, they are not preserved when the diagram is imported.

In addition, the file includes all of the project's schema data, which defines the following schema objects in the Administration module:

- Activators and macros in the **Activators** folder.
- Property definitions in the **Property Definitions** folder.
- Document templates, object templates, style sheets, and reports in the **Reports and Formatting** folder.
- Security profiles in the **Security Profiles** folder.
- Object types and subtypes in the **Type Definitions** folder.

A project administrator can import this file to create a new project automatically, complete with predefined objects in both modules.

- Export schema data only, representing schema objects in the Administration module. To a given file, you can export data for either of the following:
 - All schema objects in the project, including both system defined and user defined schema objects.
 - Selected schema objects within a folder, where you can select one object or a group of nonadjacent or adjoining objects.

A project administrator can import this file to initialize a new project with predefined schema, or to update the schema of an existing project.



- Data is not exported for user objects, proxy objects, or links that are external to the project, such as Teamcenter Interface links.

- The **.xml** file name extension is assigned to all files containing data exported from the Administration module. Siemens PLM Software recommends that you give each export file a name that clearly identifies the data.

To export project data:

1.

Do one of the following:

- To export the entire project, select the project node in the navigation tree, and then pull down the **File** menu and choose the **Export**→**Project** options.
- To export all schema data in the project, select the project node in the navigation tree, and then pull down the **File** menu and choose the **Export**→**Schema** options.
- To export selected schema data:
 - . Open the containing folder, and then select the schema objects in the content table.
You can select both nonadjacent and adjoining objects.
 - . Pull down the **File** menu and choose the **Export**→**Schema** options.

Architect/Requirements displays the Save dialog window, which lists existing folders and files in the current drive or folder.

2. In the **File name** field, enter a name that clearly identifies the exported data.

The **Save in** field displays the current drive or folder. You can save the file in the displayed location, or you can use this field to change the drive or folder.

3. Click **Save** or press the enter key.

The dialog window closes, and a message is displayed to indicate that the export is in progress. When the export is complete, the message closes, and the file is saved in the specified location.



Although the export file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the export file.

Procedure Notes

Step 1: You can also right-click the project node in the navigation tree and choose the options from the popup menu. For selected schema data, you can also right-click the selection in the content table and choose **Export Schema** from the popup menu.

Step 2: The **.xml** file name extension is added automatically. You can also select an existing XML file in the list to overwrite that file.

Importing Schema Objects

Using data exported from the Administration module to an XML file, a project administrator can easily reproduce one project's schema objects in another project. Objects defined in the file are created if they do not exist in the database. Existing objects are updated in the database as defined in the file.



The file being imported must have been exported from either the same Architect/Requirements version or the prior major version. Exporting from a newer version and importing into an older version is not supported because it may not be possible to map new data into an older database.

The following schema objects can be imported to the Administration module:

- Activators and macros in the **Activators** folder.
- Property definitions in the **Property Definitions** folder.
- Document templates, object templates, style sheets, and reports in the **Reports and Formatting** folder.
- Security profiles in the **Security Profiles** folder.
- Object types and subtypes in the **Type Definitions** folder.

Data is not imported for proxy objects, user objects, or links that are external to the project, such as Teamcenter Interface links.



- Before importing, ensure that you know the name and location of the XML file containing the data that you want.
- Although the import file can be opened in a text editor such as Microsoft WordPad or Notepad, changing the data is an advanced operation. Siemens PLM Software recommends that you do not open the import file.

To import schema objects:

1.

Select the project node in the navigation tree, and then pull down the **File** menu and choose the **Import**→**Schema** options.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder.

2. Select the import file in the list, or enter the name in the **File name** field.

The **Look in** field displays the current drive or folder. If the import file is not in the list, you can use this field to change the drive or folder.

3. Click **Open** or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

The dialog window closes, and a message indicates that the import is in progress. When the import is complete, a confirmation message is displayed.

Procedure Notes

Step 1: You can also right-click the project and choose the options from the popup menu.

Running an Import From the Command Line

You can enter the **tcradmin** script on the command line to import data to the Architect/Requirements database.

The following example shows the **tcradmin** syntax:

```
tcradmin -action import -type <typeOfImport> -file <filename> -loid <importOwner> -
user <username> -password <password> [-key <installationKey>] [-location
<originalFileName>] [-subtype <subtype>]
```

Table 1-1 describes the **tcradmin** arguments.

Table 1-1. tcradmin Arguments for Import From the Command Line

Argument	Description
-type	Import action to perform:
	XML Import a folder of data in XML format.
	XML_UPDATE Import and merge a folder of data in XML format.
	PROJECT Import an entire project in XML format.
	SCHEMA Import a project's schema or selected schema objects.
	MS_WORD Import a Word document. This is the default.
	STYLESHEET Import a Word document into a style sheet object.
	MS_EXCEL Import an Excel spreadsheet.
	EXCEL_TEMPLATE Import an Excel template.
	AP233 Import data in AP233 format.
-file	Complete path of the import file.
-loid	LOID of the object to import into.
-user	Architect/Requirements user name under which you want to log in to the server.
	 To import an entire project, this user name must have Enterprise Administrator privilege.
-password	Architect/Requirements password for the user name under which you want to log in to the server.
-key	Web server installation identifier used to look up Web application parameters such as ImportExportDir .

Table 1-1. tcradmin Arguments for Import From the Command Line

Argument	Description
	 This argument defaults to <i>machinename</i> + " 1 ", which is correct in most circumstances. This argument may be needed if there are no or more than one Architect/Requirements Web servers installed on this machine.
-location	Original Word document file name to store as property on imported requirements.
-subtype	Requirement subtype to use for requirements imported from Word.
-logToStdOut	Redirects the output of the tcradmin command to stdout . By default, most of the output of tcradmin is written to the Architect/Requirements server log file, TcrServerLog.html . The Architect/Requirements server log is written in a tabular HTML format that is convenient to view. When redirected to stdout , the Architect/Requirements log information is written as plain text.

Running an Export From the Command Line

You can enter the **tcradmin** script on the command line to export data from the Architect/Requirements database.



The data is exported to a temporary file with a system-generated name. The last line of script output displays the path to the export file.

The export file location is specified by the **ImportExportDir** configuration parameter. For more information about the **ImportExportDir** parameter and customizing the Architect/Requirements server, see the *Systems Architect/Requirements Management System Administrator's Manual*.

The following example shows the **tcradmin** syntax:

```
tcradmin -action export -type <typeOfExport> -loid <objectsToExport> -user  
<username> -password <password> [-template <outputTemplate>] [-props <properties>]  
[-live] [-notDeep] [-noOLE] [-key <installationKey>]
```

Table 1-2 describes the **tcradmin** arguments.

Table 1-2. tcradmin Arguments for Export From the Command Line

Argument	Description
-type	Export action to perform:
	XML Export data in XML format.
	PROJECT Export an entire project in XML format.
	SCHEMA Export a project's schema or selected schema objects.
	MS_WORD Export a Word document. This is the default.
	MS_EXCEL Export an Excel spreadsheet.
	AP233 Export data in AP233 format.
	TC_XML Export data for migration to Teamcenter.
-loid	LOID or comma-separated list of LOIDs of the objects to export.
-user	Architect/Requirements user name under which you want to log in to the server.
-password	Architect/Requirements password for the user name under which you want to log in to the server.
-template	The document or Excel template to use. Valid only for the MS_WORD and MS_EXCEL export types.
-props	Properties to include as column headings in a Microsoft Office Excel export file.

Table 1-2. tcradmin Arguments for Export From the Command Line

Argument	Description
-live	Make the exported Excel spreadsheet live.
-notDeep	Export only the specified objects to Microsoft Office Word and do not include their descendents.
-noOLE	Do not include OLE objects in a Microsoft Office Word export file.
-key	Web server installation identifier used to look up Web application parameters such as ImportExportDir .  This argument defaults to <i>machinename</i> + "\1", which is correct in most circumstances. This argument may be needed if there are no or more than one Architect/Requirements Web servers installed on this machine.
-logToStdOut	Redirects the output of the tcradmin command to stdout . By default, most of the output of tcradmin is written to the Architect/Requirements server log file, TcrServerLog.html . The Architect/Requirements server log is written in a tabular HTML format that is convenient to view. When redirected to stdout , the Architect/Requirements log information is written as plain text.

Managing Rule File to Export or Import Objects

Architect/Requirements provides mechanisms to export and import the Architect/Requirements objects and their properties in XML format that is compliant with the AP233 Part 28 standard. The Part-28 format of the AP233 standard provides many constructs (collection of XML elements) that can be used to represent the Architect/Requirements object types and the property types in the XML format.

A Rule File:

- Is an XML file that describes the mapping from the Architect/Requirements object types and property types to the AP233 tags.
- Is a collection of rules for Architect/Requirements object types and property types of objects.
Each rule is a collection of XML elements that the Architect/Requirements export operation processes to write objects to an XML file in the AP233 schema and the Architect/Requirements import operation processes to create or update the Architect/Requirements objects from an AP233 XML file.
- Can be used to customize the behavior of the export and import operations.

Using the Rule File

Rule file is a collection of rules for objects. A rule contains AP233 schema tags and some XML tags (such as the <Method> tag) that are used by Architect/Requirements for processing.

The **Default AP233 Rule File** is located in the **Reports and Formatting** folder of every project in the Administration module. There is only one rule file for each project, to maintain data integrity and to reduce potential errors during export or import.

Project administrators can export a rule file by right-clicking the rule file and selecting the **Export Schema** or **Copy to client** option. They can import a rule file by right-clicking the rule file to be updated and selecting the **Import→AP233 Rule File** option.

A rule file can be used by the project administrators to:

- Specify rules for object types and their property types that are used during the export and import operations.
- Specify the properties of objects to be included or excluded during export to or import from an AP233 XML file.
- Specify the unique property for updating objects during import from an AP233 XML file.

Identifying the Rule for Objects

The name of the rule is either object type name or its subtype name. If a rule with a subtype name is not present, then the rule of its parent type is processed. For example, the rule for a requirement is processed if there is no rule for a paragraph.

An extract of the rule for building block objects is shown below:

```

<BuildingBlock>
<ap233:System>
<ExportRule>...</ExportRule>
<ImportRule>...</ImportRule>
</ap233:System>
...
<ap233:System_version>
<ExportRule>...</ExportRule>
<ImportRule>...</ImportRule>
</ap233:System_version>
...
<ap233:View_definition_context>...</ap233:View_definition_context>
...
<ap233:System_view_definition>...</ap233:System_view_definition>
...
</BuildingBlock>

```

The export process looks up the mapping specified in the rule file to determine what objects to export to an output file and how to export them. The import process looks up the mapping specified in the rule file to determine what objects to create or which properties to update from an input file.

In the preceding example, `<ap233:System>`, `<ap233:System_version>`, `<ap233:View_definition_context>`, and `<ap233:System_view_definition>` are the AP233 elements that are used to describe building blocks. Similarly, other Architect/Requirements object types are described using such AP233 elements in the rule file.

The export rule for the building blocks in the selected project corresponds to the `<ExportRule>...</ExportRule>` element. Each `<ExportRule>` element contains AP233 and `<Method>` elements used to get data from Architect/Requirements. Similarly, the `<ImportRule>...</ImportRule>` element corresponds to the import rule for the building blocks. Each `<ImportRule>` element contains `<Method>` elements used to create or update data in Architect/Requirements.

Identifying the Rule for Properties of Objects

An object can have system defined properties and user defined properties. Similar to the rules for Architect/Requirements object types, there are separate rules for system defined and user defined properties of objects in Architect/Requirements. Each of these rules contains AP233 and other XML elements that describe the properties and their behavior during export and import operations.

A rule file contains rules for the following property types:

Property Type	System defined	User defined
Text	TextDefinition	TextProperty
Choice	ChoiceDefinition	ChoiceProperty
Date	DateDefinition	DateProperty
Numeric	NumericDefinition	NumericProperty

In a rule file, the name of the rule for properties is based on the property type. For example, the rule for a user defined text property is **TextProperty** and the rule for a system defined date property is **DateDefinition**.

Including and Excluding Properties

By default, all user defined properties are exported and imported. A rule element has **include** and **exclude** attributes that can be used to specify the properties of objects to be included or excluded during export to and import from an AP233 XML file.

To include any property for export and import, add the name of the property to the **include** attribute. Similarly, if you want to exclude a property, add it in the **exclude** attribute. The **include** attribute also supports the * wild card, which includes all matching properties.

For example, to include all the values of the **ChoiceProperty** property, except the **Baseline** property:

```
<ChoiceProperty include="*" exclude="Baseline">
```

To include all the values of **NumericProperty** property:

```
<NumericProperty include="*">
```

To include only the **Name** values in the **TextDefinition** property:

```
<TextDefinition include="Name">
```

To include only the **ROIN** and **Number** values in the **TextDefinition** property:

```
<TextDefinition include="ROIN,Number">
```

The above rule is used only during export and not during the import operation.

Updating Objects During Import

When you import from an AP233 XML file, the import process updates an object if it already exists in Architect/Requirements. Otherwise, a new object is created.

The import process determines whether to update an existing object or to create a new object, based on the following conditions:

- The import process checks whether the object being imported is present in Architect/Requirements within the selected folder or project.
- The property used to determine the uniqueness of the object is specified in the rule file.
- If there is an object present with the same value for the property specified in the rule file, then the import process does not create a new object.

For example, the following condition is used to evaluate the uniqueness of requirement objects:

```

<Condition name="isRequirementUnique">
    <Method name="getMatchingElement" static="false"
contextObject="importerObject" output="preserve">
        <args>
            <arg type="String" default="ROIN" />
            <arg type="Vector" reverseRef="Of_product/
ap233:Requirement_version/Defined_version/
ap233:Requirement_view_definition/Described_element/
ap233:Assigned_property/Definition/ap233:Property_representation" />
            <arg type="String"
default="Definition/ap233:Assigned_property/Name" />
        </args>
    </Method>
    <Method name="findObject" static="false" contextObject="importerObject">
        <args>
            <arg type="Object" default="previousObject" />
            <arg type="String"
default="Definition/ap233:Assigned_property/Name" />
            <arg type="String" default="Rep/ap233:Representation/Items/
ap233:String_representation_item/String_value" />
            <arg type="String" ref="Id" />
            <arg type="String" default="text" />
        </args>
    </Method>
</Condition>

```

In this example, the condition for the requirement object is **isRequirementUnique** specified in the **<Condition name="isRequirementUnique">** element. Similarly, the condition for notes is **isNoteUnique** and the condition for building blocks is **isBuildingBlockUnique**. For trace links, there is no condition to be specified. The uniqueness of a trace link is determined by the end object to which it is attached.

Specifying the Unique Property for Updating Objects During Import

To change the property that is used to determine the uniqueness of an object, users should change the property name present in the first argument value in the **getMatchingElement** method. In the above example, the **<arg type="String" default="ROIN" />** element specifies **ROIN** as the property to check for the uniqueness of a requirement object to be imported. To illustrate, if Architect/Requirements contains a requirement object with the **ROIN** value 123 and the AP233 XML file being imported also has a requirement with the same **ROIN** value, then the import process does not create a new object. Also, the import process updates the properties of the existing object with the properties of the object in the imported file.

If the unique property is changed to **Name** instead of **ROIN**, then the import process looks for the name of the requirement to determine the uniqueness of the requirement object being imported.

If the new property to be used in the condition is of a different type, such as a user defined numeric property instead of **ROIN** (a text property), then the arguments in the **findObject** method should also be modified, in addition to the property name. An example of the arguments in the **findObject** method is shown below.

```

<Method name="findObject" static="false" contextObject="importerObject">
  <args>
    <arg type="Object" default="previousObject" />
    <arg type="String"
default="Definition/ap233:Assigned_property/Name" />
    <arg type="String" default="Rep/ap233:Representation/Items/
ap233:String_representation_item/String_value" />
    <arg type="String" ref="Id" />
    <arg type="String" default="text" />
  </args>
</Method>

```

For a text property, the third argument should be changed to:

```

<arg type="String" default="Rep/ap233:Representation/Items/
ap233:String_representation_item/String_value" />

```

For a numeric property, the value of the third argument should be:

```

<arg type="String" default="Rep/ap233:Representation/Items/
ap233:Numerical_item_with_unit/Value_component/Any_number_value-wrapper" />

```

In addition, the fifth argument should be changed if text property is not specified in the third argument.

For a numeric property, the fifth argument should be changed to:

```

<arg type="String" default="number" />

```

For a choice property, the fifth argument should be changed to:

```

<arg type="String" default="choice" />

```

The date property is not supported for use as the property for checking the uniqueness of the object being imported.

Architect/Requirements XML Format

This section describes the basic organization and content for the Architect/Requirements XML format that can be used for import or export operations. In many cases, the questions not addressed here can be answered by using the **Export**→**XML** command on an object of interest and then inspecting the file.

Types of XML Files

Architect/Requirements supports three types of XML files: schema, data, and project. XML files are read and written using the **importDocument** and **exportDocument** API methods. The type of the XML file determines the file type keyword to use. For more information, see the *Systems Architect/Requirements Management API Reference*.

- Schema XML file contains the Architect/Requirements objects from a project that are visible in the administration module. Schema XML files may contain all or only some of the administration objects. Use the **SCHEMA** keyword to import or export a schema.
- Data XML file contains the Architect/Requirements objects from a project that are visible in the requirements module. Data XML files may contain all or only some of the non-administration objects. Use the **XML** keyword to export data objects. Use the **XML** or the **XML_UPDATE**

keyword to import data objects. The **XML_UPDATE** keyword is available only through the API method and allows merging changes into existing objects.

- Project XML file contains the complete schema and data XML files for a project concatenated together. The files are separated by the **##### TCR #####** string. Because a project file actually contains two separate XML files, it is not a well-formed XML file. This means the project file may not be usable in some XML viewing utilities. In that event, the file can be manually split into two pieces (using Notepad or a similar text editor) for the XML viewer. Use the **PROJECT** keyword to import or export the entire project.

Types of Data Elements

The data file contains a single **<data>** element. The data element contains a flat list of project object elements. The tag for an object uses the database object type name (**DataBean.TYPE**), for example, **<RequirementDB>**. The data for an object is placed in the sub elements.

The three types of data elements are required fields, property values, and object type-specific fields.

-

Required Fields

These elements must be included for each object. If they are not included, the object is not imported. Even if a particular field has no value, its element must be included.

- o **name** is the name of the object. Note that XML special characters must be encoded. For example, **>** is specified as **>**;
- o **id** is the unique object identifier, used for referencing other objects, as with the **<owner>** tag. To update objects with the **XML_UPDATE** keyword, the **id** field must be the LOID of the existing object. For other imports, any value can be used, as long as it is unique within the XML file.
- o **owner** is the ID of the object's owner. For objects that do not have an owner, such as trace links, use **null**.
- o **typedefinition** is the name of the object's type definition (its subtype value). If there is no type definition with this name when an object is imported, it uses the base type instead.

For example:

```
<name>R1</name>
<id>89.0.3649333</id>
<owner>89.0.3650243</owner>
<typedefinition>Requirement</typedefinition>
```

-

Property Values

Most property values, both system defined and user defined, are included using a property element. Attributes are used for the property name and the type (system or user). The element's value is the property value.

For numeric and date properties, a **rawdata** attribute may also be used. Raw data is a normalized form of the value used to avoid localization and time zone issues when parsing. For dates, raw data is the timestamp. For numbers, raw data is the number in English locale.

Read-only properties are included in exports; however, they are ignored on import. All the properties are optional for import. If a property is not included, it assumes its default value for

objects that are created, or retains its existing value for updates. Properties that are not initialized may have a blank or **null** value.

For example:

```
<property name="ROIN" type="system">0005</property>
<property name="Create Time" type="system" rawdata="1224344701202">
10/18/2008 10:45 AM</property>
<property name="Color" type="user">Blue</property>
```

•

Type-Specific Fields

Some object types support additional fields for type-specific data.

Object types that can be versioned, such as requirements and building blocks, have the following fields:

- o **master** is the ID of the current version of the object.
- o **versionnumber** is the version number of the object. **0** is used for objects that have never been frozen.

For example:

```
<master>89.0.3649333</master>
<versionnumber>0</versionnumber>
```

Objects with rich text, such as requirements and notes, have the following fields:

- o **HTML** is the encoded text content.
- o **filename** is the internally used unique file name for referencing graphics and OLE objects.

For example:

```
<HTML></HTML>
<filename>>/Requirement-0001-11d108331fd.html</filename>
```

Linking objects, such as trace links and connections, have the fields:

- o **frontend** is the ID of the object that the link points to (complying end).
- o **backend** is the ID of the object that the link points from (defining end).
- o **linktype** is the type of the link (**DataBean.LINK**).

For example:

```
<frontend>89.0.3649333</frontend>
<backend>89.0.3649284</backend>
<linktype>Complying</linktype>
```

Other type-specific fields can be obtained by examining the XML output for objects of that type.

•

Deleting Objects

When using the **XML_UPDATE** keyword, the existing objects can be deleted from the database by including `action="delete"` in the main object tag.

For example:

```
<BuildingBlockDB action="delete">
<id>410.0.11023</id>
<owner>410.0.10867</owner>
<typedefinition>Building Block</typedefinition>
</BuildingBlockDB>
```

Schema XML Format

The schema file contains a single **<schema>** element. The schema element contains a flat list of administration object elements. The tag for a schema object uses the database object type name (**DataBean.TYPE**), for example, **<UserTypeDefinitionDB>**.

Schema imports always attempt to merge with existing schema objects. Schema object references generally are by name instead of ID (the owner field is an exception). If an element in the XML file has the same name as an existing schema object of the same type, it updates that object. Where possible, the updates are non-destructive, that is, the data is added but not removed. For example, the import associates new properties with an existing type definition. However, the import does not remove the properties that are not included in the XML file.

Order of Objects

Forward object references in the XML files are not supported. Referencing another object can be done only if the referenced object appears earlier in the XML file. For example, a member of a folder must appear after the folder, and a trace link must appear after the two objects that are linked.

Removing References to a Schema Object

The **Where Used** tab displays the objects that refer to the object selected in the content table. For example, when you select a property definition, the tab shows the type definitions to which the property definition is applied. For more information, see [Notebook Pane](#), earlier in this chapter.

To remove references to a schema object:

1. Select the object and click the **Where Used** tab to display the referencing objects.



To see the locations of the objects, you can add the **Full Name** property in the tab. Right-click any column heading to display the Column Settings dialog window, check the **Full Name** checkbox, and click **OK**.

2. Depending on the schema object type and the referencing object type, do one of the following actions:

Schema Object	Referencing Object	Action
Activator	Type definition	a. In the Type Definitions folder, select the referencing type definition.

Schema Object	Referencing Object	Action
		<ul style="list-style-type: none"> b. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. c. Double-click the Activators value to display the Multi-Choice dialog window. d. Clear the checkbox for the activator that you want to remove, and then click OK.
Property definition	Type definition	<ul style="list-style-type: none"> e. In the Type Definitions folder, select the referencing type definition. f. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. g. Double-click the Properties value to display the Multi-Choice dialog window. h. Clear the checkbox for the property definition that you want to remove, and then click OK.
		 <p>For certain property definitions, references have generic names that are not found in the Type Definitions folder. For Shared State, for example, references named Menu Item and View have no explicit type definitions. Such references are system-generated and cannot be removed.</p>
Document template	Search	<ul style="list-style-type: none"> i. In the Reports and Formatting folder, select the referencing search. j. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. k. Double-click the Document Template value to display the Single-Choice dialog window. l. Clear the checkbox for the document template and click OK.
Document template	Folder instance	<ul style="list-style-type: none"> m. In the Where Used tab, select the referencing folder, and then pull down the View menu and choose the Go To→Go To Object options.

The folder is selected automatically in the Systems Engineering and Requirements Management module.

Schema Object	Referencing Object	Action
		<ul style="list-style-type: none"> n. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. o. Double-click the Document Template value to display the Single-Choice dialog window. p. Clear the checkbox for the document template and click OK.
Object template	Document template	<ul style="list-style-type: none"> q. In the Reports and Formatting folder, select the referencing document template. r. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. s. Double-click the Object Template value to display the Single-Choice dialog window. t. Check the checkbox for a different object template, and then click OK. <p> The document template must use an object template. If there is no other object template in this dialog window, you cannot remove the current reference.</p>
Object template	Type definition	<ul style="list-style-type: none"> u. In the Type Definitions folder, select the referencing type definition. v. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties. w. Double-click the Object Template value to display the Single-Choice dialog window. x. Clear the checkbox for the object template and click OK.
Style sheet	Document template	<ul style="list-style-type: none"> y. In the Reports and Formatting folder, double-click the referencing document template to display the Document Template dialog window. z. In the Select Stylesheet field, select a different style sheet. <p> The document template must use a style sheet. If there is no other style sheet in this field, you cannot remove the current reference.</p>

Schema Object	Referencing Object	Action
		aa. Click Save to change the style sheet reference.
View	Folder type definition	<p>bb. In the Type Definitions folder, select the referencing folder type definition.</p> <p>cc. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>dd. Double-click the Default View value to display the Single-Choice dialog window.</p> <p>ee. Clear the checkbox for the view and click OK.</p>
View	User	<p>ff. In the Users folder, select the referencing user.</p> <p>gg. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>hh. Double-click the Default View value to display the Single-Choice dialog window.</p> <p>ii. Clear the checkbox for the view and click OK.</p>
View	Folder instance	<p>jj. In the Where Used tab, select the referencing folder, and then pull down the View menu and choose the Go To→Go To Object options.</p> <p>The folder is selected automatically in the Systems Engineering and Requirements Management module.</p> <p>kk. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>ll. Double-click the Default View value to display the Single-Choice dialog window.</p> <p>mm. Clear the checkbox for the view and click OK.</p>
User	User group	nn. In the Users folder, click the plus sign (+) for the user group, and then select the shortcut that represents the user.

Schema Object	Referencing Object	Action
		<p>oo. Pull down the File menu, and choose Delete. You can also click the Delete button on the toolbar or press the delete key.</p> <p>A confirmation message asks if you are sure you want to delete the object.</p> <p>pp. Click Yes to remove the user reference.</p>
User or user group	Security profile	<p>qq. In the Security Profiles folder, select the referencing security profile.</p> <p>rr. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>ss. Double-click any value that includes the user or user group. Such values may be any or all of the following:</p> <p>Change Approvers</p> <p>Change Notifiers</p> <p>Full Control</p> <p>Modify and Read Access</p> <p>Read Access</p> <p>tt. In the Multi-Choice dialog window, clear the checkbox for the user or user group and click OK.</p> <p>Repeat steps c. and d. for each remaining value that includes the user or user group.</p>
Security profile	Any other schema object	<p>uu. Open the containing folder and select the referencing schema object.</p> <p>vv. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>ww. Double-click the Security Profile value to display the Single-Choice dialog window.</p> <p>xx. Clear the checkbox for the security profile and click OK.</p>

Schema Object	Referencing Object	Action
Security profile	Any object instance	<p>yy. In the Where Used tab, select the referencing object, pull down the View menu, and choose the Go To→Go To Object options.</p> <p>The object is selected automatically in the Systems Engineering and Requirements Management module.</p> <p>zz. Click the Properties tab, or display the Edit Properties dialog window by pulling down the File menu and choosing Properties.</p> <p>aaa. Double-click the Security Profile value to display the Single-Choice dialog window.</p> <p>bbb. Clear the checkbox for the security profile and click OK.</p>



To remove additional uses of the same schema object, repeat steps 1 and 2.

Procedure Notes

Step 1: You can open the floating window for the **Where Used** tab by clicking the **Open tab** button on the notebook pane toolbar.

Step 2: You can open the floating window for the **Properties** tab by clicking the **Open tab** button on the notebook pane toolbar. To display the Edit Properties dialog window, you can also right-click the object and choose **Properties** from the popup menu.

Step 2: To navigate to an object in the Systems Engineering and Requirements Management module, you can also right-click the object in the **Where Used** tab or window and choose **Go To Object** from the popup menu.

Deleting a Schema Object

In any project except the **TcSE Administration** project, you can delete the following schema objects from the database:

- Activators and macros in the **Activators** folder.
- Property definitions in the **Property Definitions** folder.
- Document templates, object templates, style sheets, and search reports in the **Reports and Formatting** folder.
- Security profiles in the **Security Profiles** folder.
- Object types and subtypes in the **Type Definitions** folder.

User objects cannot be deleted from the database. However, a user can be deactivated to revoke the user's access to all projects. For more information, see [Deactivating a User](#) in chapter 5, *Managing Users*.



A given schema object can be deleted only if it is not used by any other object in the project. Before deleting a schema object, you must remove all references to the object. For more information, see [Removing References to a Schema Object](#), earlier in this chapter.



Before deleting a property definition, a type definition, or a Tcl Where clause, ensure that the object is not referenced by a search report object. The report fails if you delete a referenced object without first removing the reference.



The object cannot be recovered.



If groups are deleted using **VDBInspector**, **maintainDB** must be run to clear broken links. **maintainDB** can be run by System Administrators only.

To delete a schema object:

1. Open the containing folder, and then select the object in the content table.
2. Pull down the **File** menu, and choose **Delete**.

A confirmation message is displayed, asking if you want to delete the object.

3. Click **Yes** or press the enter key.

A second confirmation message is displayed, stating that this operation cannot be undone and asking if you want to continue.

4. To continue, click **Yes** or press the enter key.



This action clears the queue for the **Undo** option and cannot be reversed.

The object is removed from the database.

Procedure Notes

Step 2: You can also right-click the object and choose **Delete** from the popup menu. Or, click the **Delete** button on the toolbar.

Deleting a Project

You can delete any project for which you have **Project Administrator** access privilege, whether or not you are the creator of that project.



The project and all of its objects are removed from the database and cannot be recovered.

To delete a project:

1. In the navigation tree or the content table, select the project node.
2. Pull down the **File** menu and choose **Delete**.

Architect/Requirements displays a confirmation message, asking if you want to delete this object.

3. To continue, click **Yes** or press the enter key.

Architect/Requirements displays a second confirmation message, stating that a deleted project cannot be restored and asking if you want to continue.

4. To confirm the deletion, click **Yes** or press the enter key.



This action removes all of the project's data from the database and cannot be reversed.

The project node and all lower level objects are removed from the Administration module, and the view is refreshed automatically.



Certain error messages may be generated when objects in the project are displayed in client views immediately before you delete the project. For example, such a message may state that an object is not found or that the selected object does not exist in the database. Regardless of such messages, the project is deleted.

Procedure Notes

Step 2: You can also right-click the project and choose **Delete** from the popup menu. Or, click the **Delete** button on the toolbar.

Chapter 2: Customizing Object Properties

This chapter contains an overview of property definitions and instructions for creating and modifying properties that users can apply to specific objects.

Overview of Property Definitions

All information about the objects in Architect/Requirements is stored in properties. Each property has a property definition that specifies the type of information. There are four different types of properties, and four types of property definitions:

- *Choice* properties have a choice list from which a user can select the value. Project administrators set the list by editing the **Choice List** property of the choice definition. Editing this property launches a dialog window to create or modify the list. Choice lists can be defined as single-choice, which allows only one selection, or as multiple-choice, from which users can select any combination of choices. The **Multiple Choice** property controls this. System-defined choice properties have choice lists that are automatically generated.
- *Date* properties contain a date and time. The **Date Format**, **Time Format**, and **Time Flag** properties of the property definition control the date and time display. Editing any of these properties launches a dialog window to set the format. Date property values reflect the time zone and locale of property definitions in the database, which may differ from the client's time zone and locale.
- *Numeric* properties contain a number. The **Format** property controls how the number is displayed, for example, the number of decimal places. Editing this property launches a dialog window to set the format.
- *Text* properties have values that consist of text entered by the user. The value can be plain text, in a string of unlimited length. Or, the value can be a URL formatted as a hyperlink, through which users can navigate to the URL destination in Microsoft Internet Explorer or Microsoft Windows Explorer.

A fixed set of system-defined properties is always defined for each object type. These property definitions cannot be modified, although users can change the values of some system-defined properties. Other values are system-generated and cannot be modified directly. A project administrator can extend the system-defined properties by creating custom property definitions. The new properties must then be attached to object types before they can be seen in the Systems Engineering module.

To initialize a property's value on new objects, a default value can be set through the **Current Value** property of the property definition. Changing the default value later has no effect on existing objects.

Creating a Property Definition

Architect/Requirements automatically assigns certain property definitions to each built-in object type. The system-defined properties apply to all objects of that type. Although their editable values can be modified, system-defined properties are permanently set for each object type, and individual properties cannot be added or removed. However, you can extend system-defined properties by creating custom property definitions and adding those properties to type definitions.

You can create property definitions at the top level of the **Property Definitions** folder and in subfolders at lower levels. After creating a property definition, you can add the property to any number of different type definitions, including the built-in type definitions. Users can then apply the property to new and existing objects of those types.



You cannot delete a custom property after they are applied to object types.

For more information on applying properties, see [Modifying the Properties of a Type Definition](#) in chapter 3, *Customizing Object Types*.

To create a property definition:

1. In the navigation tree or the content table, open the parent folder for the new property definition. The content table displays the existing property definitions in the folder.
2. Pull down the **File** menu and choose one of the following:
 - For a choice property, choose the **New**→**Property Definition**→**Choice** options.
 - For a date property, choose the **New**→**Property Definition**→**Date** options.
 - For a numeric property, choose the **New**→**Property Definition**→**Numeric** options.
 - For a text property, choose the **New**→**Property Definition**→**Text** options.

The content table displays the new property definition with a default name in an open text field.

3. Enter the property name, and then press the enter key.

To place the property definition according to the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the root node and choose **Refresh** from the popup menu.

Procedure Notes

Step 1: To open subfolders, click the plus signs, or double-click a folder with a plus sign. In the navigation tree, you can also select a folder and then pull down the **View** menu and choose **Expand**, or right-click the folder and choose **Expand** from the popup menu. In the content table, you can also select a folder and then pull down the **View** menu and choose **Expand All**, or right-click the folder and choose **Expand All** from the popup menu.

Step 2: You can also right-click the parent folder in the navigation tree, or right-click anywhere in the content table, and then choose the options from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.



The notebook views do not respond when the **Property Definitions** folder, or one of its subfolders, is selected in the administration module's navigation pane. The **Properties and Attachments** notebook tab displays information from the previously selected object and not for

the selected **Property Definitions** folder. The notebook tabs responds when the **Property Definitions** folder is selected in the content pane.

To view the properties or notes of a properties definition folder, select the folder in the content pane.

Modifying a Choice Property Definition

A choice property has a list of choices from which users select the property value in the Systems Engineering module. By modifying a choice property definition, you can define the choice list type as single-choice, from which users can select only one value, or as multiple-choice, from which users can select any combination of choices as the value. You can add choices, rename choices, change the list order, and set the default value for the property. Also, you can delete any number of choices from the list.



System-defined choice properties have choice lists that are automatically generated.

To modify a choice property definition:

1. In the **Property Definitions** folder, select the choice definition, or open the containing subfolder and then select the choice definition.

The **Properties** tab or window displays all viewable properties for the choice definition.

2.

In the **Value** column, double-click the value for the **Choice List** property, the **Current Value** property, or the **Multiple Choice** property.

The Modify Choice Property Definition dialog window is displayed, with the current choices in the **Choice List** pane.

3. Do any or all of the following:
 - To define the type of choice list:
 - For a list that allows only one selection, click **Single Choice**.
 - For a list that allows any number of selections, click **Multiple Choice**.
 -  If you change an existing list from multiple-choice to single-choice, the change becomes effective when a user edits a value for that property in the Systems Engineering and Requirements Management module.
 - To add choices:
 - Do one of the following with **Choices** selected:
 - To add a choice in the position directly above an existing choice, select the existing choice in the list, and then click the **Add** button.
 - To add a choice at the bottom of the list, clear any current selection by holding down the control key and clicking the selection, and then click the **Add** button.

The Add dialog window is displayed.

- o Enter the new choice in the text field, and then click **OK**.

The choice is added to the list and is cleared from the text field. You can enter another new choice in the text field, and then click **OK** to add that choice and clear the field again. When you are finished adding choices, click **OK** with the text field cleared to close the Add dialog window.

You can continue adding choices by repeating steps a and b.

- To rename a choice, do the following with **Choices** selected:

- o

Select the choice in the list, and then click the **Rename** button.

The Rename dialog window is displayed.

- o Enter the name in the text field, and then click **OK** to close the dialog window.



Consider how a name change may affect the choice's current use within your project. For example, database synchronization problems may occur if the old name is used in existing object templates, live Excel spreadsheets, or live Visio diagrams. Also, search results may not be returned by saved searches that specify the old name in the criteria.

- To change the list order, do any of the following with **Choices** selected:

- o To move a choice up in the list, select the choice, and then click the **Move Up** button until the choice reaches the intended position.
- o To move a choice down in the list, select the choice, and then click the **Move Down** button until the choice reaches the intended position.
- o To sort the choices in alphabetical order, click the **Sort List** button.

- To set the default value:

- o Click **Defaults**.

At the left of the choices in the list, buttons are displayed for a single-choice list or checkboxes are displayed for a multichoice list.

- o Do one of the following:

- . For a single-choice list, click the button for the choice that you want to set as the default value.
- . For a multi-choice list, check the checkbox for each choice that you want to add to the default value, and clear the checkbox for each choice that you want to remove from the default value.



You can remove the entire default value by clicking **Clear**. However, this action is not allowed if the choice definition's **Input Required** property value is **Yes**. To set the value to **No**, you must first close the Modify Choice Property Definition dialog window.

- To delete a choice, select it in the list, and then click the **Delete** button.



The **Delete** button is unavailable if the selected choice is included in the default value. After you remove the choice from the default value, the button is available and you can delete the choice.

The deleted choice is marked with arrow brackets (< >) and highlighted. The selection moves to the next lower choice in the list, or the deleted choice remains selected if it occupies the last position.

You can reverse this action until you close the dialog window. To change the **Delete** button label to **Undelete**, select the deleted choice in the list. Then, click **Undelete** to remove the arrow brackets and highlighting from the choice.



A choice cannot be deleted if it currently appears in the property value for any object. This condition applies to all existing objects, including those in users' Recycle Bins.

- o If the choice is used only by objects that are not irreversibly frozen, the choice must first be removed from all occurrences of the property value. Then, the choice can be deleted.
- o If the choice is used by a baselined object or a superseded frozen object, the property value cannot be edited and the choice cannot be deleted.



Database synchronization problems may occur if a deleted choice is used in existing object templates, live Excel spreadsheets, or live Visio diagrams. Search results may not be returned by saved searches that specify a deleted choice in the criteria.

4. Click **OK** to close the dialog window.

- If you did not delete a choice, the **Properties** tab or window displays your changes. You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.
- If you deleted a choice, a confirmation message states that this operation cannot be undone and asks if you want to continue.
 - o To apply your changes in the **Properties** tab or window, click **Yes**.



This action clears the queue for the **Undo** option and cannot be reversed.

- o To cancel all changes, click **No**.

To continue modifying the choice definition, you can specify that the value must contain at least one choice in the Systems Engineering module. First, set a default value for the choice definition as described in step 4-3. Then, double-click the **Input Required** property value to display the Single-Choice dialog window, and check the **Yes** checkbox to require that a value must be entered.

The **No** checkbox is checked by default, which allows the choice property value to be blank and allows users to clear existing choices. When the **Input Required** property is changed from **No** to **Yes**, the value of each occurrence of the choice property is retained in the Systems Engineering module. The next time each value is edited, at least one choice is required. Otherwise, the previous value is retained and an error message is displayed.

Procedure Notes

Step 1: To open subfolders, click the plus signs, or double-click a folder with a plus sign. You can also select a folder and then pull down the **View** menu and choose **Expand** or **Expand All**. Or, right-click the folder and choose **Expand** or **Expand All** from the popup menu.

Setting a Dynamic Choice List for a Choice Property Definition

The choice list of a choice property definition can be updated dynamically through a system-defined activator subtype, **Picklist**. Activators are used to execute scripts written in Tool Command Language (Tcl). When a **Picklist** activator is applied to a choice definition, the script result automatically populates the choice definition's **Choice List** property value.

For example, assume that a choice definition named **Assign To** has a choice list of the users in your project. Also, assume that a **Picklist** activator named **Update Users** contains the following Tcl script:

```
set project $currentProject
set members [getList $project USER_LIST]
```

In this example, the script returns a list of all users in the project. When the **Update Users** activator is applied to the **Assign To** choice definition, the choice list is automatically modified when a user is added, renamed, or removed.

The choice definition can be either single-choice or multiple-choice. By applying the choice definition to a type definition, changes to the choice list are made immediately available for selection on those objects in the Systems Engineering module.



- The **Choice List** value cannot be edited manually when a **Picklist** activator is applied to the choice definition. All choices are set automatically by the activator script result.
- A default value cannot be set for the choice definition.
- The choice list cannot be rearranged or sorted.

To create a Picklist activator:

1. Select the **Activators** folder or subfolder where you want to create the activator.
2. Pull down the **File** menu and choose **New**→**Activator Subtype**→**Picklist**.
The content table displays the activator with a default name in an open text field.
3. Enter the activator name, and then press the enter key.



The desired Tcl script must be coded in the activator. Sample scripts are provided in the following **Picklist** activators:

-

Choice List returns a list of the folders in the project.

-

User returns a list of the users in the project.

-

User Group returns a list of the users and user groups in the project.

These **Picklist** activators are included in the **Activators** folder of the Systems Architect/Requirements Management Administration project. This makes them available for use in any project. You can open a script in Microsoft Notepad by selecting the activator, pulling down the **File** menu, and choosing **Open**. Or, you can double-click the **Script** value in the **Properties** tab to open the script in a text field. For more information about using activators, see the *Systems Architect/Requirements Management API Reference* manual.

Procedure Notes

Step 2: In the navigation tree, you can also right-click the folder and choose **New→Activator Subtype→Picklist** from the popup menu. In the content table, you can also right-click any object and choose **New→Picklist** from the popup menu.

To apply a Picklist activator to a choice property definition:



- This procedure clears all instances of the property value if the property is used by existing objects and if a **Picklist** activator is not already applied.
- If you change from one **Picklist** activator to another, existing objects that use the property retain their current values. The new activator's choice list is available the next time each property instance is edited.

1. In the **Property Definitions** folder or a subfolder, select the choice definition to display its properties in the **Properties** tab.
- 2.

In the **Value** column, double-click the value for the **Update Choice List Dynamically** property.

The Single-Choice dialog window displays the **Picklist** activators in the project.



The following **Picklist** activators are included by default:



Choice List returns a list of the folders in the project.



User returns a list of the users in the project.



User Group returns a list of the users and user groups in the project.

3. Check the checkbox for the activator, and then click **OK**.

The script result populates the **Choice List** property value.

Procedure Notes

Step 1: Instead of the **Properties** tab, you can use the **Properties** floating window by clicking the **Open tab** button on the notebook pane toolbar. Or, pull down the **File** menu and choose **Properties** to use the Edit Properties dialog window.

To remove a Picklist activator from a choice property definition:



This procedure clears all instances of the property value where the property is used by existing objects.

1. In the **Property Definitions** folder or a subfolder, select the choice definition to display its properties in the **Properties** tab.
- 2.

In the **Value** column, double-click the value for the **Update Choice List Dynamically** property.

The Single-Choice dialog window displays the **Picklist** activators in the project.

3. Clear the checkbox for the activator, and then click **OK**.

Procedure Notes

Step 1: Instead of the **Properties** tab, you can use the **Properties** floating window by clicking the **Open tab** button on the notebook pane toolbar. Or, pull down the **File** menu and choose **Properties** to use the Edit Properties dialog window.

To apply a choice property definition to a type definition:

1. In the **Type Definitions** folder or a subfolder, select the type definition to display its properties in the **Properties** tab.
2. In the **Value** column, double-click the value for the **Properties** property.
The Multi-Choice dialog window displays the properties in the project.
3. Check the checkbox for each property to apply, and then click **OK**.

Step 1: Instead of the **Properties** tab, you can use the **Properties** floating window by clicking the **Open tab** button on the notebook pane toolbar. Or, pull down the **File** menu and choose **Properties** to use the Edit Properties dialog window.

Modifying a Date Property Definition

You modify a date definition to set the property value's valid format and default value. When a user double-clicks the property value in the Systems Engineering module, the valid format appears below the text field.

To modify a date property definition:

1. In the **Property Definitions** folder, select the date definition, or open the containing subfolder and then select the date definition.

The **Properties** tab or window displays all viewable properties for the date definition.

2. In the **Value** column, double-click the value for the **Current Value**, **Date Format**, **Time Flag**, or **Time Format** property.

The Modify Date Property Definition dialog window is displayed.

3. Do any or all of the following:

- To set the valid format for the date, click one of the choices under **Choose a date format**.
- To set the valid format for the time of day, do one of the following:
 - o Click one of the choices under **Choose a time format**.
 - o Clear the **Including Time** checkbox.
This action specifies that the format does not include the time of day.

-



The default value for date properties can be set to **Today**. Setting the default value to **Today** initializes instance of the property to the day they are created. If **Include Time** is selected, the time of creation is also set.

If **Include Time** is not selected, only the date is displayed on the property instances. However, a time value is still stored in the database. The time for such property instances is initialized to noon GMT. This provides consistency and prevents time zone adjustments from shifting the time to a different day. If **Include Time** is selected later, the noon GMT times becomes visible for existing property instances. The time is adjusted to the local time zone when displayed.

To set the default value of the date definition, do one of the following:

- o In the text field at the bottom of the dialog window, delete the current value and enter the new value.

The format must match the selected date format and time format.

- o

To set the value to today's date, click the **Today** button.

- o

To mark the value for setting at a later date, click the **TBD** button.

4. Click **OK**.

The **Properties** tab displays your entries.



If you set **TBD** as the default value, each instance of the property value is **TBD** until the instance value is edited or the default value is changed.

Procedure Notes

Step 4: To reverse this action, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Modifying a Numeric Property Definition

Modify a numeric definition to set the valid format, default value, and formula for calculation. When a user double-clicks the value in the Systems Engineering module, the valid format is shown below the text field. When a user calculates the value for a selected parent object, the formula operates on the value for each direct child.

To modify a numeric property definition:

1. In the **Property Definitions** folder, select the numeric definition, or open the containing subfolder and then select the numeric definition.
2. In the **Value** column of the **Properties** tab or window, double-click the value for the **Current Value**, **Format**, or **Formula** property.
3. In the Modify Numeric Property Definition dialog window, do any or all of the following:
 - To set the valid format, select **Normal**, **Percentage**, or **Scientific Notation** under **Choose display format**.
 - In the **Decimal Places** field, enter the number of digits allowed to the right of the decimal point.
 - To set the default value, enter the value in the **Enter default value** field.
The default value format must match the selection under **Choose display format**.
 - To use a formula for calculating the property value automatically, select one of the following in the **Formula** field:

Average calculates the sum of the values divided by the number of direct children.

Maximum calculates which value is the highest among all direct children.

Minimum calculates which value is the lowest among all direct children.

Multiply calculates the product of the values for all direct children.

Sum calculates the total of the values for all direct children.

For more information about calculating numeric property values, see the *Systems Architect/Requirements Management User's Manual*.

4. Click **OK** to close the dialog window and apply your changes.

Procedure Notes

Step 1: To open subfolders, click the plus signs, or double-click a folder. In the navigation tree, you can select a folder and then pull down the **View** menu and choose **Expand**, or right-click the folder and

choose **Expand** from the popup menu. In the content table, you can select a folder and then pull down the **View** menu and choose **Expand All**, or right-click the folder and choose **Expand All** from the popup menu.

Step 4: To reverse this action, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Modifying a Text Property Definition

A text property value consists of text that users enter in the Systems Engineering module. By modifying the corresponding text property definition, you can define the type of text value and you can set the default value for the property.

The value can contain text of one of the following types:

- Plain text, the default text type. There is no limit on the length of the text string. Users can enter different text strings for individual objects where the property applies in the Systems Engineering module.

-

Hypertext, for example, a URL for a Web address or a network file location. In the Systems Engineering module, the value is formatted as a hyperlink in the content table and the **Properties** tab or window. By holding down the control key and clicking the hyperlink, users can navigate to the URL destination in Microsoft Internet Explorer or Microsoft Windows Explorer.

You can include the property value in Microsoft Excel and Microsoft Word export files for hyperlink navigation to the URL destination. For more information, see [Modifying an Object Template](#) and [Modifying an Excel Template](#) in chapter 4, *Customizing the Interface With Microsoft Office*.

A unique URL can be entered for each object where the property applies in the Systems Engineering and Requirements Management module. Hyperlink navigation is disabled while the URL is edited.



The URL is not checked for validity. Therefore, users must ensure that the URL is correct and that it conforms to syntax requirements. For example, a file location must be entered in the long UNC path format. Also for example, a Web address may require a preface such as **http://**, **https://**, or **ftp://**.

For both text types, you can specify a default value or leave the default value blank.

To modify a text property definition:

1. In the **Property Definitions** folder, select the text definition, or open the containing subfolder and then select the text definition.

The **Properties** tab or window displays all viewable properties for the text definition.

2. Do one or both of the following:

- To define the type of text value:

In the **Value** column, double-click the value for the **Content Type** property to display the Single-Choice dialog window.

- Do one of the following:

- o For plain text, check the **String** checkbox.
- o For hypertext, check the **URL** checkbox.

- Click **OK** to close the dialog window and set the value type.

- To set the default value:

In the **Value** column, double-click the value for the **Current Value** property.

The Modify Text Property Definition dialog window is displayed.

Enter the default value in the text field, and then click **OK** to close the dialog window and set the property value.



- For a hypertext default value, ensure that the URL is correct and that it conforms to syntax requirements. A file location must be entered in the long UNC path format. Also, a Web address may require a preface such as **http://**, **https://**, or **ftp://**.
- The value for the **Current Value** property is not formatted as a hyperlink in the Administration module.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Procedure Notes

Step 1: To open subfolders, click the plus signs, or double-click a folder with a plus sign. In the navigation tree, you can also select a folder with a plus sign and then pull down the **View** menu and choose **Expand**, or right-click the folder and choose **Expand** from the popup menu. In the content table, you can also select a folder with a plus sign and then pull down the **View** menu and choose **Expand All**, or right-click the folder and choose **Expand All** from the popup menu.

Step 2: In setting the default value, you can enter new text, and you can change or delete any existing text.

Updates to Change Time and Change User Properties

The **Change Time** and the **Change User** properties record only the significant changes to an object. It is not intended to record every change to an object. The significance of changes varies by the object type. The business processes built around the Architect/Requirements also have an impact on what is considered significant. However, the rules cannot be configured.

The **Change Time** and the **Change User** properties are not updated by the following events:

- Setting of most properties, including **Name**.
- Adding or removing a child of a requirement or a building block.
- Moving an object to a new owner, by dragging it or by **Cut** and **Paste**.

Neither the moved object nor the new owner is updated, except the folders are updated when a member is added/removed.

- Adding or remove an attachment, such as a note or a diagram.

Table 2-1 lists the triggering events that cause the **Change Time** and the **Change User** properties to be updated. These triggering events vary by the object type.



These triggering events may change from time to time, as new functionalities are added, and based on the customer feedback.

Table 2-1. Triggering Events for Updates to Change Time and Change User Properties

Object Type	Triggering Events
All objects	<ul style="list-style-type: none"> ● Add/Remove a defining or a complying trace link. ● Delete or move to the recycle bin. ● Restore from the recycle bin. ● Change subtype. ● When passing through the change management process and setting. It includes events such as users to approve, users notified, users already approved, and users rejected.
Folder	<ul style="list-style-type: none"> ● Add/Remove a member. ● Set the icon overlay. ● Set the document template.
Requirement and Paragraph	<ul style="list-style-type: none"> ● Edit the text. ● Freeze/Unfreeze/Baseline. ● Set the text format property.
Building Block	<ul style="list-style-type: none"> ● Freeze/Unfreeze/Baseline.

Table 2-1. Triggering Events for Updates to Change Time and Change User Properties

Object Type	Triggering Events
	<ul style="list-style-type: none">● Set the numbering property (not the number).
Note	<ul style="list-style-type: none">● Edit the text.● Set the text format property.
Diagram	<ul style="list-style-type: none">● Save.● Set the data dictionary.
Connection	<ul style="list-style-type: none">● Set the data definition.
Port	<ul style="list-style-type: none">● Set the data definition.● Set the direction.
Spreadsheet	<ul style="list-style-type: none">● Save.● Set the file type.

Chapter 3: Customizing Object Types

This chapter contains an overview of type definitions and instructions for using subtypes to extend built-in object types.

Overview of Type Definitions

A type definition specifies the properties that apply to all objects of a given type. In turn, these properties determine the nature and behavior of the related objects.

For every new project, Architect/Requirements generates type definitions automatically in the Administration module. All projects receive the same type definitions, named **Folder**, **Requirement**, **Building Block**, **Group**, **Note**, **Trace Link**, **User Group**, **Connection**, **Diagram**, **Spreadsheet**, **Change Approval**, and **Change Log**. Through particular system-defined properties, each of these type definitions governs a built-in object type, or *base type*.

The project administrator can customize a base type by modifying the properties of the related type definition. Default values can be changed for editable system-defined properties, and user-defined properties can be applied to the base type definition.

By creating custom type definitions, or *subtypes*, the project administrator can extend the built-in object types for specialized purposes. From an existing type definition, any number of subtypes can be created for a given object type. Different properties and behavior can be associated with each subtype according to its purpose.

A subtype inherits the base type of the originating type definition, or parent, and also inherits the parent's system-defined and user-defined properties. Though the base type cannot be changed, a subtype can be distinguished by unique properties appropriate to its purpose. The project administrator can create specific user-defined properties for the subtype and add them to its property set. Inherited user-defined properties can be modified or removed to fit the purpose. System-defined properties also can be modified, if editable, but they cannot be added or removed.

Subtypes can be created from base type definitions and from other subtypes. The content table shows the subtype hierarchy in the **Type Definitions** folder. Once a subtype is created in the Administration module, users can assign the subtype when creating new objects of that base type in the Systems Engineering module. Users can assign the new subtype to existing objects by editing the **Subtype** properties. A system-defined folder subtype (**Document**), a system-defined requirement subtype (**Paragraph**), and a system-defined building block subtype (**TRAM**), can be assigned to new and existing objects of the related type.

Customized type definitions are specific to the project. Other projects in the same Architect/Requirements installation may be customized for different purposes, processes, or products.

Creating a Subtype

In each new project, Architect/Requirements automatically creates type definitions for the built-in object types. To customize those object types, you create subtypes, which are custom type definitions based on the system-defined type definitions.

Subtypes allow you to apply different properties among built-in objects of the same type. For example, you may create requirement subtypes named Customer Requirements and Derived Requirements, to distinguish requirements by specific purposes. Then you would apply a unique set of user-defined properties to each new subtype. For more information, see [Modifying the Properties of a Type Definition](#), later in this chapter, and chapter 2, [Unsatisfied xref title](#).

Each subtype inherits the system-defined properties of the parent type definition on which the subtype is based. For more information about system-defined properties in the Systems Engineering module, see the *Systems Architect/Requirements Management User's Manual*.

In addition, each subtype inherits the security profile named in the **Instance Security Profile** property of the parent type definition. For more information, see [Assigning a Default Security Profile to New Objects of a Type](#) in chapter 6, *Maintaining Project Security*.

After you create the subtype, users can assign it to new objects of that base type in the Systems Engineering module. For existing objects, users can assign the new subtype by changing the **Subtype** property values.

To create a subtype:

1. In the navigation tree or the content table, open the **Type Definitions** folder for the project.
The content table displays the base type definitions. In the **Types/SubTypes** column, a plus sign (+) is shown for each parent type definition with one or more subtypes at lower levels. To see lower level subtypes, click the plus signs.
2. Select the parent type definition for the new subtype, pull down the **File** menu, and choose the **New→Subtype** options.
The content table displays the new subtype at the next level below the parent, with a default name in an open text field.
3. Enter the subtype name, and then press the enter key.

Procedure Notes

Step 2: You can also right-click the parent and choose **New Subtype** from the popup menu.

Modifying the Properties of a Type Definition

After creating a subtype, you can apply a unique set of user-defined properties to customize the object type. When users create those objects in the Systems Engineering module, the new objects receive the properties that you apply to the type definition.



All existing objects of the type are updated with the changes. The extent of this transaction depends on the number of objects that are updated.

If a large number of objects is involved, Siemens PLM Software recommends one of the following:

- Perform this procedure during times of low system usage.
-

The Architect/Requirements system administrator set the **MessageQueue.Start** parameter for background processing. For more information about the **MessageQueue.Start** parameter, see the *Systems Architect/Requirements Management System Administrator's Manual*.

For example, a custom object template can be assigned to each type definition through its **Object Template** property. Also, a default security profile can be assigned to new objects through the **Instance Security Profile** property of the type definition. For more information, see [Object Templates](#) in chapter 4, *Customizing the Interface With Microsoft Office*, and [Assigning a Default Security Profile to New Objects of a Type](#) in chapter 6, *Maintaining Project Security*.

User-defined properties can be added and removed for any type definition, including the base type definitions. System-defined properties cannot be added or removed.

To modify the properties of a type definition:

1. In the navigation tree or the content table, open the **Type Definitions** folder.

The content table displays the base type definitions. To see lower level subtypes in a hierarchy, click the plus signs in the **Types/SubTypes** column.

2. In the content table, select the type definition.

The properties table displays all viewable properties for the type definition.

- 3.

In the **Value** column, double-click the **Properties** value.

Depending on the property type, the Single-Choice or Multi-Choice dialog window is displayed.

4. Do one or both of the following:

- To add an unapplied property, check the checkbox.
- To remove an applied property, clear the checkbox.



This action disables all existing reference links to all instances of the property. Each disabled reference link is marked with the label **[Broken Link: contents]**. The *contents* variable represents the last actual value of the property before the reference link was disabled. For more information about reference links, see the *Systems Architect/Requirements Management User's Manual*.

In the Multi-Choice dialog window, you can click **Select All** to add all properties simultaneously. Or, click **Unselect All** to remove all properties simultaneously.

5. Click **OK**, or press the enter key.

A confirmation message states that this operation updates all instances of this object type.



The updates are processed in the background if the **MessageQueue.Start** parameter is set to **true**.

6. To continue, click **Yes**, or press the enter key.

The changes are shown in the **Properties** value for the type definition. Architect/Requirements processes all instances of the object type and applies the updates.



If a large number of objects are involved, you can perform other actions while the updates are processed in the background. A message is displayed when updates are complete. Any instances where the update process failed are indicated.

Property updates may take a few seconds to many minutes, depending on the number of objects of that type and how often the background queue is set to run. Even when there are only a few objects to be updated, it may be a minute or more before the queued background update begins.

The objects are collected for update using a query, so all objects in a folder may not be processed at the same time. Update of some objects may also be delayed if users are modifying them at the time they are first processed.

As a result, users may open a folder and see some objects that have the property changes applied and other objects where the old properties are still in effect.

Procedure Notes

Step 6: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

The **Undo** action is applied immediately to the **Type Definition's Properties** list. However, individual objects of that type are updated in the background as described [above](#). These updates may take some time to complete, so some objects reflect the **Undo** action before others.

Customizing the Object Type Indicator for a Type Definition

You can modify the object type indicator with a custom icon for type definitions (and their subtypes) in the Administration module.

The recommended graphic size is 16 pixels by 16 pixels, which is also the smallest allowable graphic size. Larger icons can be used (up to 100 pixels x 100 pixels), however, their heights are clipped to the row height of the view they appear in. Also, the placements of the icon overlays, such as **Frozen** and **Shortcut**, are placed assuming a height of 16 pixels, and not the actual height.

The object type indicator for a selected child object is determined by the value of the child's **Icon** property:

- If the **Icon** value is the default graphic, the child inherits its object type indicator from its parent. When the parent's graphic is changed, that same graphic is automatically applied to the child.
- If the **Icon** value is a graphic other than the default, the child retains that object type indicator regardless of changes to the parent's graphic. When the child's graphic is changed, the graphic for the parent is not affected.

To customize the object type indicator for a type definition:

1. In the navigation tree or the content table, open **Type Definitions** folder for the project.

The content table displays the base type definitions. To see lower level subtypes in a hierarchy, click the plus signs in the **Types/SubTypes** column.

2. In the content table, select the type definition.

The properties table displays all viewable properties for the type definition.

- 3.

In the **Value** column, double-click the **Icon** value.

Architect/Requirements displays the **Open** dialog window, which lists existing graphic files in the current drive or folder. If the list does not display the graphic that you want, use the **Look in** field to change the drive or folder, or use the **Files of type** field to display other types of files.

4. Select the graphic, and then click **OK** to close the dialog window.

The new graphic is displayed in the **Icon** property value.

Setting the Default View for a Folder Type Definition

In the Systems Engineering and Requirements Management module, users can apply saved views to folders selected in the navigation tree. The view for each folder determines the display of property columns in the content table. The default view that you set for a folder type definition determines the initial view for new and existing folders based on that type definition.

A user can locally modify the default view by adding, removing, resizing, or sorting columns. Modifications are recorded on the user's computer and persist until the user reverts to the default view. Also, a user can select a different view in the **View** field, and the selection is recorded locally.



- When you create a folder type definition, it inherits the default view of the parent type definition. You can then set a different default view. For more information, see [Creating a Subtype](#), earlier in this chapter.
- An existing subtype's default view is not affected if you set a different default view for the parent.

To set the default view for a folder type definition:

1. Make the view visible to all users in the project by doing the following:
 - a. In the **Reports and Formatting** folder, select the view object, and then click the **Properties** tab to display the view's properties.
 - b. In the **Value** column, double-click the **Shared State** value.
 - c. In the Single-Choice dialog window, check the **Public** checkbox and click **OK**.
2. In the **Type Definitions** folder, select the folder type definition.
The **Properties** tab displays the type definition's properties.
3. In the **Value** column, double-click the **Default View** value.
4. In the Single-Choice dialog window, do one of the following:
 - a. To specify a default view, check the checkbox for the view and click **OK**.
 - b. To remove the current default view setting, clear the checkbox and click **OK**.



While a view is set as the default, its **Shared State** property value cannot be changed. Furthermore, a view object cannot be deleted if it is the default view for a folder type definition, a folder, or a user object. Before changing the **Shared State** value or deleting the view, you must remove all uses of the view by other objects. For more information, see [Removing References to a Schema Object](#) in chapter *Unsatisfied xref reference*, *Unsatisfied xref title*, and appendix [System-Defined Properties in the Administration Module](#).

Procedure Notes

Steps 1 and 2: Instead of the **Properties** tab, you can use the **Properties** floating window by clicking the **Open tab** button on the notebook pane toolbar. Or, pull down the **File** menu and choose **Properties** to use the Edit Properties dialog window.

Customizing the ROIN for a Requirement Type Definition

A Requirement Object Identification Number (ROIN) identifies each requirement within a project. The **ROIN** property, a read-only system-defined property, displays the ROINs in the Systems Engineering and Requirements Management module, providing a continuous numbering scheme. In the Administration module, you can assign custom prefixes and formatting to ROINs through the **ROIN Counter** property of requirement subtypes.



- ROINs assigned by Architect/Requirements are unique within a project.
- Custom ROINs can be duplicated within a project. Architect/Requirements does not check for duplicate prefixes or numbers.

An editable text property, the **ROIN Counter** property specifies the following:

- An alphanumeric prefix for the ROIN.
- The minimum number of digits in the numeric portion of the ROIN.
Numbers with less than the minimum digits are padded with leading zeros.
- A starting number.

For example, **REQ-0100** specifies the following:

- **REQ-** as the prefix.
- A minimum of four digits in the numeric portion.
- **100** as the starting number.

The **Requirement** base type has a default **ROIN Counter** value of **0001**, with no prefix, four minimum digits, and a starting number of **1**. When a subtype is created, its **ROIN Counter** value is blank by default. A blank value indicates that the subtype uses the base type counter.

Though all subtypes can share the base type counter, you can customize the ROIN for a subtype by changing the subtype's **ROIN Counter** property value. Thereafter, the subtype uses the new counter and cannot revert to the base type counter.

The subtype's current **ROIN Counter** value becomes the **ROIN** property value for the next new requirement of that subtype. When the requirement is created, the **ROIN Counter** value increases by one, for example, from **REQ-0100** to **REQ-0101**.



Siemens PLM Software recommends that you customize ROINs for all requirement subtypes before any requirements are created, and that you do not subsequently change the counters. However, **ROIN Counter** properties can be changed without affecting the ROINs of existing requirements. Such changes apply only to new requirements.



Architect/Requirements does not prevent **ROIN Counter** properties from being changed in ways that produce duplicate ROINs, such as assigning the same prefix to different subtypes. The project administrator is responsible for ensuring that ROINs remain unique within a project. To keep custom ROINs unique, do not change the prefix or the subtype.

To customize the ROIN for a requirement type definition:

1. In the navigation tree or the content table, open the **Type Definitions** folder.

The content table displays the base type definitions. To see other requirement subtypes in the hierarchy, click the plus sign to the left of the **Requirement** type definition and plus signs at lower levels.

2. In the content table, select the type definition.

The properties table displays all viewable properties for the type definition.

3. In the **Value** column of the properties table, double-click the **ROIN Counter** property value.

A text field opens in the cell.

4. Enter the custom prefix and format in the text field, and then press the enter key.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Chapter 4: Customizing the Interface With Microsoft Office

This chapter provides overviews and instructions for using formatting objects to customize data exported to Microsoft Office Word, Microsoft Office Excel, and Microsoft Office Visio.

Templates and Style Sheets for Export to Microsoft Office Word

In the Systems Engineering and Requirements Management module, project users can export object data from the database to Microsoft Office Word. Using a *document template* as a reference, Architect/Requirements generates a Word document containing data for each object that the user specifies.

The document template controls the export process through the following:

- Two *object templates*, each of which is assigned to several type definitions and determines the data that is exported for those object types.
- A *style sheet*, which determines the Word formatting that is applied to the data.

In the Administration module, several document templates, object templates, and style sheets are created automatically in the **Reports and Formatting** folder of every new project. The project administrator uses Word to work with object templates and style sheets. For more information about exporting objects to Microsoft Office Word, see the *Systems Architect/Requirements Management User's Manual*.

Document Templates

Architect/Requirements references a document template in generating each export document. In turn, the document template supplies instructions through the object templates and the style sheet that are associated with the document template.

Document templates can be used to customize the export process for a project. For example, the project administrator can modify the **Default Text Object Template**, the **Default Object Template**, and the **Default Style Sheet**, which are automatically associated with the **Default Document Template** in a new project.



The default document template is overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up a customized default template from the earlier version before upgrading to a later version.

Also, custom object templates can be assigned to type definitions through the **Default Document Template** or a custom document template, and custom style sheets can be associated with any document template. Users can choose different combinations of data and formatting for each export document. For more information, see [Creating a Document Template](#) and [Modifying a Document Template](#), later in this chapter.



Creating a Document Template

By its associated object templates and style sheet, a document template controls the Microsoft Office Word export process. You can customize that process for your project through unique document templates with object templates and style sheets that you create.

The **Default Text Object Template**, the **Default Object Template**, and the **Default Style Sheet** are automatically associated with the new document template. After creating the document template, you can change the object template assigned to one or more type definitions, and you can associate a different style sheet. For more information, see [Modifying a Document Template](#), [Object Templates](#), and [Style Sheets](#), later in this chapter.

Once the document template is created in the Administration module, users can choose the document template when exporting object data in the Systems Engineering and Requirements Management module.

To create a document template:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays the templates, style sheets, and reports in the project.

2. Pull down the **File** menu and choose the **New**→**Document Template** options.

The content table displays the document template with a default name in an open text field.

3. Enter the template name, and then press the enter key.

To refresh the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the **Reports and Formatting** folder and choose **Refresh** from the popup menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the **Reports and Formatting** folder and choose the **New**→**Document Template** options from the popup menu. Or, right-click in the content table and choose **New**→**Template**→**Document Template** from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

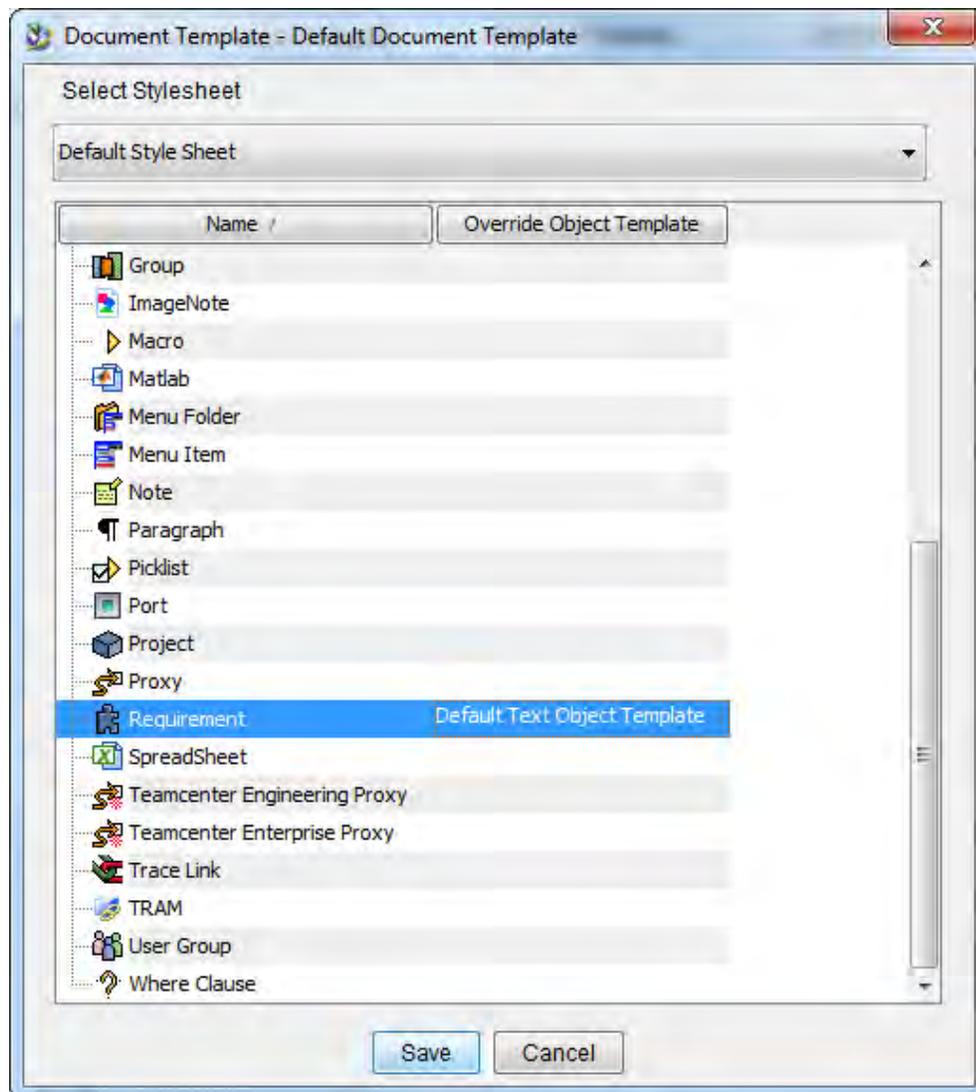
Modifying a Document Template

When you create a document template, the **Default Text Object Template**, the **Default Object Template**, and the **Default Style Sheet** are associated automatically. You can modify those associations to customize the new document template for a specific purpose in your export process.

 The default document template is overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up a customized default template from the earlier version before upgrading to a later version.

 If you use the **Object Template** property for requirements, they are overridden by the **Default Template** settings.

If the **Default Document Template** has the **Override Object Template** for Requirement set to **Default Object Text Template**, it overrides the object template property setting.



If the **Default Document Template** contains an override for the object, the **Default Document Template** is used for the preview. This is irrespective of the **Object Template** that you set for a **Type Definition**.

For built-in type definitions and custom subtypes, you can assign object templates that you create to export different data for different object types. Also, you can associate a custom style sheet with the new document template. In addition, you can modify the **Default Document Template** to use custom object templates and a custom style sheet. For more information, see [Creating a Document Template](#), earlier in this chapter, and [Object Templates](#) and [Style Sheets](#), later in this chapter.

To modify a document template:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays all templates, style sheets, and reports in the project.

2.

Select the document template, pull down the **File** menu, and choose **Open**.

The Document Template dialog window is displayed.

3. Do any of the following:

- To change the style sheet for the document template, select the new style sheet in the **Select Stylesheet** field.

-

To change the object template for a type definition in the **SubTypes** column:

- . Double-click the cell in the **Override Object Template** column to display the Single-Choice dialog window.



A blank cell in this column indicates that the **Default Text Object Template** or the **Default Object Template** is currently assigned, depending on the type definition.

- . Check the checkbox for the new object template, and then click **OK** to close this dialog window.



When data is exported with this document template, the object template that you choose overrides the one shown by the **Object Template** property for that type definition. For more information, see [Modifying the Properties of a Type Definition](#) in chapter 3, *Customizing Object Types*.

Repeat these steps for each additional type definition whose object template you want to change.



To display full content reference links as hyperlink URLs in the **Preview** tab and window:

- . Double-click the **Document Template Rules** property value to display the Single-Choice dialog window.
- . Check the checkbox for **Preserve Links**, and then click **OK** to close this dialog window.

For more information about reference links, see the *Systems Architect/Requirements Management User's Manual*.

4. To save your changes in the database and close the Document Template dialog window, click **Save**.

Procedure Notes

Step 2: You can also right-click the template and choose **Open** from the popup menu. Or, double-click the template.

Object Templates

An object template contains *tags* that represent object properties, whose values are extracted to the export document for each object that the user specifies. By default, one of two object templates is assigned to each type definition, including the system-defined subtypes for folders (**Document**) and requirements (**Paragraph**).



The **Default Text Object Template** is assigned to requirements, paragraphs, and notes. This template contains two tags. The **{%Name}** tag represents the **Name** property, whose value is an object name. The **{%HTML}** tag represents the **HTML** property, whose value is the full content of a requirement, paragraph, or note, including graphics and tables.

The **{%Name}** tag occurs in the template heading, which is formatted in Word's Heading 1 style, and the **{%HTML}** tag occurs in the template body. With this template, each object name is extracted to a heading in the document and the full content of the object follows in the body. Heading levels in the document depend on the way in which the user specifies the objects.



The **HTML** property is not displayed in the Administration or Systems Engineering and Requirements Management module because the value can exceed the effective size for table cells.



The **Default Object Template** is assigned to folders, building blocks, and groups. This template contains only the **{%Name}** tag. The tag represents the **Name** property, whose value is an object name, and occurs in the template heading, formatted in Word's Heading 1 style. With this template, each object name is extracted to a heading in the document. Heading levels in the document depend on the way in which the user specifies the objects.

The project administrator can modify the default templates to customize exported data for the related object types. Also, custom object templates can be created to export different data for different object types.



The default object templates are overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up customized default templates from the earlier version before upgrading to a later version.

In any object template, tags can be added for other properties to include that data in the export document, and tags can be deleted to exclude that data. Tags can be added to run activators when the export document is generated. By adding tags for notes, note content and properties can be appended to exported data.

Boilerplate can be created by inserting standard text, special keyboard characters, graphics, symbols, equations, and hyperlinks.

For more information, see [Creating an Object Template](#) and [Modifying an Object Template](#), later in this chapter.

Creating an Object Template

An object template determines the data that is exported to Microsoft Office Word for the type definitions to which the object template applies. Tags in the object template represent object properties, whose values are extracted to the export document for each specified object. You can customize the exported data by creating unique object templates, containing tags that you enter, and then assigning your object templates to type definitions through a document template.

Initially, a new object template contains only the **{%Name}** tag, which represents the **Name** property as in the **Default Object Template**. After you create the object template, you can add tags for any other properties that suit your purpose, and you can add tags for activators and notes. Also in the object template, you can create boilerplate for the export document. For more information, see [Object Templates](#) earlier in this chapter, and [Modifying an Object Template](#), later in this chapter.

Before or after you modify the new object template, you can assign it to type definitions through the **Default Document Template** and through document templates that you create. For more information, see [Creating a Document Template](#) and [Modifying a Document Template](#), earlier in this chapter.



- You must ensure there are no empty headers while creating an object template. Export using the object template generates no content if there are empty headers.
- The content before the first header in the object template is not used during export. This behavior is similar to import wherein any content before the first header becomes a note.

To create an object template:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays the templates, style sheets, and reports in the project.

2. Pull down the **File** menu and choose the **New**→**Object Template** options.

The content table displays the new object template with a default name in an open text field.

3. Enter the template name, and then press the enter key.

To refresh the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the **Reports and Formatting** folder and choose **Refresh** from the popup menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the **Reports and Formatting** folder and choose the **New**→**Object Template** options from the popup menu. Or, right-click in the content table and choose **New**→**Template**→**Object Template** from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Modifying an Object Template

To customize exported data for an object type, you can modify the tags in the object template that is assigned to the type definition. The object template can be the **Default Text Object Template**, the **Default Object Template**, or an object template that you create. For more information, see [Object Templates](#) and [Creating an Object Template](#), earlier in this chapter.

Do not modify the default templates in the Administrator module. If you need to modify a template, copy the template and make the required changes in the copied template.



The default object templates are overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up customized default templates from the earlier version before upgrading to a later version.

In any object template, you can add tags for properties that you want to include in the export document. For example, you can add the **{%ROIN}** tag to extract the Requirement Object Identification Number (ROIN) of each requirement and paragraph that the user specifies. You can also do the following:

- Add tags that run activators when the export document is generated.
- Add tags for notes to append note content and properties to exported data.
- Delete existing tags to exclude that data from the document, and copy or move tags within the template.
- Create boilerplate by inserting standard text, special keyboard characters, graphics, symbols, equations, and hyperlinks.

An object template can contain any number of tags. They can be entered anywhere in the template, in elements such as headings, body paragraphs, lists, and tables. The elements can be formatted in any available Word styles. However, headings must be formatted in Word's built-in heading styles.



To generate headings in the document, Architect/Requirements uses only the first heading style in the template. Data for all tags in that heading is exported to a heading for each object. Data for tags in all other elements is exported to the body, regardless of the formatting style.

As in the default templates, tags can be set apart in separate elements. Tags also can be entered in conjunction with boilerplate, at any location within the same element. For example, brackets ([]) can be inserted around the **{%ROIN}** tag to enclose each ROIN with brackets in the document.

However, if you have an Outline Level (Heading Style) in the object template, it is treated as the initial object. Anything inserted before the heading row in the template is ignored. If the object template has no Outline Level (Heading Style) then everything is evaluated. If you want to insert an activator so that it is executed and its output appears before the heading, put the activator tag at the beginning of the heading row. For example:

{%Activator%ConditionalPageBreak}{%NAME}

{%HTML}

After you save your changes, Architect/Requirements references the modified template for each exported object of that type. Although only one object template can be assigned to any type definition, different subtypes with different object templates can be exported to the same document.



- Microsoft Office Excel 2013, or Excel 2016 must be installed on your computer.
- Tools for checking spelling and grammar should be disabled in Word before you save the template.

To modify an object template:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays the templates, style sheets, and reports in the project.

2. Select the template, pull down the **File** menu, and choose **Open**.

The template opens in Word as an **.mhtml** file.

3. Do any or all of the following:



To add tags:

- o For an object property, enter `{%Property-Name}`.
- o For an activator, enter `{%Activator%Name}`.
- o For a note, enter `{%Note%Name%Property-Name}`.

The note should be attached to a requirement that is specified for export.

Replace *Property-Name* and *Name* with the exact name displayed in the Systems Engineering and Requirements Management or Administration module. All tags are case sensitive.



A property or note tag can contain any property name, whether user-defined or system-defined. For more information about system-defined properties, see appendix [System-Defined Properties in the Administration Module](#) and the corresponding appendix for the Systems Engineering and Requirements Management module in the *Systems Architect/Requirements Management User's Manual*.

- To delete, move, or copy existing tags, use the related Word functions.
- To create boilerplate, use the Word functions for inserting standard text, special characters, graphics, symbols, equations, or hyperlinks.
- To apply character and paragraph formatting, use the manual formatting features or the list of available styles in Word.



Formatting applied to an `{%HTML}` tag in the template has no effect in the export document. Because the referenced HTML text may contain several styles, the appearance of exported HTML is set by the formatting within the requirement, paragraph, or note content.

4. Save your changes in Microsoft Word.

Procedure Notes

Step 2: You can also right-click the template and choose **Open** from the popup menu. Or, double-click the template.

Style Sheets

A style sheet determines the Microsoft Office Word formatting that is applied to the data in the export document. Architect/Requirements creates the **Default Style Sheet** for every new project, in the **Reports and Formatting** folder. This style sheet is automatically associated with the **Default Document Template**.

The **Default Style Sheet** can be modified to change the attributes of existing styles and to create new styles. Also, custom style sheets can be created and associated with the **Default Document Template** and with custom document templates. In any style sheet, styles can be created and modified as in any other Word document. Styles can also be imported from an existing Word document. For more information, see [Creating a Blue Sheet](#) and [Modifying a Style Sheet](#), later in this chapter.



The default style sheet is overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up a customized default style sheet from the earlier version before upgrading to a later version.



The **Default Style Sheet** contains styles for Heading 1 through Heading 9. Siemens PLM Software recommends that you add those styles to style sheets from earlier Architect/Requirements versions and to custom style sheets in your current version.

Creating a Style Sheet

A style sheet determines the Microsoft Office Word styles in which data is formatted in the export document. You can specify that formatting in advance by creating a style sheet containing custom styles, and then associating your style sheet with a document template.

Initially, a new style sheet contains the same styles as the **Default Style Sheet**. After creating the style sheet, you can modify existing styles and create new styles as in any other Word document. You can also import styles from an existing Word document. For more information, see [Modifying a Style Sheet](#), later in this chapter.

Before or after you modify the style sheet, you can associate it with the **Default Document Template** and with a custom document template, by changing the current style sheet for the document template. For more information, see [Creating a Document Template](#) and [Modifying a Document Template](#), earlier in this chapter.

To create a style sheet:

1. With the **Reports and Formatting** folder open, pull down the **File** menu and choose the **New→Style Sheet** options.

The content table displays the style sheet with a default name in an open text field.

2. Enter the style sheet name, and then press the enter key.

To refresh the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the **Reports and Formatting** folder and choose **Refresh** from the popup menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 1: You can also right-click the **Reports and Formatting** folder and choose the **New→Style Sheet** options from the popup menu. Or, right-click in the content table and choose **New→Template→Style Sheet** from the popup menu.

Step 2: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Modifying a Style Sheet

When you create a style sheet, it initially contains the same styles as the **Default Style Sheet**. You can modify those styles in the new style sheet, and you can create styles to further customize the formatting in the export document. In addition, you can modify and create styles in the **Default Style Sheet**.



The default style sheet is overwritten when a new Architect/Requirements version is installed. Siemens PLM Software recommends that you back up a customized default style sheet from the earlier version before upgrading to a later version.

To create and modify styles automatically, you can import styles from an existing Word document. You can also open the style sheet in Word to enter and change styles directly, as in any other Word document.



Microsoft Office Excel 2013 or Excel 2016 must be installed on your computer.

To import styles to a style sheet:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays the templates, style sheets, and reports in the project.

2. Select the style sheet, pull down the **File** menu, and choose **Import Style Sheet**.

A new Word session starts with the Open dialog window, which lists existing folders and files in the current drive or folder. If the list does not display the document that contains the styles, use the **Look in** field to change the drive or folder, or use the **Files of type** field to display other types of files.

3. In the list, select the document that contains the styles, and then click **Open**.

The document opens in Word, and a message states that Architect/Requirements is importing the file. You can click **OK** to close this message and perform other actions while the import is in progress. When the import is complete, the document closes and a confirmation message is displayed.

Procedure Notes

Step 2: You can also right-click the style sheet and choose **Import Style Sheet** from the popup menu.

To enter or change styles directly in a style sheet:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.

The content table displays the templates, style sheets, and reports in the project.

2. Select the style sheet, pull down the **File** menu, and choose **Open**.

The style sheet opens in Word as an **.mhtml** file. Through Word's **Format** menu, you can display the **Styles and Formatting** task pane to create, modify, or delete styles.

3. To save your changes in the database, pull down Word's **File** menu and choose **Save**, or click the **Save** button on Word's toolbar.

Procedure Notes

Step 2: You can also right-click the style sheet and choose **Open** from the popup menu. Or, double-click the style sheet.

If the information required to view graphics in a browser is missing in the **Default Style Sheet**, the graphics do not appear in the **Preview** tab. This condition occurs when a project or schema is imported from TcSE version prior to TcSE 8.0. To ensure that the graphics appear in the **Preview** tab, use schema import to obtain a more recent copy of the **Default Style Sheet**.

To update the Default Style Sheet to support graphics:

1. In the administration module, select the **Reports and Formatting** folder.
2. Select the **Default Style Sheet** and open it for edit in Microsoft Word.
3. Insert a graphics file into the document.
Select **Insert** → **Picture** → **From File**.
4. Exit Word and save.
5. Open the **Default Style Sheet** in Word again.
6. Select the graphic and delete it.
7. Exit Word and save.

Templates for Export to Microsoft Office Excel

When users export objects to Microsoft Office Excel, Architect/Requirements references an *Excel template* in extracting data from the database to an Excel spreadsheet. To customize the data that is exported, the project administrator enters *tags* that represent data values in the template's property columns. In the template's *rule table*, the project administrator enters *key fields* that filter the tag data according to specific criteria. The rule table also filters data for reports that are output to Excel from the Search module. For more information about exporting objects to Microsoft Office Excel or using the Search module, see the *Systems Architect/Requirements Management User's Manual*.

The **Default Excel Template** is created automatically in the **Reports and Formatting** folder for every new project in the Administration module. Using Excel, the project administrator can modify the **Default Excel Template** and can create and modify custom Excel templates for specialized purposes.

Creating an Excel Template Object

After you create an Excel template object, you can delete default data tags in the new template, and you can add tags for other properties and for activators. You can enter key fields in the rule table to define custom criteria for exported data. In addition, you can create boilerplate to include with the tag data. Users can choose the template in the Systems Engineering and Requirements Management module when they export objects to Excel. For more information, see [Modifying an Excel Template](#), later in this chapter.

To create an Excel template object:

1. In the navigation tree or the content table, open the **Reports and Formatting** folder for the project.
The content table displays the templates, style sheets, and reports in the project.
2. Pull down the **File** menu and choose the **New**→**Excel Template** options.
The content table displays the new Excel template with a default name in an open text field.
3. Enter the template name, and then press the enter key.
To refresh the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the **Reports and Formatting** folder and choose **Refresh** from the popup menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 2: You can also right-click the **Reports and Formatting** folder and choose the **New**→**Excel Template** options from the popup menu. Or, right-click in the content table and choose **New**→**Template**→**Excel Template** from the popup menu.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Modifying an Excel Template

For an Excel template selected in the **Reports and Formatting** folder, you can open the template in Excel to enter changes directly, and you can import content from an Excel template that is stored on a local or shared drive. For more information, see [Excel Template Modification Concepts](#), later in this chapter.



- If the template contains multiple worksheets and if the **<start>** tag and the **<end>** tag are not entered on a worksheet, all entries on that worksheet are exported as constant values.
- If a rule is duplicated on two or more worksheets, duplicate data is included in the export file.
- If the template has the **<Start>** tag in the first row, a **-1** error is displayed. Ensure that the **<Start>** tag is in the **A2** cell or below in the Excel worksheet.

To enter changes directly in an Excel template:

1. In the **Reports and Formatting** folder, select the Excel template, and then pull down the **File** menu and choose **Open**.

You can also right-click the template and choose **Open** from the popup menu. Or, double-click the template.

2. In the **.xlsm** Excel file, do any or all of the following:



To add tags in property columns:

- o For an object property, enter **{%Property-Name}**.

Replace *Property-Name* with the exact name. The tag is case sensitive. A property tag can name any user-defined or system-defined property. For more information about system-defined properties, see appendix [System-Defined Properties in the Administration Module](#) and appendix B in the *Systems Architect/Requirements Management User's Manual*.



In a cell with a tag for a date property, apply an Excel **Date** format. Without a **Date** format, the exported value is the Excel numeric equivalent of the date. You can apply a **Date** format through Excel's Format Cells dialog window.

A **Date** format is applied by default to the **Create Time** property, which is included automatically in the **Default Excel Template** and in new Excel templates that you create. Siemens PLM Software recommends that you apply a **Date** format to all date properties that you add in any Excel template, and to date properties in templates from older versions of Architect/Requirements.

- o For an activator, enter **{%Activator%Name}**.

Replace *Name* with the exact name. The tag is case sensitive. For more information, see [Activator Tag Results](#), later in this chapter.



To modify the rule table:

- o For a key field in the **Level** column, enter `{%L-n}`.
Replace *n* with the number that represents the level of the objects to which the rule applies.
- o For a key field in the **Relationship** column, enter `{%R-Relationship-Name}`
Replace *Relationship-Name* with the relationship to which the rule applies. The syntax must match that of a **Relationship** field option for an **ADD** statement. For more information about **ADD** statements, see the *Systems Architect/Requirements Management User's Manual*.
- o For a key field in the **Subtype** column, enter `{%S-Subtype-Name}`
Replace *Subtype-Name* with the name of the system-defined or user-defined type definition to which the rule applies.

For more information, see [Rule Table Concepts](#), later in this chapter.

- To delete, move, or copy existing tags or key fields, use the Excel functions.
- To create boilerplate, use the Excel functions for inserting elements such as standard text, special characters, symbols, or hyperlinks.
- To apply cell formatting, use the Excel formatting features.
- To use formulas, use the following guideline:

A formula in a row between the `<start>` and `<end>` tags is copied to the rows in the exported document that uses the rule row. The formula text is copied as it exists in the source. Explicit row references in the formula may not work as required as they are not adjusted. For example, to add numeric property values in columns B and C, the formula `=SUM(B4,C4)` does not work. You must use `=INDIRECT("B"&ROW())+INDIRECT("C"&ROW())` instead.

- To apply cell merges, use the following guideline:

Cell merges on a rule row are applied to exported rows that use the rule.

3. To commit your changes to the database, save the changes in Excel.

To import content to an Excel template:

1. In the **Reports and Formatting** folder, select the Excel template that you want to modify, and then pull down the **File** menu and choose **Import**→ **Excel Template**.

Architect/Requirements displays the Open dialog window, which lists existing folders and files in the current drive or folder. If the list does not display the import template, use the **Look in** field to change the drive or folder.



The import file must have **.xlsm** or **.xlsx** file name extension.

2. In the list, select the Excel template whose content you want to import, and then click **Open**.



This action overwrites the content of the template selected in step 1.

A message states that Architect/Requirements is importing the file. You can close this message and perform other actions while the import is in progress. When the import is complete, a confirmation message is displayed.

Procedure Notes

Step 1: You can also right-click the Excel template and choose **Import**→**Excel Template** from the popup menu.

Packing Multiple Objects on One Row

Grouping multiple identical objects in one level of a hierarchy is known as *packing*. When the **Level** key field of the rule table is used in conjunction with the **Excel Template Rules** property of the template, data for two or more objects can be placed on the same row of Microsoft Office Excel export files.

For more information, see [Modifying an Excel Template](#), earlier in this chapter, and [Rule Table Concepts](#) and [Data Placement in Export Files](#), later in this chapter.



This feature applies only if the user exports all objects in the view. If the user exports selected objects only, data for each object is placed on a separate row.

To pack multiple objects on one row:

1. In the **Reports and Formatting** folder, select the Excel template.
2. In the **Properties** tab or window, double-click the value for the **Excel Template Rules** property to display the Multi-Choice dialog window.
3.
Check the checkbox for **Apply Packing**.
4. In the **Level** column of the rule table, enter key fields according to the way you want to fill the rows in the export file.

Excel Template Modification Concepts

A new Excel template object initially contains the same property columns, tags, and rule table as the **Default Excel Template**. You can modify a new template to customize the data that is exported when users choose the template. Also, you can modify an existing custom template and the **Default Excel Template** itself.

To modify a template automatically, you can import content from an Excel template that is stored on a local or shared drive. You can also open a template in Excel to enter changes directly, as you do the following in any other Excel file:

- Add and remove columns for system-defined and user-defined properties that you want to include in the export spreadsheet.
- Add rows and tags for data that you want to export, delete existing rows and tags to exclude that data, and modify, copy, and move rows and tags.
- Add tags that run activators when the spreadsheet is generated.

An activator can return either of the following:

- The current value of a property, which may be associated with any object in the database.
- A constant value.

For more information, see [Activator Tag Results](#), later in this chapter.

- Create boilerplate above the **<start>** tag and below the **<end>** tag.

Boilerplate can include elements such as:

- Cell merges.
- Standard text.
- Special characters.
- Symbols.
- Hyperlinks.
- Formulas.



Microsoft Excel shifts the rows below the **<end>** tag downwards to accommodate exported objects. You must account for the shift in the rows in the formula that you write.

You can also enter those elements as constant values in the cells of property columns.



Boilerplate entered in or to the right of the rule table is not exported.

- Format cells containing tags and boilerplate to apply that formatting to the data in the corresponding cells of the export spreadsheet.
- Modify the rule table to define custom criteria for exported data.

For more information, see [Rule Table Concepts](#), later in this chapter.

-

Enter Visual Basic macros that users can run from the export file.



If a live Excel file that contains macros is opened on a computer where the live Office interface is not installed:

- o No message is displayed to warn that live Office is not installed.
- o The file cannot be connected to the database.
- o Changes in the spreadsheet are not accumulated and cannot be applied to properties in the database.

If you or other users plan to distribute such a file (for example, by E-mail), Siemens PLM Software recommends that recipients be notified of these conditions.

- Insert multiple worksheets to separate data in the export file.

Each additional worksheet initially contains empty cells. In the formats described in this procedure, you enter the tags, rule table, and boilerplate that you want to include on the corresponding worksheet in the export file. You can also enter Visual Basic macros and delete worksheets.



- o If the **<start>** tag and the **<end>** tag are not entered on each worksheet, all entries on that worksheet are exported as constant values.
- o If a rule is duplicated on two or more worksheets, duplicate data is included in the export file.
- o If you have a special character, such as **'**, *****, or **@**, in the name of a worksheet in an Excel template, the Excel workbook exported as live workbook with this template may not work as expected. This is because of the **xlsx** or **xlsm** encoding of the special characters. Static export has no such problem.

Activator Tag Results

In an activator tag, the activator named with the **{%Activator%Name}** syntax can return its result in either of two forms:

- A static string to be displayed in that cell of the exported Excel spreadsheet.

The returned string appears exactly as it does in the cell. Even if the spreadsheet is exported to live Excel, this text is not associated with any object or property.

- An object and property reference to create a *live* cell.

The activator script must be coded to return a string in the form **:nn.n.nnnn:propName:**, where **nn.n.nnnn** is the LOID of an object (in valid format) and **propName** is the name of a property on that object.

In the Excel export file, the cell displays the current value of that property on that object. If the spreadsheet is exported to live Excel, cells exported in this form are associated with the object and property and can be edited in the same way as other live cells.

For more information about activators, see the *Systems Architect/Requirements Management API Reference* manual.

Rule Table Concepts

The rule table begins with the column that contains a <rule> field in each cell. Each <rule> field delimits a rule that filters the tag data in the property columns to the left in the same row.

Key Fields

To the right of a <rule> field, three key fields are used to define the criteria that comprise the rule for that row:

-

In the **Level** and **Relationship** columns, the key fields on a given row map the corresponding property tag data to the Excel file as follows:

-

The **Level** key field specifies the level of the objects for which the property tag data is exported. For example, with **2** as the key field argument, data is exported for objects that appear at level two in the view.



The level number does not imply a hierarchy but instead reflects the relationship in the **Relationship** key field. Level precedence is according to numeric value. For example, level **2** is greater than level **1**.

-

The **Relationship** key field applies when search results are output directly to Excel or are exported to Excel from the Search Results dialog window. The field specifies the relationship to which the rule applies. For example, with **Complying Objects** as the key field argument, data is exported for objects that appear in the search results because they comply with other objects.

The **Level** and **Relationship** key fields are similar to **ADD** statements in the Advanced Search view. Therefore, you can run report scripts in the Search Results dialog window as a model to construct rules in your template. For more information about the Advanced Search view, see the *Systems Architect/Requirements Management User's Manual*.

-

In the **Subtype** column, the key field names the type definition for which the property tag data is exported. For example, with **Requirement** as the key field argument, data is exported for requirements that are selected for export to Excel, and also for requirements that are included in search results.

Levels and Relationships

Levels and relationships depend on how the user specifies the objects to export:

- When only selected objects are exported from a view, the level is **1** and the relationship is **Member** for each object.
- When all visible objects are exported from a view, the level is **1** for the leftmost objects, with level numbers increasing from left to right by numeric value.

The relationship depends on the view from which the objects are exported:

- From the content table, the relationship is **Member**.
- From the **Trace** subtab of the **Links** tab or window:
 - For the **Defining Trace** pane, the relationship is **Defining**.
 - For the **Complying Trace** pane, the relationship is **Complying**.
- From the **Complying Object Traceability** window, the relationship is **Complying**.
- From the **Defining Object Traceability** window, the relationship is **Defining**.
- From the Search Results dialog window, the relationship is the one used in the script to populate the dialog window. This is also the relationship for search results that are output directly to Excel.

Data Placement in Export Files

When users export objects to Excel, objects in the database are examined to determine if their levels match the key fields in the rule table of the selected template. For the object that is currently being processed, data is placed in the export file as follows:

- If the level of the current object is lower than the level of the previous object, data for the current object is placed in a new row.
- If the level of the current object is greater than or equal to the level of the previous object:
 - If the cells following the previous object are empty, data for the current object is placed in the same row.
 - If the cells following the previous object contain any character, data for the current object is placed in a new row.



- A cell that contains a space is considered to be empty. To keep a cell from being overwritten, you must enter a visible character in the cell.
- Siemens PLM Software recommends that you do not apply any formatting to a cell where you intend to begin one object on the same row after another object. Cell formatting may be processed as content, which in turn may prevent this data placement.

For example, assume the following:

- Objects **A**, **D**, and **E** are at level one.
- Objects **B**, **C**, **F**, and **G** are at level two.

Assume also that tags for properties named **1**, **2**, **3**, and **4** are entered in the property columns of the template, with key fields entered in the rule table as in the example below:

					Level	Relationship	Subtype
{%1}	{%2}	{%3}			{%L-1}		
			{%1}	{%4}	{%L-2}		

By default, the property tags and key fields in the example above produce the following output in the export file:

A1	A2	A3		
			B1	B4
			C1	C4
D1	D2	D3		
E1	E2	E3		
			F1	F4
			G1	G4

When the **Apply Packing** value is applied to the template's **Excel Template Rules** property:

- **B1**, **B4**, **F1**, and **F4** are placed in the row above with the parent objects.
- **C1**, **C4**, **G1**, and **G4** are placed in the row immediately below.

A1	A2	A3	B1	B4
			C1	C4
D1	D2	D3		
E1	E2	E3	F1	F4
			G1	G4

For more information, see [Packing Multiple Objects on One Row](#), earlier in this chapter.



Data is packed only if the user exports all objects in the view. If the user exports selected objects only, data for each object is placed on a separate row.

Criteria Matching

The more key fields that you enter, the more criteria that the objects must match. Thus, a rule containing three key fields specifies the most restrictive criteria, and data is extracted for fewer objects. With a rule containing only the **Subtype** key field, for example, data is extracted for all objects of the specified type definition.

Beginning with the topmost rule in the table, rules filter tag data as follows:

- If an object is found that matches all criteria in the current rule, the tag data is extracted to the spreadsheet and rule processing stops for that object. Then, rule processing for another object starts with the topmost rule.
 - If any portion of a key field is blank, that criteria is matched by default.
 - If the current rule is all blank, all criteria are matched by default. Therefore, Siemens PLM Software recommends that you place rules containing more key fields above rules with fewer criteria.
- If an object does not match any rule in the template, no tag data is extracted for that object. When the spreadsheet is generated, a warning message states that objects that do not match any template criteria are ignored.

Conditions

The rule table in a template is not exported to the spreadsheet, nor are the **<start>** and **<end>** tags exported. In addition, the following conditions apply:

- A **<rule>** field must precede the headings in the **Level**, **Relationship**, and **Subtype** columns. The sequence of the **Level**, **Relationship**, and **Subtype** columns must not be changed, and additional columns must not be inserted within the rule table.
- If any column in the rule table is missing, including the **<rule>** field column, an error message is displayed when the user attempts to generate a spreadsheet based on the template.
- The template must contain only one **<start>** tag and only one **<end>** tag. The **<start>** tag must appear to the left of the **<rule>** field in the rule table heading.
- All property columns must be located to the left of the rule table. Any cell content to the right of the rule table is not exported to the spreadsheet.
- All rows containing data tags must be located below the **<start>** tag and above the **<end>** tag. Cell content outside that area is exported as boilerplate.
- If a property column cell contains both a data tag and a constant value, the entire cell contents are exported as a constant value.



If properties in an Excel template do not exist in the database, the text string **Property does not exist** is placed in the property columns of the Excel export file. To change this text, select the template, double-click the **Error String** value in the **Properties** tab, and enter the new text in the open text field.

Date Style Support for Export to Microsoft Office Excel

Architect/Requirements uses locale-specific date styles to export data in Excel using views. If the date styles for client locale are not found, Architect/Requirements prompts the user to confirm before exporting the data in English (US) date format. Architect/Requirements OOTB supports English (US), English (UK), English (Australia), English (Canada), German, Dutch (Belgium), Dutch (Netherlands), Korean, Japanese, Chinese, and Russian date formats.

Users can extend the support for these styles by adding locale-specific styles in a custom folder and specifying this folder path in the **Custom.Folder.Path** Web Application Configuration parameter.

Users can use the **Date Style Generator** utility to generate these custom date property file (**.properties** files) and copy them to the custom folder specified in the **Custom.Folder.Path** parameter. For more information about the prerequisites and the usage of the utility, see appendix C, *Date Style Generator Utility*.

Stencils for Microsoft Office Visio



This section assumes that you are familiar with Microsoft Office Visio and with XML.

An Architect/Requirements stencil object is referenced when users do the following in the Systems Engineering and Requirements Management or Search module:

- Create diagrams in the Architect/Requirements live interface with Microsoft Office Visio. For more information about using live Visio diagrams, see the *Systems Architect/Requirements Management User's Manual*.
- Choose **Microsoft Visio** as the output option for search results. For more information about exporting a report to Microsoft Office Visio, see the *Systems Architect/Requirements Management User's Manual*.

The stencil object determines the Visio shapes that represent objects in the Architect/Requirements client and the properties of the shapes in the diagram. In the Administration module, two stencil objects are created automatically in the **Reports and Formatting** folder of every new project:

- The **Default Stencil**, for live Visio diagrams. Because live Visio diagrams are synchronized with the Architect/Requirements database, they can be used to dynamically add, modify, and delete objects in the client.
- The **Default Static Tree Stencil**, for exporting search results to static Visio diagrams that are not synchronized with the database. In such a diagram, results are displayed in a multiple level tree according to qualifying relationships specified in the search criteria, for example, by using subordinate **ADD** statements with **FOR EACH** statements in the Advanced Search view. For more information about **FOR EACH** and **ADD** statements, see the *Systems Architect/Requirements Management User's Manual*.
- The **Port Type Stencil**, for creating port type live Visio diagrams. Because live Visio diagrams are synchronized with the Architect/Requirements database, they can be used to dynamically add, modify, and delete objects such as ports, connections, trace links, in the client.

The project administrator can modify the default stencil objects and can create and modify custom stencil objects. Every stencil object has two components:

- A Visio stencil file (**.vss**), containing the shapes that represent Architect/Requirements objects. Through the standard Visio functions, shapes can be edited in existing stencil files, and new stencil files with custom shapes can be created. For more information, see the Microsoft Office Visio documentation.



Microsoft Office Visio 2013 or Visio 2016 must be installed on your computer.

- A mapping file (**.xml**), containing XML tags that map shapes in the stencil file to objects and properties in the Architect/Requirements database. Through an XML editor, or a text editor such as Microsoft Notepad or WordPad, tags can be edited in existing mapping files, and new mapping files with custom tags can be created. For more information, see [Configuring a Mapping File](#), later in this chapter.

The default stencil file is **TeamcenterRequirements.vss** and the default mapping file is **tcr_visio_prop_map.xml**. The project administrator can modify each of these files. In addition, the project administrator can create stencil files and mapping files, and can apply those components to the default stencil objects and to custom stencil objects. For more information, see [Creating a Stencil Object](#) and [Modifying a Stencil Object](#), later in this chapter.

New Visio 2013 Stencil and Diagram Formats Not Supported

Architect/Requirements 9.1.4 and later supports integration with Visio 2013. Visio 2013 introduces new formats for diagrams (**VSDX**) and stencils (**VSSX**), however, they are not supported by Architect/Requirements. This affects the Architect/Requirements Project Administrator in performing the following function:

When creating a Stencil Object in the Architect/Requirements database, the initial step is to create a Visio stencil file (**.vss**), which contains shapes that represent objects in Architect/Requirements. You create the stencil file in Microsoft Office Visio and add the shapes through the standard Visio functions. This same procedure is followed with Visio 2013. However, you must note the following:

- When using Visio 2013 to create a stencil file for use in creating a Stencil Object, save the file in **VSS** format and use ***.vss** extension.
- After clicking the folder button on the right side of the **Stencil File** field to display the **Open** dialog window, select a stencil that was created in **VSS** format.

Architect/Requirements does not function correctly if you select stencil files created in **VSSX** format.

Configuring a Mapping File

A mapping file allows you to assign shapes in a stencil file to objects and properties in the database. To configure the file, you enter and change certain XML tags according to syntax described in this section.

Using an XML editor, or a text editor such as Microsoft WordPad or Notepad, you can create a new mapping file and enter new tags. Or, you can modify the tags in an existing mapping file, including the default mapping file, **tcr_visio_prop_map.xml**.



If you add a new shape to the stencil file, Siemens PLM Software recommends that you copy the **shapeMap** tag for a standard shape in the mapping file and modify that tag to suit your requirements.

In a mapping file, you can do any or all of the following:

- Map a type definition to live Visio master shapes.
- Map text properties on a shape.
- Map an Architect/Requirements type or subtype to a Visio shape based on properties.
- Map stylistic properties on a shape.
- Map a master shape in the stencil to a connection type definition in the database.
- Map a Visio master shape in the stencil to a trace link object type in the database.
- Map line style properties for connectors defined by Visio to a relationship shape in a live Visio static tree stencil.

Mapping a Type Definition to Live Visio Master Shapes

Through the `tcr_visio_prop_map.xml` file, you can map a type definition in Architect/Requirements to one or more master shapes in a live Visio stencil.

The `visioMaster` attribute for the `shapeMap` indicates the master shape that you are attempting to map.

```
<shapeMap visioMaster="Requirement">
  <typeMaps>
    <typeMap tcrType="Requirement" tcrSubType="" default="true" />
    <typeMap tcrType="Requirement" tcrSubType="Paragraph" />
  </typeMaps>
</shapeMap>
```

`typeMap` indicates the object type and subtype being mapped to this particular master. You can associate multiple type and subtype combinations to one master.

However, `default="true"` can be only on one of the entries under the `<type maps>` tag. Default indicates that when the shape is dragged from the stencil to the page, an object of that type and subtype is created in the database.



- You must have at least one `shapeMap` tag per shape in the `TeamcenterRequirements.vss` file.
- You can map a single type or subtype to multiple master shapes. Then, users can choose various master shapes when creating objects of that type or subtype in a live Visio diagram.
 - o Multiple master shapes are not available for member objects that are added to a live Visio diagram owner in Architect/Requirements while the diagram is open. The first master shape mapping for a given type or subtype in the XML file is used for corresponding member shapes.
 - o When a live Visio diagram is created for a diagram owner in Architect/Requirements, the first master shape mapping for a given type or subtype in the XML file is used for corresponding member shapes.
 - o When a property is updated in Architect/Requirements, the property is updated on the corresponding shapes in live Visio diagrams, regardless of their mapping sequences in the XML file.

Mapping Text Properties on a Shape

Map text properties on a shape by defining a tag such as the following:

```
<textPropertyMaps>
  <textPropertyMap tcrProp="ROIN" visioProp="1.Text" />
  <textPropertyMap tcrProp="Name" visioProp="Text" />
</textPropertyMaps>
```

For this shape, the **ROIN** property is mapped to text property of the first subshape, by the prefix **1.** preceding **Text**. To map to the **Text** property of the second subshape, for example, enter **2.Text**. If there is no prefix, the property is mapped to the **Text** property of the master shape and not to any of the subshapes.

Property-based Mapping of an Architect/Requirements Type or Subtype to a Visio Master Shape

Shapes in a Visio diagram can be created based on the property value of an Architect/Requirements object. This feature eliminates the need to maintain an extensive library of Architect/Requirements types or subtypes for each Visio shape.

When you create a shape in Visio, Architect/Requirements creates an object with the correct type or subtype and set the property value specified in the mapping defined in the property mapping file for the object's type or subtype and its property value. For example, you can create a single Architect/Requirements subtype called **Aircraft Wing**. This subtype can have a single choice property called **WingType** with **Delta**, **Straight**, and **Swept** as the possible values. When you create an **Aircraft Wing** object with **WingType** value as **Delta** and send it to Visio, a Visio shape corresponding to the **Delta** value is picked up. Similarly, when you drag a shape named **Straight** from Visio stencil to a diagram, Architect/Requirements creates an **Aircraft Wing** object and sets the **WingType** value to **Straight**.



- Multi-choice property value is not supported.
- Mapping date property value is not supported.
- The change in shape due to a change in the mapping file is not live. For example, when you change the property value of a Architect/Requirements object for which a different shape is mapped in the property mapping file, the shape will not change in Visio diagram.
- When you select a start object for any link in Architect/Requirements and its end object in Visio, the property mapping is not applied for the link that is created. You should create a link either in Architect/Requirements or in Visio completely.

You can map a property value of a stencil object to a Visio shape by defining a **tcrAttributeMap** tag in the **shapeMap** tag, as shown below:

```
<shapeMap visioMaster="Requirement">
  <typeMaps>
    <typeMap tcrType="Requirement" tcrSubType="" default="true" />
      <tcrAttributeMap>
        <tcrAttribute name="color" value="blue" />
      </tcrAttributeMap>
    </typeMaps>
    ...
  </shapeMap>
```

For this shape, the **color** property is mapped to the property value **blue**. All requirement objects or its subtypes created will have **blue** set as the property value for the **color** property.

A property of an attribute and the property's value should be mapped to an appropriate Architect/Requirements type or subtype in order to view the shape in a Visio diagram. The following are a few scenarios and the corresponding behavior of the property-based mapping.

- If an attribute and its value (any numeric value with or without decimal) are specified in the mapping file, and then you create a numeric property in Architect/Requirements, the value for this numeric property should be the same as specified in the mapping file. For example, if the value of an attribute is **8.0** in the mapping file, the value in Architect/Requirements should also be **8.0** (not 8) for an object in order to view the shape in a Visio diagram.

- If a property and its value are mapped to a Visio shape and that property exists in Architect/Requirements but is not associated to an appropriate type or subtype, then the drag-drop functionality of that type or subtype in Visio is not supported. Also, if you try to create a diagram for a folder that contains objects with unassociated types or subtypes, the objects are not sent to Visio.
- If a property and its value are mapped to a Visio shape and that property exists in Architect/Requirements but is associated to an appropriate type or subtype with a false value, then the objects of such types or subtypes are not sent to Visio when you create a diagram for the parent folder. Also, when the objects are created in Visio, they remain in the Visio diagram despite the false values (values specified other than the values in the mapping file).
- Applicable to building blocks only:
If a child of a building block has incorrect value for a specified attribute and the parent has a correct value, Visio displays only the parent building block object. When the parent object is opened with the child diagram, the child object is not displayed. Similarly, if a child building block object has a child with a correct value for the specified attribute, then opening the top level building block with the child diagram does not show its immediate child, and therefore, the child of its child.

Mapping Stylistic Properties on a Shape

Map stylistic properties on a shape by defining a tag such as the following:

```
<stylePropertyMap tcrProp="Number" visioProp="1.FillForegnd"
visioDefaultValue="5">
  <valueMap tcrValue="1" visioValue="3" />
  <valueMap tcrValue="2" visioValue="4" />
  valueMap tcrValue="3" visioValue="2" /
</stylePropertyMap>
```

Similar to a text property map, this maps properties like color or border on a shape. In the example above, the **Number** property is mapped to **FillForegnd** (color) of first subshape.

In addition to property mapping, you must provide value mapping as well. For example, **tcrValue="1"** is mapped to **visioValue="3"** (green), and **tcrValue="2"** is mapped to **visioValue="4"** (blue). If the Architect/Requirements value does not match any in the **valueMap**, **visioDefaultValue** is used.

Mapping a Master Shape to a Connection Type Definition

Map a Visio master shape in the stencil to a connection object type in the database.

The **Connection** attribute denotes that this master shape is of the **ConnectionDB** type.

```
<shapeMap visioMaster="Connection" Connection="true">
  <typeMaps>
    <typeMap tcrType="Connection" tcrSubType="" default="true" />
  </typeMaps>
  <propertyMaps>
    <textPropertyMaps>
      <textPropertyMap tcrProp="Name" visioProp="Text" />
    </textPropertyMaps>
  </propertyMaps>
</shapeMap>
```

A connection can be configured to create only a particular object type at each end of the connection. In the following example, the connection is configured to create only objects of the **IDEF0 Input Port** type at the terminating end and only objects of the **IDEF0 Output Port** type at the originating end.

```
<PortSubtypeMaps>
  <PortSubtypeMap connectionToEnd = "IDEF0 Input Port"
    connectionFromEnd = "IDEF0 Output Port"/>
</PortSubtypeMaps>
```

Mapping a Master Shape to a Port Type Definition

Map a Visio master shape in the stencil to a port type definition in the database.

```
<shapeMap visioMaster="Port">
  <typemaps>
    <typeMap tcrType="Port" direction = "External" tcrSubType=""
      default="true" />
  </typemaps>
```

The "External" value allows the port to be attached to a building block or a building block subtype.

Mapping a Master Shape to a Port Subtype

Map a Visio master shape in the stencil to a port subtype in the database.



- The stencil object's **Uses Ports** property value must be set to **Yes**.
- Each external port subtype master shape must have a corresponding internal port subtype master shape.

```
<shapeMap visioMaster="IDEF0 Input Port">
  <typeMaps>
    <typeMap tcrType="Port" direction = "External"
      tcrSubType="IDEF0 Input Port" default="true" />
  </typeMaps>
```

The subtype must exist in the **Type Definitions** folder. Type the exact name of the subtype as the **tcrSubType** attribute value. For more information, see [Creating a Subtype](#) in chapter 3, *Customizing Object Types*.

You can use the following attributes for additional control of the port location and the direction of the connection that the port can accept:

Attribute	Value	
ConnectBBposition	Left	Port attaches to the left side of the building block.
	Right	Port attaches to the right side of the building block.
	Top	Port attaches to the top of the building block.
	Bottom	Port attaches to the bottom of the building block.
portDir	In	Port accepts incoming connection.
	Out	Port accepts outgoing connection.

For example:

```
<shapeMap visioMaster="IDEF0 Input Port">
  <typeMaps>
    <typeMap tcrType="Port" direction = "External"
      tcrSubType="IDEF0 Input Port" ConnectBBposition="Left"
      portDir="In" default="true" />
  </typeMaps>
```

Mapping a Master Shape to a Trace Link Type Definition

Map a Visio master shape in the stencil to the trace link object type in the database.

The **Trace Link** attribute denotes that this master shape is of the **Tracelink** type.

```
<shapeMap visioMaster="Trace Link" Tracelink="true">
  <typeMaps>
    <typeMap tcrType="Trace Link" tcrSubType="" default="true" />
  </typeMaps>
  <propertyMaps>
    <textPropertyMaps>
      <textPropertyMap tcrProp="Name" visioProp="Text" />
    </textPropertyMaps>
  </propertyMaps>
</shapeMap>
```

Mapping Line Style Properties for Connections Defined by Visio to a Relationship Shape in a Live Visio Static Tree Stencil

Map line style properties for connectors defined by Visio to a relationship shape in a live Visio static tree stencil.

The `tcr_visio_prop_map.xml` file contains one section for each relationship on which the user can run a search in the Architect/Requirements client. The following example represents one of the sections in the property mapping XML file for a live Visio static tree diagram stencil:

```
<relationshipMap type="Defining Objects">
  <Line Pattern="1" Weight="0.12" Cap="0" Transparency="0">
    <Color Numeric="2" Red="" Green="" Blue="" />
  </Line>
  <LineEnds Begin="0" End="0" BeginSize="Medium" EndSize="Medium" />
  <RoundCorners Rounding="0" />
</relationshipMap>
```

To set the values in the XML section, adjust the desired values in the Line dialog window. Open the Line dialog window for the connector that is used for showing the relationships in a static tree diagram. The **Line** subsection of the XML file represents the **Line** section of the dialog window. Similarly, the **LineEnds** and **RoundCorners** sections in the XML file represent the corresponding sections of the dialog window.

The **Line** section contains a **Color** subsection that corresponds to the **Color** field in the dialog window. Each can also have custom colors with different combinations of red, green, and blue (RGB), as well as Visio defined colors represented by numeric codes. Thus, the **Color** subsection has attributes for setting numeric values for color as well as for setting custom RGB color on a relationship shape.

To get the Visio values that represent a particular style, select the relationship shape, and then pull down the **Window** menu and choose the **Shape Sheet**→**Line Format** options to display the **Line Format** section. Then, copy the numeric values from the **Line Format** section and add the values to the XML file for the corresponding relationship section.



For **LineColor**, copy only the values in the parentheses of the formula. Do not copy the entire formula, for example, **HSL** (*n*, *n*, *n*).

Creating a Stencil Object

Every stencil object consists of two component files. For a new stencil object, you create the following files on your computer:

- A stencil file (.vss), which contains shapes that represent objects in Architect/Requirements. You create this file in Microsoft Office Visio and add the shapes through the standard Visio functions. For more information, see the Microsoft Office Visio documentation.



Microsoft Office Visio 2013 or Visio 2016 must be installed on your computer.

- A mapping file (.xml), which contains XML tags that map shapes in the stencil file to objects and properties in the Architect/Requirements database. You use an XML editor, such as Microsoft WordPad or Notepad, to create this file and add the tags. For more information, see [Configuring a Mapping File](#), earlier in this chapter.

After you create the custom components, you create the new stencil object by loading the components from your computer to Architect/Requirements.

To create a stencil object:

1. On your computer, create the stencil file and the mapping file that you want to associate with the new stencil object.

2.

Open the **Reports and Formatting** folder, and then pull down the **File** menu and choose **New**→**Template**→**Stencil**.

The Select dialog window is displayed.

3.

Associate your custom components with the stencil object by doing the following:

- a. Click the folder button to the right of the **Stencil File** field to display the Open dialog window.
 - b. Select the stencil file, and then click **Open** to fill in the field automatically.
 - c. Click the folder button to the right of the **Mapping XML File** field to display the Open dialog window.
 - d. Select the mapping file, and then click **Open** to fill in the field automatically.
4. To load the components and create the stencil object, click **OK**.

The content table displays the new stencil object with a default name in an open text field.

5. Enter the stencil object name, and then press the enter key.

To refresh the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the **Reports and Formatting** folder and choose **Refresh** from the popup menu. You can also click the **Refresh** button on the toolbar.

Procedure Notes

Step 3: In the Open dialog window, you can use the **Look in** field to change the drive or folder.

Modifying a Stencil Object

By modifying stencil objects, you can customize the shapes in live Visio diagrams and the shapes in search results that users output to Microsoft Office Visio. Two default stencil objects are generated automatically for every new project, in the **Reports and Formatting** folder. The folder also contains stencil objects that you create. For more information, see [Creating a Stencil Object](#), earlier in this chapter.

Every stencil object consists of two component files:

- A stencil file (**.vss**), where you can add and change shapes through the standard Visio functions. The default stencil file is **Default Stencil.vss**.



Microsoft Office Visio 2013 or Visio 2016 must be installed on your computer.

- A mapping file (**.xml**), where you can add and change tags that define the properties of shapes in the stencil file. The default mapping file is **tcr_visio_prop_map.xml**. You edit a mapping file with a text editor, such as Microsoft Notepad or WordPad, or an XML editor. For more information, see [Configuring a Mapping File](#), earlier in this chapter.

You can modify any stencil object, including the **Default Stencil** (for live Visio diagrams) and the **Default Static Tree Stencil** (for search results). For the selected stencil object, you copy the stencil file and the mapping file from the Architect/Requirements client to your computer. There, you edit one or both of the files. When you complete those changes, you reload each changed file from your computer to the stencil object in the client.

To modify a stencil object:

1. Copy the stencil file and the mapping file to your computer by doing the following:
 - a. In the **Reports and Formatting** folder, right-click the stencil object and choose **Copy to client** from the popup menu.

The Select Location to Copy dialog window displays the folders and files in the current drive or folder. You can use the **Look in** field to change the drive or folder.

- b. Select the drive or folder, and then click **Open**.
A message confirms the location of the copied files.
2. On your computer, edit one or both of the copied files.
 3. Pull down the **File** menu and choose **Import**→**Stencil** to display the Select dialog window.
 4. Do one or both of the following:

a.

If you edited the stencil file:

- Click the folder button to the right of the **Stencil File** field.
The Open dialog window displays the folders and files in the current drive or folder. You can use the **Look in** field to change the drive or folder.
- Select the edited stencil file, and then click **Open** to fill in the field automatically.

b.

If you edited the mapping file:

- Click the folder button to the right of the **Mapping XML File** field.
The Open dialog window displays the folders and files in the current drive or folder. You can use the **Look in** field to change the drive or folder.
- Select the edited mapping file, and then click **Open** to fill in the field automatically.



If you edited only one of these files, clear the checkbox for the unchanged file.

5. To reload the edited files into the stencil object, click **OK**.

The Select dialog window closes, and the modifications are applied to the stencil object.



If the stencil object's **Uses Port** property value was **Yes** before the modifications, ensure that you reset this value to **Yes** after reloading edited files into the stencil object.

Procedure Notes

Step 3: You can also right-click the stencil object and choose **Import**→**Stencil** from the popup menu. Or, double-click the stencil object.

Step 4: You can also enter the path, the file name, and the appropriate file name extension directly in each field.

Viewing Diagram Links

Architect/Requirements allows you to view the diagram links in a project. However, you must enable the display of diagram links before you can view them.

Enabling diagram links display

Diagram links are not visible by default. You must configure Architect/Requirements to view the diagram links. The configuration to view the diagram links is done on the project level.

To make the diagram links visible:

1. Select the project for which you want to view the diagram links in the **Navigation** tree.
2. Click **Properties** tab in the **Notebook** pane.
3. Scroll down and select **Project Settings**.
4. In the Project Settings window, select **Show Diagram Links** check box.
5. Click **OK**.

Viewing diagram links

You can view the diagram links in **Relations** tab of link view.

To view the diagram links:

1. Select the diagram in the **Content** table.
2. Click **Links** in the **Notebook** pane to make the **Links** tab active.
3. Open the tab in a floating window.
Click the **Open tab in a new window** button to open the tab in a floating window.
4. Click the **Relations** tab in the **Links** window.
5. Select the diagram in the **Attachment** tab.
The diagram links are displayed in the **Links** window.

Deleting diagram links

You can delete the diagram links if the following conditions are true:

- If Diagram Content property of a diagram is set to static.
- The owner of a diagram is in non frozen state.

To set the Diagram Content property to static:

- Select the diagram in the **Content** table.
- Right click the diagram in the **Attachment** tab and select **Properties**.
- Double click the values box for **Diagram Content**.
- In the **Diagram Content** dialog, select **Static** and click **OK**.
- Click **Close**.

To delete a diagram link:

- Set the **Diagram Content** property to static and open the diagram link in the floating **Links** window.

For information on opening the diagram link in a floating window, see [Viewing diagram links](#).

- Right click the diagram link that you want to delete.
- To confirm the deletion, click **Yes**.

Searching for diagram links

You can search for requirements that contain diagram links or all the diagram links in a project. You need to use the **Search** module to perform the search.

To search for diagram links:

1. Select the project in which you want to search for the diagram links and click the **Search** button in the **Module** bar.
2. Select **Requirement** in the **Types/Subtypes** list.
3. Select **Advanced** for the **Search Type**.
4. In the **Modify** group, click the **Add Indented** button.
5. In the **Query Edit** area, select **FOR EACH** for the **Command**.
6. In the **Modify** group, click the **Add Indented** button.
7. In the **Query Edit** area, select **Diagram Links** from the **Relationship** drop down list.

The query in the **Query View** is `SELECT Requirement FOR EACH ADD Diagram Links`.

8. Click the **Search** button.

Architect/Requirements displays the search results in a new window.

To search for all the diagram links in a diagram:

1. Select the project in which you want to search for the diagram links and click the **Search** button in the **Module** bar.
2. Select **Diagram** in the **Types/Subtypes** list.
3. Select **Advanced** for the **Search Type**.
4. In the **Modify** group, click the **Add Indented** button.
5. In the **Query Edit** area, select **FOR EACH** for the **Command**.
6. In the **Modify** group, click the **Add Indented** button.
7. In the **Query Edit** area, select **Diagram Member Links** from the **Relationship** drop down list.

The query in the **Query View** is `SELECT Diagram FOR EACH ADD Diagram Member Links`.

8. Click the **Search** button.

Architect/Requirements displays all the all the diagram links for the diagrams in a new window.

Guidelines for Working In Live Visio Stencils

This section describes some general guidelines for working in live Visio stencils.

Considerations for Creating Port Master Shapes

In creating port master shapes, some special considerations are involved:

- Create a port master shape as a group shape with two subshapes.



The live Visio **Hide/Unhide** feature requires two sub-shapes. Use one subshape for displaying the port and one subshape for displaying the port name. You can add text on the port name subshape.

-

Open the shape sheet for each subshape and add a layer membership section.

- o Add a **Port** layer for the port shape subshape.
- o Add a **PortName** layer for the port name subshape.

Connection Point Sections for Port Master Shapes

After you create a port master shape and before you save it, ensure that it has a connection point section:

- Select the port master shape, and then pull down the Visio **Window** menu and choose **Show ShapeSheet**.

The connection point section is displayed if it exists.

To create a connection point section:

- . Select the shape, and then pull down the Visio **Insert** menu and choose **Section**.

The Insert Section dialog window is displayed.

- . Check the **Connection points** check box and click **OK**.

The connection point section is added to the shape sheet.



The connection point section initially contains one row for one connection point. You can assign the X and Y coordinates for that connection point.

You can also insert rows for additional connection points. To rename a connection point, select the default row name and enter the new name.

Then do the following in the connection point section:

1. Delete all connection point rows from the connection point section.
2. Insert a new row in the connection point section.
3. Assign the X and Y coordinates so that the connection point is created at the center of the port shape.
4. Select the **Type/C** cell of the new row, and then select **2 – visCnnctTypeInwardOutward** from the list.
5. Ensure that only one connection point exists on the port master shape.



Live Visio makes additional connection points available when a corresponding port shape is used on the Visio page.

6. Close the shape sheet and save the master.
7. In a new or existing mapping file, assign the port master shape to objects and properties in the Architect/Requirements database.

For more information, see [Configuring a Mapping File](#), earlier in this chapter.

For information about working with connection points, see the Microsoft documentation at the following URL:

<http://office.microsoft.com/en-us/visio/HP010473091033.aspx>

Using connection points

Connections are attached to Architect/Requirements building blocks and ports. You must specify connection points when master shapes for building blocks and ports are defined. The connection points indicate points where connections are attached. There are restrictions on where you can define the connection points for Architect/Requirements master shapes. In Microsoft Visio, master shapes are defined by grouping together multiple sub-shapes. Architect/Requirements interacts correctly only with connection points defined at the group level. If the shape sheet for a sub-shape has a **Connection Points** section, you must remove it. You can add the required connection points at the group level.

Chapter 5: Managing Users

This chapter contains an overview of users and instructions for creating new users and user groups and for managing project access, user profiles, and user passwords.

Overview of Users

Individuals are represented in Architect/Requirements by *user* objects, whose registration determines project access and licensing. When a new user is created, the user object is registered in the database and is added to the **Users** folder of the **TcSE Administration** project, through which all user related transactions are recorded.

In the **TcSE Administration** project, the **Users** folder contains all activated and deactivated user objects for all projects in the Architect/Requirements installation. In every other project, the **Users** folder contains a user object for each user who currently has access to that project. A user can be managed from any project in which the **Users** folder contains that user object. Users can be managed also through *user groups*, which consist of shortcuts to existing user objects in a project.

Project Access

For each user, a *maximum privilege level* limits the *access privilege* that can be granted in any project. In turn, the access privilege controls the user's actions in a specific project. The maximum privilege level assigned to the user determines project access as follows, from the highest maximum privilege level to the lowest:

- **Enterprise Administrator** allows the user to create new projects, to delete any existing project, and to perform any action in all projects. The user does not need project specific access privileges because enterprise administrators automatically have full access to every project.

When an Architect/Requirements database is created through the normal installation process, the creator becomes the original enterprise administrator by default. A user object is generated for the database creator, with a user name of **tradm** and a blank password. The maximum privilege level for that user object cannot be changed by any user, including the database creator.

- **Project Administrator** allows the user to perform any action in and to delete specific projects, those where **Project Administrator** access privilege is granted. However, the user cannot create new projects.

-

Power User is a non-administrative **Read and Write** user who is given certain project-specific privileges to assist with project administration. The project-specific privileges are managed through special user groups and security profiles.

-

Read and Write allows the user to view, modify, create, and delete objects in the Systems Engineering module, and to view objects in the Administration module, in projects where **Read and Write** access privilege is granted.

-

Read Only allows the user only to view objects in the Systems Engineering module, in projects where **Read Only** access privilege is granted. The user cannot view schema objects in the Administration module.

-

No Access revokes access to all projects, thereby deactivating the user. The user object remains in the **Users** folder of the **TcSE Administration** project.

Except for **No Access**, a given maximum privilege level can be assigned to any number of new and existing users. **No Access** can be assigned only to existing users.

Whereas a user's maximum privilege level applies to all projects, access privileges can differ from one project to another. For example, a user may have **Project Administrator** access privilege in one project, **Read and Write** access privilege in a second project, and **Read Only** access privilege in a third project. The access privilege in any project must be equal to or lower than the current maximum privilege level. For a higher access privilege, a higher maximum privilege level must first be assigned. For more information, see [Modifying a User Profile](#), later in this chapter.



Notwithstanding the access privilege, a user's actions in a project may be further controlled by a security profile. For more information, see [Security Profiles and Access Control](#) in chapter *****Unsatisfied xref reference*****, *****Unsatisfied xref title*****.

Special Privileges

In addition to a user's access privilege, special privileges can be granted in specific projects:

-

Script Authoring privilege allows a user to create, edit, and run activators in the Administration module. This privilege is granted for a project by adding **Script Authoring** to the **Additional Privilege** property of the user object. For more information about activators, see the *Systems Architect/Requirements Management API Reference* manual.

-

Architect privilege allows a user to work directly with building blocks, TRAMs, and diagrams in the Systems Engineering module. The user can do the following:

- o Create building blocks and TRAMs, using system-defined and user-defined subtypes.
- o Promote building blocks and TRAMs to higher levels in a hierarchy.
- o Demote building blocks and TRAMs to lower levels in a hierarchy.
- o Move building blocks and TRAMs up or down within a level in a hierarchy.

- o Create and edit diagrams with the Architect/Requirements live interface to Microsoft Office Visio.
- o Embed diagrams in Microsoft Word documents.
- o Copy building blocks, TRAMs, and diagrams to create new objects.
- o Move building blocks, TRAMs, and diagrams to different owners.
- o Rename building blocks and TRAMs.
- o Create shortcuts to building blocks, TRAMs, and diagrams.
- o Import building blocks and TRAMs from Microsoft Excel and XML files to new and existing projects.
- o Delete building blocks, TRAMs, and diagrams.
- o Change the values of editable properties for building blocks and TRAMs.
- o Calculate numeric properties for all object types.
- o Freeze and unfreeze building blocks and TRAMs.
- o Create versions, variants, and baselines of building blocks and TRAMs.
- o Submit changed building blocks and TRAMs for approval.

This privilege is granted for a project by adding **Architect** to the **Additional Privilege** property of the user object. Users who do not have this privilege can create notes and trace links for building blocks, TRAMs, and diagrams, and also can view those objects.

Power Users

Power users are non-administrative users who are assigned with project-specific privileges (between **Read and Write** and **Project Administrator** access privileges) to assist with the project administration. Power users perform relatively routine tasks based on the project-specific privileges assigned to them. Project administrators can add the **Power Users** package to a project and assign privileges by adding users to one of the power user groups.

To enable the **Power Users** package for a project:

1. From the **Administration** module, select the **Projects** node.
The content table displays all projects to which you have access.
2. Select the project in the content table.
The **Properties** tab or window displays all viewable properties for the project.
3.
In the **Value** column, double-click the **Packages** property value.
The Multi-Choice dialog window is displayed.
4. Check the **Power Users** checkbox, and then click **OK** to close the dialog window.



This action cannot be reversed. The **Power Users** package cannot be removed from the project.

The **Power Users** value is added to the **Packages** property, indicating that the power user groups are enabled for the project. The power user groups can be accessed from the **Users→Power Users** folder of the project.

Power user access to administration objects is controlled by security profiles. Without a security profile, a power user has the same rights to an administration object as a read/write user. Installing the Power User package creates the security profiles and user groups for an access system based on the administration folders. The new security profiles are placed in a sub folder called **Power_User_Security_Profiles**. The user groups are placed in a sub folder called Power Users. The security profile can be easily modified or replaced when required as it uses the normal security profile mechanism. Inheritable security profiles are set on the administration folders so that new administration objects have the appropriate security profile. Out of the box objects and other administration objects created before the Power User package is enabled, do not inherit the security profile. Security profiles must be set manually on such objects if power user access is required. If power user access to certain objects is not required, the security profile can be removed. If power user access is not required for a particular administration folder, the folder's security profile can be removed or the security profile can be changed to ensure that the security profile is not applied to new descendants.

Users can be granted project-specific power user privileges by adding them to one the power user groups described below. These privileges are controlled through security profiles. To add users to a power user group, copy the selected users to the desired power user group. For more information, see [Modifying a User Group](#), later in this chapter.

- **PU_Activators_Access** privilege allows a user to promote menu folders and items. The user can create and modify all types of activators, which also requires script authoring privilege.
- **PU_Property_Definitions_Access** privilege allows a user to create definitions for choice, numeric, text, and date properties.
- **PU_Reports_and_Formatting_Access** privilege allows a user to create documents, excel files, object templates, stencils, and style sheets. The user can also promote reports and views.
- **PU_Security_Profiles_Access** privilege allows a user to create security profiles.
- **PU_Type_Definitions_Access** privilege allows a user to create subtypes of type definitions. Before granting this privilege, project administrators must add a security profile to a type definition that grants complete access to the power users.
- **PU_User_Access** privilege allows a user to create a new user group and assign users to that group. It also allows a user to create new users in the project, and manage user properties such as name, e-mail ID, and password.



For Excel imports, a power user must be added to the following user groups:

- **PU_Type_Definitions_Access**
- **PU_Property_Definitions_Access**

Licensing

Licenses for an Architect/Requirements installation are consumed only by activated users, those who currently have access to at least one project. For such a user, licensing is determined as follows:

- A **Requirements** license is consumed when the maximum privilege level is **Enterprise Administrator, Project Administrator, Power User, or Read and Write**. If the user has access to two or more projects, only one license is consumed.

For a user with this license, additional licenses can be assigned for special privileges:

-

A **Script Authoring** license is consumed, in conjunction with the **Requirements** license, when **Script Authoring** privilege is granted to the user in at least one project. If the user has this privilege in two or more projects, only one **Script Authoring** license is consumed. For more information about activators, see the *Systems Architect/Requirements Management API Reference* manual.

-

An **Architect** license is consumed, in conjunction with the **Requirements** license, when **Architect** privilege is granted to the user in at least one project. If the user has this privilege in two or more projects, only one **Architect** license is consumed.

Additional licenses are assigned through the **Additional Privilege** property of the user object. For more information, see [Special Privileges](#), earlier in this chapter.

- A **Consumer** license is consumed when the maximum privilege level is **Read Only**. If a **Consumer** license is not available, a **Requirements** license is consumed. If the user has access to two or more projects, only one license is consumed.

A user does not consume a license of any type when the maximum privilege level is **No Access**. The user is deactivated, although the user object remains in the **Users** folder of the **TcSE Administration** project. Users who currently have project access can be deactivated to free those licenses, which can then be assigned to other users or held in reserve. For more information, see [Deactivating a User](#), later in this chapter.

Without deactivating a user, the maximum privilege level can be changed to free the current **Requirements** or **Consumer** license and assign the other type, if available. Also, a user's **Script Authoring** license or **Architect** license can be freed by removing the corresponding value from the **Additional Privilege** property in each project where the user has that privilege. For more information, see [Modifying a User Profile](#), later in this chapter.

Table 1 identifies the menu options available in the Systems Engineering and Requirements Management module for each license type.

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
File	New→Requirement	<input type="checkbox"/>	<input type="checkbox"/>		control-R
File	New→Paragraph	<input type="checkbox"/>	<input type="checkbox"/>		control-H
File	New→Building Block		<input type="checkbox"/>		control-B
File	New→Folder	<input type="checkbox"/>	<input type="checkbox"/>		control-L
File	New→Note	<input type="checkbox"/>	<input type="checkbox"/>		control-N
File	New→Group	<input type="checkbox"/>	<input type="checkbox"/>		control-G
File	New→TRAM		<input type="checkbox"/>		
File	New→Document	<input type="checkbox"/>			
File	New→Spreadsheet	<input type="checkbox"/>	<input type="checkbox"/>		
File	New→Child	<input type="checkbox"/> ¹	<input type="checkbox"/>		control-K
File	New→Subtype	<input type="checkbox"/> ²	<input type="checkbox"/>		control-U
File	Open	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ³	control-O
File	Open Read-only	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
File	Excel Open	<input type="checkbox"/>	<input type="checkbox"/>		
File	Visio→Create Diagram		<input type="checkbox"/>		
File	Visio→Open	<input type="checkbox"/> ⁴	<input type="checkbox"/>	<input type="checkbox"/> ⁵	
File	Delete	<input type="checkbox"/> ⁶	<input type="checkbox"/>		delete
File	Rename	<input type="checkbox"/> ⁷	<input type="checkbox"/>		F2

¹ Disabled if a building block is selected.

² Disabled for building block and TRAM subtypes.

³ Opens object in read-only mode.

⁴ Opens object in read-only mode.

⁵ Opens object in read-only mode.

⁶ Disabled if a building block, TRAM, or diagram is selected. Disabled also if a building block, TRAM, or diagram is subordinate to a selected object.

⁷ Disabled if a building block or TRAM is selected.

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
File	Properties	<input type="checkbox"/> ⁸	<input type="checkbox"/>		
File	Export→Word Document	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ⁹	
File	Export→Excel Spreadsheet	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> ¹⁰	
File	Export→AP233	<input type="checkbox"/>	<input type="checkbox"/>		
File	Export→XML	<input type="checkbox"/>	<input type="checkbox"/>		
File	Import→Word Document	<input type="checkbox"/>	<input type="checkbox"/>		
File	Import→Excel Spreadsheet	<input type="checkbox"/> ¹¹	<input type="checkbox"/>		
File	Import→AP233	<input type="checkbox"/>	<input type="checkbox"/>		
File	Import→XML	<input type="checkbox"/> ¹²	<input type="checkbox"/>		
File	Exit	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	control-Q
Edit	Undo	<input type="checkbox"/>	<input type="checkbox"/>		control-Z
Edit	Cut	<input type="checkbox"/> ¹³	<input type="checkbox"/>		control-X
Edit	Copy	<input type="checkbox"/> ¹⁴	<input type="checkbox"/>		control-C
Edit	Paste	<input type="checkbox"/>	<input type="checkbox"/>		control-V
Edit	Paste Shortcut	<input type="checkbox"/> ¹⁵	<input type="checkbox"/>		
Edit	Copy URL	<input type="checkbox"/>	<input type="checkbox"/>		

⁸ Edit Properties dialog window is read-only if a building block or TRAM is selected.

⁹ Document in read-only mode.

¹⁰ Spreadsheet in read-only mode.

¹¹ Creating building blocks and TRAMs is disabled.

¹² Creating building blocks and TRAMs is disabled.

¹³ Disabled if a building block, TRAM, or diagram is selected.

¹⁴ Disabled if a building block, TRAM, or diagram is selected. Disabled also if a building block, TRAM, or diagram is subordinate to a selected object.

¹⁵ Disabled if a building block, TRAM, or diagram is selected.

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
Edit	Copy SnapShot		<input type="checkbox"/>		
Edit	Links→Start Trace Link	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Links→End Trace Link	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Links→End Trace Link Subtype	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Links→Link to Application	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Links→Link to TcEngineering	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Links→Link to TcEnterprise	<input type="checkbox"/>	<input type="checkbox"/>		
Edit	Promote	<input type="checkbox"/> ¹⁶	<input type="checkbox"/>		control-P
Edit	Demote	<input type="checkbox"/> ¹⁷	<input type="checkbox"/>		control-D
Edit	Move Up	<input type="checkbox"/> ¹⁸	<input type="checkbox"/>		
Edit	Move Down	<input type="checkbox"/> ¹⁹	<input type="checkbox"/>		
Edit	Proxy Info	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→Search	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→TcSE	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→Administration	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→Navigation Tree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→Content Table	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Go To→Notebook Pane	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

¹⁶ Disabled if a building block is selected.

¹⁷ Disabled if a building block is selected.

¹⁸ Disabled if a building block is selected.

¹⁹ Disabled if a building block is selected.

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
View	Go To→To Object	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Save	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Save As	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Set View As Default	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Module Bar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Address Bar	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Content Table	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Navigation Tree	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Notebook Pane	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Root Node	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Format Columns	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→All	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→Editable	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→Read-only	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→System	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→User Defined	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→View Specific	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Filter Properties→Toggle Controls	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Lines Per Row→1 Line	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
View	Lines Per Row→5 Lines	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Lines Per Row→10 Lines	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Lines Per Row→20 Lines	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Expand All	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
View	Refresh	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Tools	Send Email	<input type="checkbox"/>	<input type="checkbox"/>		
Tools	Change Password	<input type="checkbox"/>	<input type="checkbox"/>		
Tools	Change User Info	<input type="checkbox"/>	<input type="checkbox"/>		
Tools	Calculate Properties		<input type="checkbox"/>		
Tools	Traceability→Complying	<input type="checkbox"/>	<input type="checkbox"/>		control-T
Tools	Traceability→Defining	<input type="checkbox"/>	<input type="checkbox"/>		
Tools	Versions→Create Variant	âœ“ ²⁰	<input type="checkbox"/>		control-I
Tools	Versions→Create Version	âœ“ ²¹	<input type="checkbox"/>		control-E
Tools	Versions→Submit For Approval	âœ“ ²²	<input type="checkbox"/>		
Tools	Versions→Freeze→Selected Objects	âœ“ ²³	<input type="checkbox"/>		
Tools	Versions→Freeze→Deep	âœ“ ²⁴	<input type="checkbox"/>		

²⁰ Disabled if a building block is selected.

²¹ Disabled if a building block is selected.

²² Disabled if a building block is selected.

²³ Disabled if a building block is selected.

²⁴ Disabled if a building block is selected.

Table 5-1. Menu Options for Architect/Requirements License Types: Systems Engineering Module

Menu	Option	Requirements License	Architect License	Consumer License	Shortcut Keys
Tools	Versions→Unfreeze→Selected Objects	âœ“ ²⁵	<input type="checkbox"/>		
Tools	Versions→Unfreeze→Deep	âœ“ ²⁶	<input type="checkbox"/>		
Tools	Versions→Create Baseline	âœ“ ²⁷	<input type="checkbox"/>		
Tools	System Information	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Tools	Search	<input type="checkbox"/>	<input type="checkbox"/>		control-F
Tools	Run Macro	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	control-M
Tools	Run Report	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
Help	Manuals	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	F1
Help	About	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

²⁵ Disabled if a building block is selected.

²⁶ Disabled if a building block is selected.

²⁷ Disabled if a building block is selected.

Creating a New User

You can create a user in any project where you have **Project Administrator** privilege, including the **TcSE Administration** project. If you create a user in the **TcSE Administration** project, the user is added to the **Users** folder and project specific access can then be granted. When you create a user in any other project, the user automatically has access to that project and is added to the **Users** folder of the **TcSE Administration** project. For more information, see [Project Access](#), earlier in this chapter, and [Granting or Revoking Project Access](#), later in this chapter.

To create a new user:

1. In the navigation tree or the content table, open the **Users** folder for the project.

2.

Pull down the **File** menu and choose the **New**→**User** options to display the Create User dialog window.

3. Enter the Architect/Requirements user name in the **User Name** field.

You can enter the user's actual names in the **First name** and **Last name** fields.

In the **Email** field, you can enter the user's E-mail address to enable the user to send and receive E-mail from the Architect/Requirements client. This field must be filled in to enable the function for the user. For more information about sending E-mail from the client, see the *Systems Architect/Requirements Management User's Manual*.

4. Enter the password in the **Password** and **Re-enter Password** fields.

5.

In the **Privilege** field, do one of the following to assign the maximum privilege level:

- Select **Read Only** to allow the user only to view objects in both the Systems Engineering and Requirements Management and Administration module for all projects.
- Select **Read and Write** to allow the user to view, modify, create, and delete objects in the Systems Engineering and Requirements Management module, and to view objects in the Administration module, for all projects.
- Select **Project Administrator** to allow the user to create new projects, to delete any existing project, and to perform any action in specific projects.
- Select **Enterprise Administrator** to allow the user to create new projects, to delete any existing project, and to perform any action in all projects.



With this maximum privilege level, the user is added to the **Users** folder of every project in the Architect/Requirements installation. You must have **Enterprise Administrator** privilege to assign this maximum privilege level.

6. Click **OK** to close the dialog window and display the user in the content table.

Procedure Notes

Step 2: You can also right-click the **Users** folder in the navigation tree, or right-click anywhere in the content table, and then choose the **New**→**User** options from the popup menu. Or, click the **New User** button on the toolbar or press control-R.

Granting or Revoking Project Access

When you create a new user in a project, the user automatically has access to that project through the maximum privilege level that you assign. You can grant access to any or all other projects for which you have **Project Administrator** privilege.

For a new or existing user, you can grant or revoke access to one project at a time. You can also grant and revoke access to multiple projects in a single action.

The user is added to the **Users** folder in each project to which access is granted. When access is revoked, the user object is removed from that **Users** folder. The user object remains in the **TcSE Administration** project when access to all projects is revoked.

To grant or revoke project access:

1. In any project that contains the user object, open the **Users** folder, and then select the user in the content table.

The properties table displays all viewable properties that apply to the user.



For this project only, you can revoke access and remove the user object by pulling down **File** menu and choosing **Delete**. You can also right-click the user and choose **Delete** from the popup menu. Or, click the **Delete** button on the toolbar or press the delete key.

- 2.

In the **Values** column, double-click the **Projects** property value.

The Multi-Choice dialog window is displayed, listing all projects for which you have **Project Administrator** privilege. Checkmarks indicate the projects to which the user currently has access.

3. Do one or both of the following:
 - Check the checkbox for each project to which you want to grant access, or click **Select All** to check all checkboxes simultaneously.
 - Clear the checkbox for each project for which you want to revoke access, or click **Unselect All** to clear all checkboxes simultaneously.

You can check or clear checkboxes in any combination, and you can use the checkboxes and buttons together.

4. Click **OK** to apply the changes in the database and close the dialog window.

If the user remains in this project, the **Projects** value shows the name of each project where the user now has access. If you revoked access to this project, you can see the **Projects** value in any other project where the user has access. If you revoked access to all projects, the **Projects** value is blank in the **TcSE Administration** project.

Procedure Notes

Step 1: You can also select the user in the **Users** folder of the **TcSE Administration** project. However, you cannot delete the user from that project.

Step 4: To reverse this action, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Modifying a User Profile

A user profile consists of properties that record information about a user object. You modify a user profile by changing any of the following property values:

- - **Maximum Privilege** specifies the highest access privilege that can be granted to the user for any project. The maximum privilege level applies to all projects, whether or not the user currently has access to any project. For more information, see [Project Access](#), earlier in this chapter.
 - **User Access** specifies the access privilege that is granted to the user for this project. The access privilege for any project must be equal to or lower than the maximum privilege level. A different access privilege can be granted for each project to which the user has access. For more information, see [Project Access](#), earlier in this chapter.
 - **Additional Privilege** lists special privileges that are granted to the user for this project, such as:
 - **Script Authoring** privilege allows the user to create, edit, and run activators in the Administration module.
 - **Architect** privilege allows the user to work with building blocks, TRAMs, and diagrams in the Systems Engineering and Requirements Management module.For more information, see [Special Privileges](#) and [Licensing](#), earlier in this chapter.
- **Projects** lists all projects to which the user currently has access.
- **Email** specifies the user's E-mail address, enabling the user to send and receive E-mail from the Architect/Requirements client. For more information about this feature, see the *Systems Architect/Requirements Management User's Manual*.
- **Name** uniquely identifies the Architect/Requirements user name and applies to all transactions associated with the user. That association changes in the database when the user name is changed.
- **First Name** and **Last Name** describe additional user name information.

To modify a user profile:

1. In the **Users** folder for the project, select the user object.

The properties table displays all viewable properties that apply to the user. Values that you cannot change are dimmed in the **Value** column.
2. Do any or all of the following:
 - To change the maximum privilege level for all projects:

Double-click the **Maximum Privilege** property value to display the Single-Choice dialog window.

Check the checkbox for one of the following:

- o **No Access** deactivates the user and revokes access to all projects.
- o **Read Only** allows the user only to view objects in the Systems Engineering and Requirements Management module for all projects. The user cannot view schema objects in the Administration module.
- o **Read and Write** allows the user to view, modify, create, and delete objects for all projects.
- o **Power User** allows the user to perform certain project administration tasks in specific projects.
- o **Project Administrator** allows any action in specific projects.
- o **Enterprise Administrator** allows any action in all projects. You must have **Enterprise Administrator** privilege to see this value.

Click **OK** to apply the changes and close the dialog window.

- To change the access privilege for this project:

Double-click the **User Access** property value to display the Single-Choice dialog window.

Check the checkbox for one of the following:

- o **No Access** revokes the user's access to this project.
- o **Read Only** allows the user only to view objects in the Systems Engineering and Requirements Management module for this project. The user cannot view schema objects in the Administration module.
- o **Read and Write** allows the user to view, modify, create, and delete objects in this project.
- o **Power User** allows the user to perform certain project administration tasks in specific projects.
- o **Project Administrator** allows any action in this project.
- o **Enterprise Administrator** allows any action in all projects. You must have **Enterprise Administrator** privilege to see this value.

Click **OK** to apply the changes and close the dialog window.

- To change special privileges for this project:

Double-click the **Additional Privilege** property value to display the Multi-Choice dialog window.

Do one or both of the following:

- o Check the checkbox for each privilege that you want to add, or click **Select All** to check all checkboxes simultaneously.
- o Clear the checkbox for each privilege that you want to remove, or click **Unselect All** to clear all checkboxes simultaneously.

You can check or clear checkboxes in any combination, and you can use the checkboxes and buttons together.

- . Click **OK** to apply the changes and close the dialog window.
- To change the projects to which the user currently has access:

. Double-click the **Projects** property value to display the Multi-Choice dialog window.

. Do one or both of the following:

- o Check the checkbox for each project to which you want to grant access, or click **Select All** to check all checkboxes simultaneously.
- o Clear the checkbox for each project to which you want to revoke access, or click **Unselect All** to clear all checkboxes simultaneously.

You can check or clear checkboxes in any combination, and you can use the checkboxes and buttons together.

. Click **OK** to apply the changes and close the dialog window.

- To change the E-mail address:

. Double-click the **Email** property value to open a text field.

. Enter the E-mail address, and then press the enter key.

- To change the Architect/Requirements user name:

. Double-click the **Name** property value to open a text field.

. Enter the new user name, and then press the enter key.

In the content table, you can open a text field for the user object by pulling down the **File** menu and choosing **Rename**. You can also right-click the user object and choose **Rename** from the popup menu. Or, press the F2 key.

- To change additional user name information:

. Double-click the **First Name** or **Last Name** property value to open a text field.

. Enter the information, and then press the enter key.

Procedure Notes

Step 2: You can reverse any action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Resetting a User Password

Although users can change their own passwords in the Systems Engineering and Requirements Management module, you may need to reset a password in some situations. For example, a user who has forgotten the current password cannot supply that password before entering a new one, as is necessary in the Systems Engineering and Requirements Management module. Or, perhaps a user has left your team and you want to assign that user name to someone else, with a different password for security.

You must reset the password for a user who attempts to log in to Architect/Requirements with an expired password when Security Services is disabled. Any user whose password is reset must log in to Architect/Requirements with the new password to start the client or to continue the current session.

You enter the new password in the Change Password dialog window. The current password is not required.



The Administration module does not list user passwords.

To reset a user password:

1. In the navigation tree or the content table, open the **Users** folder for the project.
The content table displays the users who have access to the project.
2.
Select the user in the content table, pull down the **Tools** menu, and choose **Change Password**.
Architect/Requirements displays the Change Password dialog window.
3. Enter the new password in the **New Password** and **Verify New Password** fields.
4. Click **OK**.



This action clears the queue for the **Undo** option and cannot be reversed.

The dialog window closes, and a confirmation message is displayed.

Creating a User Group

A *user group* consists of shortcuts to existing user objects in the project. By creating a user group and modifying it to add the users, you can define a collection of shortcuts to individual user objects. Then you can manage those users through the shortcuts in the user group, as if you selected the actual user objects. For more information, see [Modifying a User Group](#), later in this chapter.

For a user represented by a shortcut, you can grant or revoke project access, modify the user profile, and reset the user password. For more information, see [Granting or Revoking Project Access](#), [Modifying a User Profile](#), and [Resetting a User Password](#), earlier in this chapter.

To create a user group:

1. In the navigation tree or the content table, open the **Users** folder for the project.
2. Pull down the **File** menu and choose the **New**→**Group** options.



This action cannot be reversed. A user group cannot be deleted from the database.

The content table displays the new user group with a default name in an open text field.

3. Enter the user group name, and then press the enter key.

Procedure Notes

Step 2: You can also right-click the **Users** folder in the navigation tree, or right-click anywhere in the content table, and then choose the **New**→**Group** options from the popup menu. Or, click the **New User Group** button on the toolbar or press control-G.

Step 3: You can reverse this action by pulling down the **Edit** menu and choosing **Undo Rename**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Modifying a User Group

After creating a new user group, you modify the user group to add individual users. For an existing user group, you can add and remove users at any time.

Users are associated with user groups through shortcuts to the actual user objects. Consequently, a given user can belong to many user groups concurrently and can be managed from any one. Changes to a shortcut in a user group affect the actual user object and are reflected in all other user groups to which that user belongs. When you remove users, you delete only the shortcuts that associate those users with the user group, and the user objects themselves remain in the database.



Project Administrator privilege is not required if:

- Your user name is listed in the **Modify And Read Access** property value of the security profile that is applied to the user group.
- A maximum privilege level of **Read and Write** or higher is assigned to your user object in the **TcSE Administration** folder.

To add users to a user group:

1. In the **Users** folder for the project, select each user that you want to add.

You can select nonadjacent and adjoining users.



You can drop the selection on the intended user group to add these users and complete this procedure.

2. Pull down the **Edit** menu, and choose **Copy**.
3. Select the user group, pull down the **Edit** menu, and choose **Paste**.

The content table displays the user object shortcuts below the user group.

Procedure Notes

Step 2: You can also right-click the selection and choose **Copy** from the popup menu. Or, click the **Copy** button on the toolbar or press control-C.

Step 3: You can also right-click the user group and choose **Paste** from the popup menu. Or, click the **Paste** button on the toolbar or press control-V. You can reverse this action by pulling down the **Edit** menu and choosing **Undo Copy**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

To remove users from a user group:

1. In the **Users** folder for the project, click the plus sign (+) for the user group, and then select each user that you want to remove.

You can select nonadjacent and adjoining users.

2. Pull down the **File** menu and choose **Delete** to display a confirmation message, and then click **Yes** to delete the shortcuts from the database.

Procedure Notes

Step 2: You can also right-click the selection and choose **Delete** from the popup menu. Or, click the **Delete** button on the toolbar or press the delete key. To reverse this action, you can pull down the **Edit** menu and choose **Undo Delete**, click the **Undo** button on the toolbar, or press control-Z.

Deactivating a User

User objects cannot be deleted from the database. By deactivating a user, however, you can revoke the user's access to all objects in the database. The database retains the records of all previous transactions associated with that user name.

Deactivation also removes the association between the user name and the current license. That license can then be assigned to a new user, or the license can be held in reserve. For more information, see [Licensing](#) and [Creating a New User](#), earlier in this chapter.

The user object is removed from the **Users** folder of each project to which the deactivated user had access. However, that user object remains in the **Users** folder of the **TcSE Administration** project, where project access can be granted to reactivate the user in the future if a license is available. For more information, see [Granting or Revoking Project Access](#), earlier in this chapter.

To deactivate a user:

1. In the navigation tree or the content table, open the **Users** folder for the project.
2. In the content table, select the user.

The properties table displays all viewable properties that apply to the user.

- 3.

In the **Value** column of the properties table, double-click the value for the **Maximum Privilege** property.

The Single-Choice dialog window is displayed.

4. Check the checkbox for **No Access**, and then click **OK** to close the dialog window.

The deactivated user object is removed from all projects except the **TcSE Administration** project.



For this project only, you can restore the user object and reactivate the user by immediately pulling down the **Edit** menu and choosing **Undo Change Properties**. You can also click the **Undo** button on the toolbar or press control-Z.

Although the user object is retained in the **TcSE Administration** project, the user object is not restored in any other project, nor is the user reactivated for those projects.

Emptying a User's Recycle Bin

An Architect/Requirements enterprise administrator can empty a selected user's Recycle Bin for system management purposes. For example, a Recycle Bin that is consuming excess space can be emptied to increase storage capacity in the database. Also, the administrator can empty a Recycle Bin to remove deleted instances of an object type and free the corresponding type definition for deletion.

A Recycle Bin can be emptied whether or not the user is logged on to Architect/Requirements.



Objects cannot be restored after a Recycle Bin is emptied. The objects are permanently removed from the database.



- You must have **Enterprise Administrator** access privilege to empty a user's Recycle Bin.
- An enterprise administrator can set the number of days after which all users' Recycle Bins are emptied automatically. This setting is the value of the **DB.RecycleBinTimeout** parameter, under **Web Application Configuration** in **Administrative Tools**. For more information about customizing the Architect/Requirements server, see the *Systems Architect/Requirements Management System Administrator's Manual*.

To empty a user's Recycle Bin:

1. In the **TcSE Administration** project, open the **Users** folder.
2. Select the user in the content table, pull down the **Tools** menu, and choose **Empty Recycle Bin**.

Architect/Requirements displays a message asking you to confirm this action.

3. Click **Yes**.



This action clears the queue for the **Undo** option and cannot be reversed.

Architect/Requirements removes the objects from the database.

Chapter 6: Maintaining Project Security

This chapter contains an overview of security profiles and instructions for using these profiles to define rules for user permissions within a project.

Introduction

Chapter 5, *Managing Users*, described access rights that can be granted to users at the project level. This chapter describes access rights that can be defined at the object level using security profiles. For a user, the project-level access rights are enforced first and can be limited by security profiles, but cannot be extended. For example, if a user has been granted **Read Only** access to a project, then no security profile can grant that user modification rights to any object within the project.

Security Profiles and Access Control

A security profile is a set of access rules that control user actions on specific objects in Architect/Requirements. Security profiles can be applied to objects in the Systems Engineering module and the Administration module, controlling which users can view, modify or delete the objects. A given security profile can be applied to any number of objects.

A security profile defines the users who can work with the objects to which the security profile applies. Each user's actions on those objects are determined by the permission granted to that user. User permissions can be granted individually, by specific user names, and generally, through keywords relative to all user names.

Security profiles can be automatically applied when an object is created. Applying security profile when an object is created prevents users from creating objects of a certain type or with a certain owner. If a new object inherits a security profile from its owner or gets the **Instance Security Profile** from its type definition, that security profile takes effect when the object is created. If the created users do not have at least write access in the profile, they are not allowed to create the object. If **Creator** is specified in the profile's **Full Control** or **Modify And Read Access** list, any user is able to create the object.

Individual users can be granted the following permissions:

- **Full Control** allows users to view, modify, and delete the objects. These users can also change the security profile itself.
- **Modify And Read Access** allows users to view and modify the objects. These users cannot delete the objects or change the security profile itself.
- **Read Access** allows users only to view the objects. However, these users can attach notes and create trace links to the objects if they have **Read and Write** access to the project as a whole.



Enterprise administrators have full access to all the objects in all projects, irrespective of security profiles. Users with **Project Administrator** rights in a project have full access to all the objects in that project, irrespective of security profiles.

By default, a user who is not granted any of those permissions cannot work with the objects. For that user name, the objects do not appear in the Systems Engineering module.

In addition, each permission can be granted generally by the following keywords:

- **Creator** grants the permission to the user who originally created the object.
- **Everyone** grants the permission to all users in the project.
- **No One** revokes the permission for all users in the project.



Siemens PLM Software recommends against using the **No One** keyword for the **Read Access** permission.

Security profiles can be created as needed. However, it is best to manage security profiles in a structured way that is consistent with your configuration management processes. For example:

- By project phase.

Create a security profile for each phase of the project life cycle, defining each profile with access control appropriate for that phase. For example, security profile names may correspond to project phases such as Proposal, Preliminary Design, Post-PDR, Post-CDR, and Released, with access control becoming more stringent for each successive phase.

As each object moves through the life cycle, the object's security profile setting is changed to the profile for the current phase. The security profiles themselves do not change. In addition, different types of objects may have different life cycles, and each type may have its own cycle of security profiles.

Users can change the security profile setting for an object in the Systems Engineering and Requirements Management module by editing the object's **Security Profile** property. For more information about editing properties, see the *Systems Architect/Requirements Management User's Manual*.

- By object relationships.

Create a common security profile for a set of closely related objects that move together through the project life cycle. For example, a separate security profile with unique access control may be defined for each of three requirement subtypes, such as Customer Requirements, Functional Requirements, and Design Requirements.

The security profile for a subtype is applied to those objects as they are created. As the entire set passes from one phase, the shared security profile is modified for the next phase by the project administrator.

With this method, it is not necessary to change the security profile settings for individual objects. Also, static inheritance can be used to apply security profiles to new objects automatically. For more information, see [Inherited Security](#) and [Setting Access Control Inheritance](#), later in this chapter.

Inherited Security

Where new objects are managed under the same access control as their superiors, common use cases call for inheritance. A property of security profiles, **Apply To New Descendents**, can specify that new objects reference this property value in the owner's security profile.

If the value is **MEMBER**, the new object is set to point to that same security profile, and the setting is shown in the **Security Profile** property value for the new object. That behavior applies to newly created objects, including objects that are created by copying existing objects.

When an object is created through a copy operation, the new object does not take its security profile setting from the originating object. Instead, the setting for the copy depends on the **Apply To New Descendents** property value of the security profile for the copy's owner:

- If the value is **MEMBER**, the copy inherits that security profile.
- If the value is **None**, the copy receives a blank value in its **Security Profile** property.

The behavior for move operations is more complicated, because descendants of a moved object are also moved to preserve the hierarchy:

- A moved object inherits the security profile of its new owner if:
 - The moved object and its previous owner point to the same security profile.
 - For that security profile, the **Apply To New Descendents** property value is **MEMBER**.
 - The new owner points to a security profile whose **Apply To New Descendents** property value is **MEMBER**.

Otherwise, the moved object keeps its current security profile, if any.

- For descendants of a moved parent object:
 - If a descendant points to the same security profile as its parent before the move, the descendant inherits the security profile of the parent's new owner.
 - If a descendant and the parent point to different security profiles before the move, the descendant keeps its current security profile, if any. If that descendant is itself a parent, each lower level object keeps its current security profile, if any.

To track inheritance, Architect/Requirements creates a text property named **Security Profile Reason** for the object. This property value is a text string that explains how the security profile became associated with the object.

For a moved object, for example, the value may provide information such as:

```
Inherited from superior at Move <create user name> hh:mm mm/dd/yy
```

Also, a security profile applied directly to the object may produce a value such as:

```
Set by user <change user name> hh:mm mm/dd/yy
```

The value is overwritten each time the object's security profile is changed.

Security Profile Support for Schema Objects

As with all objects in the Systems Engineering and Requirements Management module, Architect/Requirements supports the security profile mechanism on schema objects in the Administration module. Except for user objects, security profiles can be applied to all schema objects. The project administrator can apply a security profile to specific schema objects to allow certain users to modify those objects in the Administration module. For more information, see [Applying a Security Profile to a Schema Object](#), later in this chapter.



Security profiles do not keep users from viewing objects in the Administration module. To prohibit schema object visibility for a certain user, assign the **Read Only** access privilege to the **User Access** property of the user object. For more information, see [Project Access](#) and [Modifying a User Profile](#) in chapter 5, *Managing Users*.

To create activators in a project, users must have **Project Administrator** privilege and the additional **Script Authoring** privilege for the project. Users can modify an existing activator if they have **Script Authoring** privilege for the project and **Modify** permission for the activator.

Access Control for Object Properties

For a property definition in the Administration module, the **Instance Security Profile** property monitors access to all instances of that property in the Systems Engineering and Requirements Management module, regardless of the object types on which the instances appear.

- The value of the **Instance Security Profile** property is the name of a specific security profile. This security profile governs user access to all instances of the property.
When the value is blank, access to instances of this property is checked using the owner's security profile. The value is blank by default.
- User access to property instances is checked dynamically using the **Instance Security Profile** value on the property definition. The value is not copied onto every instance of the property.
- If the **Instance Security Profile** value is set for a property definition, that security profile overrides each owning object's **Security Profile** settings for instances of that property. Access rights of the two security profiles are not combined in any way. When a property's **Instance Security Profile** is set, there is no way for any object to apply any different level of access to its own value for that property.
- The **Instance Security Profile** value determines only those users who can modify the property instances. The value does not control access to view the property instances. If a user has **Read Access** to an object, the user can view all properties for an object.

Creating a Security Profile

A security profile defines the users who can work with the objects to which the security profile applies. Each user's actions on those objects are determined by the permission granted to that user.

After you create a security profile, you follow up by editing the security profile to specify users and permissions. For more information, see [Setting User Permissions](#), later in this chapter.

To create a security profile:

1. In the navigation tree or the content table, open the **Security Profile** folder for the project.
2. Pull down the **File** menu and choose the **New**→**Security Profile** options.

The content table displays the new security profile, with the temporary name in an open text field.

3. Enter a meaningful name for the security profile, and then press the enter key.

The content table displays the security profile at the bottom of the pane. To place the security profile according to the sort sequence, pull down the **View** menu and choose **Refresh**. Or, right-click the root node, and then choose **Refresh** from the popup menu.

Procedure Notes

Step 2: You can also right-click the **Security Profile** folder and choose the **New**→**Security Profile** options from the popup menu.

Setting User Permissions

A security profile defines user permissions for the objects to which the security profile applies. Permissions correspond to the following properties of the security profile:

- **Full Control** identifies the users who can view, modify, and delete the objects. These users can also change the security profile itself.
- **Modify And Read Access** identifies the users who can view and modify the objects. These users cannot delete the objects or change the security profile itself.
- **Read Access** identifies the users who can only view the objects. However, these users can attach notes and create trace links to the objects.

To set user permissions:

1. In the **Security Profiles** folder or subfolder, select the security profile.
- 2.

In the **Value** column of the **Properties** tab or window, double-click the value for a permission to display the Multi-Choice dialog window.

3. Do one or both of the following:
 - To set this permission for individual users:
 - Check the checkbox for a user to grant the permission.
 - Clear the checkbox for a user to revoke the permission.
 - To set this permission generally by a keyword:
 - Check the checkbox for **Creator** to grant the permission to the user who created the object. Or, clear the checkbox to remove this keyword.
 - Check the checkbox for **Everyone** to grant the permission to all users in the project. Or, clear the checkbox to remove this keyword.
 - Check the checkbox for **No One** to revoke this permission for all users in the project. Or, clear the checkbox to remove this keyword.



Siemens PLM Software recommends that you do not set **No One** as the only value of the **Read Access** property.



- If you check **Everyone** and one or more user names or other keywords, the permission is granted to all users in the project. The user names and other keywords are ignored.
- If you check **No One** and one or more user names or **Creator**, and if **Everyone** is not checked, the permission is granted to the specified users. **No One** is ignored.

4. Click **OK**.

The property value reflects the net result of your selections.

To set an additional permission, repeat steps 2 and 3. To reverse the setting, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Setting Access Control Inheritance

For new members of an existing object, access control can be set automatically through the **Apply To New Descendents** property of the owner's security profile. When the property value is **MEMBER**, all new members of the object inherit the same security profile.

Later changes to the owner's security profile setting do not automatically propagate down to the object's descendants. To change access control throughout an existing object hierarchy, the security profile setting of each object must be changed individually. To see what controls are in effect for an object, users can view the properties of the applicable security profile in the **Security Profiles** folder for the project. For more information about editing properties, see the *Systems Architect/Requirements Management User's Manual*.

Inheritance for moved objects involves additional considerations. For more information, see [Inherited Security](#), earlier in this chapter.



- A security profile must exist in the project. For more information, see [Creating a Security Profile](#), earlier in this chapter.
- In some cases, a user who creates an object may not be allowed to edit the new object's properties. To avoid this situation, include the **Creator** keyword in the **Modify And Read Access** property of the security profile that you assign to the owning object. For more information, see [Setting User Permissions](#), earlier in this chapter.
- Access control inheritance overrides a default security profile. For more information, see [Assigning a Default Security Profile to New Objects of a Type](#), later in this chapter.

To set access control inheritance:

1. In the **Security Profiles** folder, select the security profile that you want to assign to the owning object, or open the containing subfolder and then select the security profile.

The **Properties** tab or window displays all viewable properties for the security profile.

2.

In the **Value** column, double-click the **Apply To New Descendents** property value to display the Single-Choice dialog window.

3. Check the **MEMBER** checkbox, and then click **OK** to close the dialog window and set the property value.

4. In the Systems Engineering module, select the owning object, and then set the object's **Security Profile** property value to the security profile selected in step 1.

Assigning a Default Security Profile to New Objects of a Type

Through the **Instance Security Profile** property of a type definition, you can limit user access for all new objects of that type. In the property value, the default security profile that you assign is automatically applied to new objects that are subsequently created in the Systems Engineering module, including those created by copy and import operations.

The **Instance Security Profile** property is a system-defined property, whose initial value is blank. When a new object is created, the property value of the corresponding type definition is referenced. If the property value names a security profile, that security profile is applied to the new object by default. The default security profile can be changed by users in the Systems Engineering module.

When a subtype is created, it inherits the security profile named in the **Instance Security Profile** property for the parent type definition. Therefore, all new objects of the new subtype receive the security profile in the Systems Engineering module. For more information, see [Creating a Subtype](#) in chapter 3, *Customizing Object Types*.



- A security profile must exist in the project. For more information, see [Creating a Security Profile](#), earlier in this chapter.
- In some cases, a user who creates an object may not be allowed to edit the new object's properties. To avoid this situation, include the **Creator** keyword in the **Modify And Read Access** property of the security profile that you assign to the type definition. For more information, see [Setting User Permissions](#), earlier in this chapter.
- Access control inheritance overrides the default security profile specified in the **Instance Security Profile** property. For more information, see [Setting Access Control Inheritance](#), earlier in this chapter.

To assign a default security profile to new objects of a type:

1. In the navigation tree or the content table, open the project's **Type Definitions** folder.
The content table displays the base type definitions. To see lower level subtypes in a hierarchy, click the plus signs in the **Types/SubTypes** column.
2. In the content table, select the type definition.
The **Properties** tab or window displays all viewable properties for the type definition.
3.
In the **Value** column, double-click the **Instance Security Profile** property value to display the Single-Choice dialog window.
4. Check the checkbox for the security profile, and then click **OK** to close the dialog window and set the property value.
To reverse this action, you can pull down the **Edit** menu and choose **Undo Change Properties**, click the **Undo** button on the toolbar, or press control-Z.

Applying a Security Profile to a Schema Object

If a security profile is not applied to a given schema object, only users with **Project Administrator** privilege can modify that object. By applying a security profile, you can give certain users permission to modify the object, without designating additional project administrators.



Security profiles do not keep users from viewing objects in the Administration module. To prohibit schema object visibility for a certain user, assign the **Read Only** access privilege to the **User Access** property of the user object. For more information, see [Project Access](#) and [Modifying a User Profile](#) in chapter 5, *Managing Users*.

You can apply a security profile to any schema object in the following primary folders:

- **Activators**
- **Property Definitions**
- **Reports and Formatting**
- **Security Profiles**
- **Type Definitions**

In the **Users** folder, security profiles can be applied to user groups to enforce access control on those objects. However, a security profile has no effect when applied to an individual user object.

To apply a security profile to a schema object:

1. In the navigation tree, open the primary folder that contains the object.
The content table displays the schema objects and subfolders in the primary folder.
2. In the content table, select the object to which you want to apply the security profile.
The **Properties** tab or window displays all viewable properties for the object.
3.
In the **Value** column, double-click the **Security Profile** property value to display the Single-Choice dialog window.
4. Check the checkbox for the security profile, and then click **OK** or press the enter key to close the dialog window and set the property value.

You can reverse this action by pulling down the **Edit** menu and choosing **Undo Change Properties**, by clicking the **Undo** button on the toolbar, or by pressing control-Z.

Chapter 7: Configuring the Change Management Package

This chapter provides an overview of the Architect/Requirements change management package and contains instructions for setting up the change approval process and for activating change logs for type definitions.

Change Approval

From the Systems Engineering module, requirements and building blocks can be submitted through the Architect/Requirements change management package for change approval. A submitted object may be either of the following:

- An object with no prior versions is a new submission. When the changes are submitted, the object is frozen for the first time.
- An object with prior versions is a revision to previously submitted content. When the changes are submitted, this revision is frozen.



To use the change approval process, the **Version** package must be enabled for the project. For more information about enabling versions for a project, see [Enabling the Versions Package for the Project](#), later in this chapter.

The change approval process tracks the following events:

- - A **Submit** event occurs when a requirement or a building block is submitted for approval. E-mail is sent to recipients in the following categories:
 - *Approvers* have the responsibility to approve or reject the changes. Each of these users receives an E-mail message containing two hyperlinks:
 - A hyperlink to the object in the Architect/Requirements client. The user can click this hyperlink to navigate to the object and review the changes.
 - A hyperlink to the change management package. The user can click this hyperlink to navigate to the approval and rejection page in Microsoft Internet Explorer.

Notifiers do not approve or reject the changes, but are notified of submitted changes for informational purposes. Each of these users receives an E-mail message containing a hyperlink to the submitted object in the Architect/Requirements client. The user can click the hyperlink to navigate to the object and review the changes.

Each message also contains the submitter's comments, if any. In addition, each approver and notifier receives the object's content in **.mhtml** format as an E-mail attachment. For an object that has prior versions, the previously submitted content is included.

The **Submit** event automatically creates a *change approval object* for the submitted object. Displayed in the **Attachments** tab or window, the change approval object contains a table of information related to the approval process. The initial table row is added for the **Submit** event.

●

A **Response** event occurs each time an approval or a rejection is received.

For each response, a row is added to the change approval object that is attached to the submitted object. Delinquent responses are also tracked.

●

An **Approved** event occurs if the changes are approved by all approvers. The approved object remains frozen. An E-mail message is sent to each approver and to the submitter, stating that the changes are approved.

The **Approved** event concludes the approval process, and a final row is added to the change approval object.

●

A **Rejected** event occurs each time the changes are rejected by one approver. When the first **Rejected** event occurs, the rejected object is unfrozen and remains in that state.

For each **Rejected** event:

- An E-mail message is sent to the submitter only, stating that the changes are rejected by the individual approver.
- A row is added to the change approval object for this event.

If other **Response** events are received after the first rejection, each later event is added to the change approval object.

When changes are submitted, an E-mail message is sent to the approvers and notifiers. The approvers and notifiers must be identified in a security profile, and that security profile must be applied to the submitted object in the Systems Engineering module. Users can apply the security profile through the object's **Security Profile** property in the **Properties** tab or window.



Each approver automatically has full access to the changes until that user approves or rejects the change request.

A change approval object is automatically generated for a requirement or a building block that is submitted for approval. The change approval object contains a table in which the rows show all responses that are currently received. Each row shows the time of response, the responding user name, the user's comments, and the response type. A new row is added for each subsequent approval or rejection. If the changes are approved, a final row is added for that event. The change approval object is displayed in the **Attachments** tab or window in the Systems Engineering module.



The change management package can be customized with the Architect/Requirements application programming interface (API). For more information, see the *Systems Architect/Requirements Management API Reference*.

Enabling the Versions Package for the Project

In the Administration module, the **Packages** property for the project determines whether versions are enabled. When the **Version** value is added, users can do the following in the Systems Engineering and Requirements Management module:

- Create versions of requirements, building blocks, and their subtypes, and also create versions of prior versions of those objects.
- Choose effectivity rules through controls displayed in the **Address** bar.
- View a version tree in the **Versions** tab and window in the notebook pane.
- Create baselines of requirements, building blocks, and their subtypes.
- View a baseline in the hierarchical content table.

For more information about working with versions, see the *Systems Architect/Requirements Management User's Manual*.



Once the **Version** package is added, it cannot be removed from the project.

To enable the Versions package for the project:

1. In the navigation tree, select the **Projects** node.
The content table displays all projects to which you have access.
2. Select the project in the content table.
The **Properties** tab or window displays all viewable properties for the project.
3.
In the **Value** column, double-click the **Packages** property value.
The Multi-Choice dialog window is displayed.
4. Check the **Version** checkbox, and then click **OK** to close the dialog window.
A confirmation message states that the **Version** package cannot be removed, and asks if you want to continue.
5. Click **Yes**.



This action cannot be reversed. The **Version** package cannot be removed from the project.

The **Version** value is added to the **Packages** property, indicating that versions are enabled for the project.

Enabling Effectivity-Sensitive References for a Project

A reference to a versionable object can be either fixed or dynamic. Fixed references always get the information from the same version of the object. Dynamic references get the information from the currently effective version of the object. For more information about the behavior of reference links, see *Systems Architect/Requirements Management User's Manual*.

There are two ways to specify how dynamic links behave. In the Administration module, the **Project Settings** property of a project controls the default behavior for reference links in that project. Selecting the **Promote References** option indicates dynamic references. By default, a project uses fixed references.

The behavior for a specific link can be controlled with the **Reference Settings** property (which can be accessed by right-clicking a reference link). Double-clicking the **Reference Settings** property allows you to select or deselect the **Promote References** value. The **Reference Settings** property allows a specific reference link to override the default behavior.

The use of fixed or dynamic references does not affect how the references are stored in the database. The setting for a project can be changed at any time to change the behavior of the existing references.



Although the **Reference Settings** property is a multi-select property, the **Promote References** and **Fixed References** options are mutually exclusive. If you select both the options, **Promote References** takes precedence.

The **Reference Settings** property requires **Read & Write** privileges to the objects for which the reference link is created. When the **Project Settings** property is set to **Promote References**, the **Reference Settings** property on a reference link is displayed as blank. In this case, enabling or disabling **Promote References** does not matter because the project-wide behavior of **Promote References** is honored.



The **Promote References** behavior is supported for references to versionable objects, requirements, buildings blocks, and shortcuts to versionable objects. References to notes do not support the **Promote References** behavior.

When the **Promote References** option is enabled, reference links are sensitive to effectivity. The symbolic references track the effective version of the referenced object. This behavior occurs whether the referencing object is frozen or not. When effectivity is changed, the symbolically referenced information changes to be consistent with the new effectivity. The effectively referenced version is used when a requirement is opened in Microsoft Office Word, shown in the preview pane, or exported to a document.

For example, consider a frozen object A referencing a non-frozen object B. B1 is created as the new version of B. The similarities and differences in the resolution of reference links with and without enabling the **Promote References** option setting can be seen in the **Links→Relations** and the **Where Used** tabs in the notebook pane.

Table 7-1 describes the resolution of the reference links with the **Promote References** option disabled. Table 7-2 describes the resolution of the reference links with the **Promote References** option enabled.

Table 7-1. Reference Links with Promote References Disabled

Tab in the Notebook pane	Effectivity: Current Version	Effectivity: Current Frozen Version (same behavior)
Links→Relations	Selecting A displays B as the target.	Selecting A displays B as the target.

Table 7-1. Reference Links with Promote References Disabled

	Selecting B1 displays no reference.	Selecting B displays A as the source.
Where Used	Selecting B1 displays no reference.	Selecting B displays A.

Table 7-2. Reference Links with Promote References Enabled

Tab in the Notebook pane	Effectivity: Current Version	Effectivity: Current Frozen Version (same behavior)
Links→Relations	Selecting A displays B1 as the target. Selecting B1 displays A as the source.	Selecting A displays B as the target. Selecting B displays A as the source.
Where Used	Selecting B1 displays no reference.	Selecting B displays A.

Setting Up the Change Approval Process

The change approval process uses E-mail for notification of change submissions, approvals, and rejections. Therefore, the following E-mail information must be supplied:

- The IP address of your E-mail server.

The E-mail server IP address is a Web application configuration parameter of the Architect/Requirements administrative tools. You gain access to these tools through Microsoft Internet Explorer.

 **Enterprise Administrator** privilege is required to enter the E-mail server IP address and all other configuration parameters.

- A valid E-mail address for each approver, notifier, and submitter. You enter these E-mail addresses in the **Users** folder for the project in the Administration module.

In a security profile, you designate users and user groups in the following categories:

- Approvers have the responsibility to approve or reject changes.
 - Notifiers only receive notification of submitted changes for informational purposes.

You can create a new security profile especially for this change approval process. Or, you can modify an existing security profile. For more information, see [Creating a Security Profile](#) or [Setting User Permissions](#) in chapter ***Unsatisfied xref reference***, ***Unsatisfied xref title***.



Each approver automatically has full access to the changes until that user approves or rejects the change request.

The security profile must be applied to each requirement and building block that is submitted for change approval in the Systems Engineering and Requirements Management module. Users can apply the security profile through the object's **Security Profile** property in the **Properties** tab or window.



You must enable the **Version** package for the project before changes can be submitted for approval. For more information, see [Enabling the Versions Package for the Project](#), earlier in this chapter.

To set up the change approval process:

1. To enter the E-mail server IP address, do the following in Internet Explorer:



You must have **Enterprise Administrator** privilege to perform this step.

- a. Open the Architect/Requirements home page, and then click the **Administrative Tools** hyperlink.

The Administrative Tools page is displayed.

- b. On the Administrative Tools page, click the **Web Application Configuration** hyperlink.

The Architect/Requirements log in page is displayed.

- c. Enter your Architect/Requirements user name and password, select a language, and click **Log In**.

The Architect/Requirements Configuration Parameters page is displayed.

- d. In the **MailServerIP** field, enter the E-mail server IP address.

- e. At the bottom of the page, click the **Update** button.

The Architect/Requirements Configuration Parameters page is displayed with read-only information for verification.

- f. Do one of the following:

- To verify that the E-mail server IP address is correct, click the **Ok** button.

The Architect/Requirements Configuration Parameters page is displayed, and the E-mail server IP address is entered for the change approval process.

- To change the E-mail server IP address, click the **Back** button to redisplay the fields on the Architect/Requirements Configuration Parameters page. Then repeat steps d through f.

2. To enter E-mail addresses for approvers, notifiers, and submitters, do the following in the Administration module:

- a. In the navigation tree, select the **Users** folder for the project.

The content table displays all users in the project.

- b. Do the following for each approver, notifier, and submitter:

- Select the user object in the content table.
The **Properties** tab or window displays all viewable properties for the user.
- In the **Value** column, double-click the **Email** property value to open a text field.
- Enter the user's E-mail address in the text field, and then press the enter key.

Also in the **Users** folder, you can give users permission to modify the activators related to change approval process. For each of these users, select the user object in the content table, and double-click the **Additional Privilege** property value to display the Multi-Choice dialog window. Check the checkbox for **Script Authoring**, and then click **OK** or press the enter key.

In the **Activators** folder, the related activators are the following:

c.

Change Submit executes when a user submits changes for approval in the Systems Engineering and Requirements Management module.

d.

Change Response executes when an approval or a rejection is received.

e.

Change Approved executes when all approvers listed in the security profile have approved the changes.

f.

Change Rejection executes when any one approver rejects the changes.

3. To designate the approvers and notifiers, do the following in the Administration module:

a. In the navigation tree, select the **Security Profiles** folder for the project.

The content table displays all security profiles in the project.

b. In the content table, select the security profile that you want to use for this change approval process.

The **Properties** tab or window displays all viewable properties for the security profile.

c. Do one or both of the following:

- To designate approvers:

c.

In the **Value** column, double-click the **Change Approvers** property value to display the Multi-Choice dialog window.

c. Check the checkbox for each user and user group that you designate as an approver, and clear the checkbox for each user and user group that you want to exclude.

You can use the checkboxes in conjunction with the buttons. Click **Select All** to check all checkboxes simultaneously. Click **Unselect All** to clear all checkboxes simultaneously.

c. To close the dialog window and apply your choices, click **OK** or press the enter key.



Each approver automatically has full access to the changes until that user approves or rejects the change request.

- To designate notifiers:

c.

In the **Value** column, double-click the **Change Notifiers** property value to display the Multi-Choice dialog window.

- c. Check the checkbox for each user and user group that you designate as a notifier, and clear the checkbox for each user and user group that you want to exclude.

You can use the checkboxes in conjunction with the buttons. Click **Select All** to check all checkboxes simultaneously. Click **Unselect All** to clear all checkboxes simultaneously.

- c. To close the dialog window and apply your choices, click **OK** or press the enter key.



Notifiers must have a minimum privilege of **Read Access** for the security profile. If necessary, double-click the **Read Access** property value and check the checkbox for each designated notifier who does not have the minimum privilege.

Change Logs

A change log captures the history of modifications to an object of a given type definition. Change logs can be activated for folders, requirements, building blocks, groups, notes, trace links, connections, diagrams, and spreadsheets.

For each type definition and user-defined subtype, the project administrator can specify the change events that are tracked by the change log. In the Systems Engineering module, the change log is automatically created for an object of that type when a specified change event occurs. For the object selected in the content table, the change log is displayed in the **Attachments** tab and floating window.

The change log contains a table in which the rows show all change events that are currently recorded. Each row shows the date and time of change, the user who made the change, and the type of change. For object property changes that are written to the change log, the log also shows the previous and new values. A row is added for each later change event for that object.



- The date and time shown for each change reflects the time zone and locale of the Architect/Requirements server. This time zone and locale may differ from that of the Architect/Requirements client.
- Change logs can be edited only by Architect/Requirements enterprise administrators and project administrators.

Enabling Change Logs for the Project

Change logs must be enabled for the project before change logs can be generated in the Systems Engineering module. You enable change logs in the Administration module by adding the **Change Log** value to the **Packages** property for the project. After you set up change logs for type definitions, modifications to the related objects can be tracked in the Systems Engineering module.

To enable change logs for the project:

1. In the navigation tree, select the **Projects** node.
The content table displays all projects to which you have access.
2. Select the project in the content table.
The **Properties** tab or window displays all viewable properties for the project.
3.
In the **Value** column, double-click the **Packages** property value.
The Multi-Choice dialog window is displayed.
4. Check the **Change Log** checkbox, and then click **OK**.
The dialog window closes, and the **Change Log** value is added to the **Packages** property, indicating that change logs are enabled for the project.



Use change logs cautiously because they add database and processing overhead. To minimize the overhead, the administration controls for enabling change logging are very fine-grained. Enable change logging only on the object types where detailed change tracking has a necessary business purpose. For each of those types, enable change logging only for the events and properties that are essential for that type.

Setting Up Change Logs for a Type Definition

Change logs allow you to specify which change events are tracked for all objects of a given type definition. For an object of that type in the Systems Engineering and Requirements Management module, a change log is generated automatically when the first specified change event occurs. A row is added to the change log for each specified change event that subsequently occurs. Users can view the change log in the **Attachments** tab or floating window. For more information about viewing a change log, see the *Systems Architect/Requirements Management User's Manual*.

Change logs must be enabled for the project before change logs can be generated in the Systems Engineering module. After you enable change logs, you can set up change logs for folders, requirements, building blocks, groups, notes, trace links, connections, diagrams, and spreadsheets. Also, you can set up change logs for system-defined and user-defined subtypes. For more information, see [Overview of Type Definitions](#) and [Creating a Subtype](#) in chapter 3, *Customizing Object Types*.

To set up change logs for a type definition:

1. In the navigation tree or the content table, open **Type Definitions** folder for the project.
The content table displays the base type definitions. To see lower level subtypes in a hierarchy, click the plus signs (+) in the **Types/SubTypes** column.
2. In the content table, select the type definition.

The **Properties** tab or window displays all viewable properties that apply to the type definition.

3.

In the **Value** column, double-click the **Change Log** property value.

The Multi-Choice dialog window displays the change events that are defined for the project.

4. Check the checkbox for each change event that you want to add to the change log.

To remove a single change event, clear the checkbox.

You can use the checkboxes in conjunction with the buttons:

- Click **Select All** to check all checkboxes simultaneously.
- Click **Unselect All** to clear all checkboxes simultaneously.

5. To close the dialog window and apply your choices, click **OK**.

The change events are added to the **Change Log** property value for the type definition.

Change Event Flags

Below is a list of the system defined change events.

Event	Description
Add Complying Event	The end object for a new trace link or connection
Add Defining Event	The start object for a new trace link or connection
Add Incoming Event	A generic link has been created ending at this object
Add Member Event	A child is added to a container
Add Note Event	A note or other attachment is added
Add Outgoing Event	A generic link has been created starting from this object
Add Where Used Event	A short cut or group membership link is created to an object
Copy Object Event	An object is created by a copy operation
Delete Object Event	An object is deleted
Freeze Event	A requirement or building block is frozen
Modify Name Event	An object's name is changed
Modify ROIN Event	A requirement's ROIN is changed
Modify Security Event	A Security Profile property is set
Modify Text Event	The text content of a requirement or note is changed
Modify Type Definition Event	The Subtype of an object is changed
Move Object Event	An object is moved to a new owner
New Object Event	An object is created
New Variant Event	A variant of a requirement or building block is created

New Version Event	A version of a requirement or building block is created
Remove Complying Event	A trace link or connection to this object is deleted or removed
Remove Defining Event	A trace link or connection from this object is deleted
Remove Incoming Event	A generic link starting from this object has been deleted
Remove Member Event	An object is deleted or moved from a container
Remove Note Event	A note or other attachment is deleted or moved
Remove Outgoing Event	A generic link ending at this object has been deleted
Remove Where Used Event	A short cut or group membership link is deleted from an object
Restore Object Event	Object is restored from the recycle bin
Unfreeze Event	A requirement or building block is frozen

In addition to the system defined events all property instances defined for the selected type definition will appear as change events. These events are triggered whenever an instance of that property is modified. The list of property instances can be seen in the type definition's *Properties* property.



Processing a large number of change event can be time consuming and may hurt performance. Only set change flags which are needed.

Transaction Management

Requirements Service Transaction Logging

Architect/Requirements has an enhancement to the log server functionality that adds trace records for transaction start and end events associated with requirements service method invocations. You can use this information for tracking down issues related to server usage or performance. The following example shows the trace records associated with an invocation of the requirements service's **setObject** method:

Time(mm:dd:yyyy; hh:mm:ss:ms)	Thread	UserName	SessionName	className	originatingMethod	Level	Message
11-06-2012 23:42:27:146	http-bio-8080-exec-10	eriadm	session0	setObject	Transaction Started	TRACE	USER_PREFERENCES
11-06-2012 23:42:27:148	http-bio-8080-exec-10	eriadm	session0	setObject	Transaction End	TRACE	Success, USER_PREFERENCES

In the example, the following information is displayed:

Time(mm:dd:yyyy; hh:mm:ss:ms) Time on the Architect/Requirements server when the event was logged.

Thread Thread in the web server that is hosting the Architect/Requirements application. For example, Tomcat, WebSphere, or any other application server according to your configuration.

UserName	Architect/Requirements user responsible for the current transaction.
SessionName	Versant session name. It allows you to connect message with Versant administrator commands such as db tools. This information helps you in troubleshooting Versant issues.
ClassName	Requirement service method name.
originatingMethod	API called in the transaction.
Level	Distinguishes the message from other messages being logged. It is TRACE for such messages.
Message	Contains selected parameter information about the call. They provide additional information regarding the transaction. The end transaction method indicates success or failure for the transaction.

To enable the requirements service transaction logging, set the web application configuration parameter **Log.Server.Trace** to **true**.

It is not necessary to restart the Architect/Requirements application server to enable or disable this feature.

Requirements Service Transaction Monitoring

Architect/Requirements allows you to monitor the current requirements service transactions.

To view a snapshot summary of requirements service transactions currently in progress:

1. On the Architect/Requirements home page, click **Administrative Tools**.
2. On the **Administrative Tools** page, click **Diagnostic Tools**.
3. On the **Diagnostic Tools** page, scroll down and click **List Current Transactions**.

Following is an example of the **List Current Transactions** page:

Active Transaction on PNI6W1793:8080 at Wed Apr 10 21:45:39 IST 2013						
Active Transaction = 2						
Start Time	Duration (Min)	Thread	User Name	Session Name	Operation	Additional Info
04-10-2013 21:12:45:931	32	http-8080-5	tcradm	session2	runActivator	Wait until cancel null
04-10-2013 21:12:54:295	32	http-8080-2	tcradm	session3	runActivator	Enter wait time null

The following information is displayed about the current transactions:

Start Time Time on the Architect/Requirements server when the transaction started.

Duration (Min)	Duration (in minutes) for which the transaction is running.
Thread	Thread in the web server that is hosting the Architect/Requirements application. For example, Tomcat, WebSphere, or any other application server according to your configuration.
User Name	Architect/Requirements user responsible for the current transaction.
Session Name	Versant session name. It allows you to connect message with Versant administrator commands such as db tools. This information helps you in troubleshooting Versant issues.
Operation	Requirement service method name.
Additional Info	Some additional debug information.

Requirements service transaction monitoring is always enabled. It helps you in identifying transactions running for a long time on Architect/Requirements. Use your web browser to print or refresh the snapshot.

When troubleshooting a performance related issue on your Architect/Requirements instance, you can use this feature in conjunction with requirements service transaction logging. Once the transactions are complete, they are not displayed in the requirements service transaction monitoring report. However, you can refer the transaction start and end records in the requirements service transaction log.

Appendix A: Glossary

This appendix defines Architect/Requirements terms.

A

Access Control

Protection of objects from viewing, modification, or deletion by unauthorized users. Access control is enforced through user access privileges and security profiles. See also *Access Privilege*, *Maximum Privilege Level*, and *Security Profile*.

Access Privilege

User's level of access to a specific project. The access privilege for any project cannot be higher than the user's maximum privilege level. The access privilege must be equal to or lower than the maximum privilege level. A user can be assigned a different access privilege for each project. Compare with *Maximum Privilege Level*.

C

Circular Reference

Defining trace link from a complying object back to its defining object. Circular references are allowed only if the defining and complying trace links are of different subtypes. See also *Complying Object*, *Defining Object*, and *Trace Link*.

Creator

Security profile identifier for the user who created the object to which the security profile applies. This general identifier can be used instead of a specific user name in assigning a permission to the object's creator. Compare with *Everyone* and *No One*. See also *Security Profile*.

E

Enterprise Administrator

Access privilege with which the user can perform any action on all objects in all projects. Compare with *Project Administrator*. See also *Access Privilege* and *Maximum Privilege Level*.

Everyone

Security profile identifier representing all Architect/Requirements users. For the object to which the security profile applies, this general identifier can be used to assign a permission globally to all users. Compare with *Creator* and *No One*. See also *Security Profile*.

F

Full Control

Security profile rule that allows the user to view, modify, and delete the object to which the security profile applies. **Full** permission also allows the user to modify the security profile itself. Compare with *Modify Permission* and *Read Permission*.

M

Maximum Privilege Level

Highest access privilege that can be granted to a specific user for any project in Architect/Requirements. The user's maximum privilege level applies to all projects, whether or not the user currently has access to any project. Compare with *Access Privilege*.

Modify Permission

Security profile rule that allows the user to view and modify the object to which the security profile applies. **Modify** permission prevents the user from deleting the object and from modifying the security profile itself. Compare with *Full Permission* and *Read Permission*.

N

No Access Privilege

Access privilege denying the user access to a specific project. For that user name, the project does not appear in the Systems Engineering and Requirements Management module. Compare with *Read and Write Privilege* and *Read Only Privilege*. See also *Access Privilege* and *Maximum Privilege Level*.

No One

Security profile identifier denying a permission to all Architect/Requirements users. No user has the specified permission for the object to which the security profile applies. Compare with *Creator* and *Everyone*. See also *Security Profile*.

P

Permission

Rights that determine the actions that a user can perform on individual objects. Compare with *Privilege*.

Privilege

Rights that determine a user's access to specific projects. Compare with *Permission*.

Project Administrator

Access privilege with which the user can perform any action on all objects in a specific project. Compare with *Enterprise Administrator*. See also *Access Privilege* and *Maximum Privilege Level*.

Property Definition

Set of properties that apply to all objects of a given type. Certain property definitions are automatically assigned to each object type. These system-defined property definitions always apply and cannot be modified. A project administrator can extend the system-defined properties by creating custom property definitions and applying those properties to object types. See also *Subtype* and *Type Definition*.

R

Read and Write Privilege

Access privilege with which the user can view and modify, but not delete, the objects in a specific project. Compare with *No Access Privilege* and *Read Only Privilege*. See also *Access Privilege* and *Maximum Privilege Level*.

Read Only Privilege

Access privilege with which the user can only view, not modify, the objects in a specific project. Compare with *No Access Privilege* and *Read and Write Privilege*. See also *Access Privilege* and *Maximum Privilege Level*.

Read Permission

Security profile rule that allows the user to only view the object to which the security profile applies. A user with **Read** permission can attach notes to the object and can create trace links to the object. However, the user cannot modify or delete the object, and cannot modify the security profile itself. Compare with *Full Permission* and *Modify Permission*.

S

Security Profile

Set of access rules that control user actions on individual objects. A security profile defines a specific set of users, and specifies each user's permission for the objects to which the security profile applies. A given security profile can be applied to any number of objects. See also *Creator*, *Everyone*, *Full Permission*, *Modify Permission*, *No One*, and *Read Permission*.

Subtype

Custom object type definition based on system-defined object type definition. Project administrators can create subtypes of folders, requirements, and notes. Each subtype inherits the system-defined properties of the built-in object type on which the subtype is based. In addition, project administrators can apply user-defined properties to each subtype. Users can assign subtypes when creating new objects in the Systems Engineering and Requirements Management module. Users can also assign subtypes by changing the **Subtype** property of existing objects. See also *Property Definition* and *Type Definition*.

T

Type Definition

Stores the list of properties that apply to all objects of a given object type. Each project has its own set of type definitions, so the customization is done on a project by project basis. Project administrators can customize object types by editing the properties that apply to the type definition. See also *Property Definition* and *Subtype*.

U

User

Object that represents an individual who is registered in the Architect/Requirements database. Licensing and access to projects are determined by each user's access privilege. See also *User Profile*.

User Access

See *Access Privilege*.

User Profile

Set of properties that record information about a user. Those properties are **Maximum Privilege**, **User Access**, **Security Profile**, **User Name**, **First Name**, and **Last Name**. See also *User*.

Appendix B: System-Defined Properties in the Administration Module

This appendix describes the system-defined properties of the object types used in the administration of a project.

Overview of System-Defined Properties

Architect/Requirements automatically assigns system-defined properties to all object types in both modules. In the Administration module, the object types are:

- Property definitions
- Type definitions
- Reports, templates, and style sheets
- Users and user groups
- Security profiles
- Activators

System-defined properties fall into the following categories:

- Read-only, with a value that cannot be changed.
- Editable, with a default value that can be changed by a project administrator.

Table of System-Defined Properties

In the Administration module, all viewable system-defined properties that apply to the selected object are displayed in the properties table, below the content table.

Table B-1 lists each system-defined property, including its application and description.



Attempting to remove system-defined properties from the applicable schema objects may have unintended consequences.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
Activators	Type Definition	Object that triggers an action. This property is editable.
Additional Privilege	Access Link, User	Additional privilege assigned to a user or required for a project. Values are Script Authoring , Architect , and DFSS Package . This property is editable.
Apply To New Descendents	Security Profile	Determines whether the security profile is inherited by new members of the object to which the security profile is applied. Values are the following: MEMBER All new members of the object inherit this security profile. None New members of the object do not inherit a security profile. This property is editable.
Change Approvers	Change Approval, Security Profile	List of change approvers. This property is editable.
Change Log	Type Definition	Captures the history of modifications to an object of a given type. This property is editable.
Change Notifiers	Change Approval, Security Profile	List of change notifiers. This property is editable.
Change Time	Type Definition, Property Definition, Security Profile	Date and time when the object was last modified. This property is read-only. Change events depend on the object type: <ul style="list-style-type: none"> For a type definition, the value is updated when a subtype is created as a child of the type definition.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
		<ul style="list-style-type: none"> For a property definition, the value is updated when the property definition is added to a type definition. For a security profile, the value is updated when a user or a user group is added to or removed from the security profile. <p>For more information, see Updates to Change Time and Change User Properties in the chapter <i>**Unsatisfied xref reference**</i>, <i>**Unsatisfied xref title**</i>.</p>
Change User	Type Definition, Property Definition, Security Profile	<p>Login name of the user who last modified the object. Change events for this property are the same as those described above for the Change Time property. This property is read-only.</p> <p>For more information, see Updates to Change Time and Change User Properties in the chapter <i>**Unsatisfied xref reference**</i>, <i>**Unsatisfied xref title**</i>.</p>
Choice List	Choice Definition	List of choices for this choice property. This property is editable.
Complying Objects Count	All	<p>Number of complying objects. This property is read-only.</p> <p>This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.</p>
Content	Diagram, Spreadsheet	<p>Binary content of the object. This property is editable.</p> <p>This is a non-displayed property. The name can be entered in tags for object templates and Excel templates.</p>
Content Type	Text property definition	Identifies whether the text property value is a plain string or a URL. If it is a URL, the value is presented to the user as a hyperlink. This property is editable.
Create Time	All	Date and time when the object was created. This property is read-only.
Create User	All	Login name of the user who created the object. This property is read-only.
Current Access	Access Link	Choice attribute object describing the access this user has to the project being accessed. This property is editable.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
Current Value	All property definitions	Default value of a property definition. This property is editable.
Database Class Name	Type Definition	Versant class name for objects of this type. This property is read-only.
Date Format	Date Definition	Valid format for this property value. This property is editable.
Default View	Folder Type Definition	Named view assigned by default to instances of this folder type definition in the Systems Engineering and Requirements Management module. When a folder is selected in the navigation tree, the view determines the column settings in the content table. Users can override the default view and can select other views. This property is editable.
Defining Objects Count	All	Number of defining objects. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.
Description	Search, objects in other Teamcenter applications	Text describing an external object (in another Teamcenter application) that is referenced by an object in Architect/Requirements. This property is editable.
Document Template Rules	Document Template	Controls display of full content reference links as hyperlinks in the Preview tab and window. This property is editable.
Effectivity Date	User	Date as of which to display versions for this user. Versions created from this date to the present are displayed. This property is editable. This property automatically reflects the user's latest settings in the Effectivity field in the Systems Engineering and Requirements Management module. Conversely, editing this property in the Administration module changes the user's settings in the Effectivity field.
Effectivity Label	User	Name of the baseline to display for this user. Versions assigned to this baseline are displayed. This property is editable. This property automatically reflects the user's latest settings in the Effectivity field in the Systems

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
		Engineering and Requirements Management module. Conversely, editing this property in the Administration module changes the user's settings in the Effectivity field.
Effectivity Type	User	Effectivity rule under which objects are displayed for this user. Values are Current Frozen Version, Current Version, Frozen As Of Date, Version As Of Date, Baseline , and Baseline with in-Work . This property is editable. This property automatically reflects the user's latest settings in the Effectivity list in the Systems Engineering and Requirements Management module. Conversely, editing this property in the Administration module changes the user's settings in the Effectivity list.
Email	User	User's E-mail address. This property is editable.
Error String	Excel Template	Text string placed in the property columns of the Excel export file if properties in an Excel template do not exist in the database. The default string is <i>Property does not exist</i> . This property is editable.
Events	Activator, Macro	Identifies which events cause the activator or macro to run, for example, After Modify or Before Delete . This property is editable.
Excel Template Rules	Excel Template	Controls the placement of data in a Microsoft Excel export file. With the Apply Packing value, data for two or more objects is placed on the same row. This property is editable.
First Name	User	User's first name. This property is editable.
Form Values	Activator, Macro	List of properties for which the user is prompted when running an activator or a macro. This property is editable.
Format	Numeric Definition	Valid format for this property value. This property is editable.
Formula	Numeric Definition	Allows the property value to be calculated based on subordinate objects. Values are Average, Maximum, Maximum, Multiply , and Sum . This property is editable.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
Full Control	Security Profile	List of users who can perform any action on objects to which this security profile applies. This property is editable.
Has Trace Links	All	Identifies whether the object has trace links or not. True indicates that the object has trace links, while False indicates that there are no trace links for the object.
HTML	Requirement Type Definition, Note Type Definition	Full content of a requirement, paragraph, or note, including graphics and tables. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates.
Icon	Type Definition	Graphic identifying objects of this type. The default icon for a requirement, for example, is a puzzle piece. This property is editable.
Input Required	Choice Definition	Determines if the property in the Systems Engineering module can have a blank value or if at least one choice is required. This property is editable. The No checkbox is checked by default, which allows the choice property value to be blank and allows users to clear existing choices. When the Yes checkbox is checked, the value of each occurrence of the choice property is retained. The next time each value is edited, at least one choice is required. Otherwise, the previous value is retained and an error message is displayed.
Instance Security Profile	All	Security profile assigned to an instance of an object. This property is editable.
Last Name	User	User's last name. This property is editable.
Launcher URL	All	Object URL used to launch the Architect/Requirements client when navigating to the object. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.
LightWeight Properties	Individual objects selected in the Systems Engineering and Requirements Management module	Text property that can be created for any object and can be viewed in the Architect/Requirements client. The flexibility is useful for objects that are used in interfaces with other Teamcenter products. This property is editable only through the Architect/Requirements API.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
		For more information about using Tcl or the Java API, see the <i>Systems Architect/Requirements Management API Reference</i> manual.
Maximum Privilege	User	Maximum access privilege that this user can have to any project. This property is editable.
Maximum Privilege Level	User	Maximum access privilege that this user is allowed in the current project. This property is editable.
MHTML	Requirement Type Definition, Note Type Definition	Full content of a requirement, paragraph, or note, including graphics and tables. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates.
Menu Text	Menu Folder, Menu Item	Text of the menu or menu item as it appears in the Architect/Requirements client. If this value is blank, the name of the object is used as the text. This property is editable.
Mnemonic	Menu Folder, Menu Item	Mnemonic used on the menu or menu item. This property is editable.
Modify And Read Access	Security Profile	List of users who can modify objects to which this security profile applies. This property is editable.
Module	Menu Folder	Sets the corresponding menu for display in any combination of the Systems Engineering and Requirements Management module, the Administration module, and the Search module. This property is editable.
Multiple Choice	Choice Definition	True if this is a multiple-choice property definition. False if this is a single-choice property definition. This property is editable.
Object Template	Type Definition	Determines the data that is exported to Microsoft Word for objects of this type definition. This property is editable.
Open Icon	Type Definition	Graphic identifying currently open objects of this type. This property is editable.
Open Property	Activator, Macro, Text Definition, Where Clause	Determines which property the Architect/Requirements client uses when an Open command is run. For example, when an activator is opened, the Script

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
		property is displayed. When a text property is opened, the Current Value is displayed. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates.
Output Template	Search	Saved view, Word document template, Excel template, or live Visio stencil used to display the output from a Search. This property is editable.
Output Type	Search	Type of output format for a report. Values are Window , Word , Excel , Visio , and None . This property is read-only.
Override Object Template	Type Definition	Object template that overrides the default object template for an object. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates.
Owner	All	LOID number of the containing object. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.
Packages	Project	Packages available in the project. Packages correspond to this property's valid values, which are Change Log and Versions . This property is editable.
Pass Owner to TcL Context	Choice Definition	Determines whether a Picklist activator applied to the choice definition must recognize the ID of the object to which the corresponding choice property is applied. Values are Yes and No . This property is editable.
Path	All	Sequence of LOIDs from the project node to the object. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.
Projects	User	Projects to which this user has access. This property is editable.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
Properties	Type Definition	List of properties that apply to objects of this type. This property is editable.
Property	Reference Link	Name of the property being referenced. This property is read-only.
Read Access	Security Profile	List of users who can only view objects to which this security profile applies. Users with higher permission automatically receive this permission. This property is editable.
ROIN Counter	Requirement Type Definition	Used to assign custom prefixes and formatting to the ROINs of requirement subtypes. This property is editable.
Routing Status	Requirement	Used by Teamcenter Community to track the stages of approval for requirements. This text property must be applied to requirement type definitions by the Architect/Requirements project administrator. This property is editable.
Script	Activator, Macro, Search, Where Clause	Prewritten group of commands. May be written using Tcl, Java or C#. This property is editable.
Script Type	Activator, Macro, Where Clause	Type of script, such as Tcl, Java, or C#. This property is read-only.
Shared State	Search, View, Menu Folder, Menu Item	User visibility of Search module reports, content table views, and custom menus and menu items. Values are the following: Private Visible only to the creator. Not displayed for any other user. Private is the default value for each new report, view, menu folder, and menu item. Private also allows the user to disable the object temporarily, for example, during editing. Pending Marked as a request to be made public by a project administrator. The creator can set the value to Pending . Public

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
		Visible to all users in the project. Any user who has Project Administrator privilege can set the value to Public .
		This property is editable.
Show Non-Effective	Trace links	Setting to make the non-effective object visible in the Links tab for trace links.
Snapshot	Diagram	Image of the diagram. This property is read-only. This is a non-displayed property for internal use.
Snapshot Filename	Diagram	File name of the diagram image. This property is read-only. This is a non-displayed property for internal use.
Style Sheet	Document Template	List of Microsoft Word styles associated with a document template. The style sheet determines the formatting of data that is exported to a Word document based on the document template. This property is editable.
Style Sheet Source	Project	File name of the Word document from which the style sheet was imported. This property is editable. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates.
Time Flag	Date Definition	Specifies whether the date includes the time of day. This property is editable.
Time Format	Date Definition	Valid format for the time of day. This property is editable.
Type	Type Definition	Built-in object type on which this subtype is based. This property is read-only.
Type Name	All	Type of schema object. This property is read-only.
Update Choice List Dynamically	Choice Definition	Used to assign a Picklist activator to populate the choice list automatically. The activator script, written in Tcl, returns a result that updates the choice list dynamically in the Systems Engineering and Requirements Management module. This property is editable.

Table B-1. System-Defined Properties in Administration Module

Property	Applies To	Description
Usage	Activator, Macro, Where Clause	Description or explanation of the object, for example, its purpose or how it is used. This property is editable.
User Access	Project	Access privilege to the selected project for the current user of this session. This property is read-only.
User Access	User	Access privilege to the project for the user selected in the Users folder. This property is editable.
User Access Level	Project	Numeric equivalent of the User Access property. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates.
Word URL	Requirement Type Definition, Note Type Definition	URL for editing the object's content in Microsoft Word. The URL can also be pasted into the Address bar in Microsoft Internet Explorer and edited in the browser. This property is read-only. This is a non-displayed property. The name can be entered in tags for object templates and Excel templates, and also can be used in Search criteria.

Appendix C: Date Style Generator Utility

Architect/Requirements provides OOTB date style sheets for all supported locales to export data in Excel using views. You can extend the support for these styles by adding locale-specific styles (**.properties** files) in a custom folder specified in the **Custom.Folder.Path** Web application configuration parameter. You can use the **Date Style Generator** utility to generate these custom date properties file.

The following are the prerequisites for the utility:

- JDK installed on the user's machine. The **JDK_BIN** environment variable should be set to a valid *JDK/bin* home folder.
- Microsoft Excel 2013 or Excel 2016 to generate the **.properties** file.

To generate a custom date style sheet using the utility:

1. Extract the **DateStyleCreator.zip** file to a temporary folder (for example, **C:\TcSE_DateStyle_Generator**).
2. In the extracted contents, the **templates** folder includes the **DateFormat_Template.xlsx** file, which contains the mapping information between the Architect/Requirements date formats and the Excel date styles.
3. Take a backup of **DateFormat_Template.xlsx** for future use.
4. Set the regional setting of the machine to the language for which you are creating a style.
5. Open **DateFormat_Template.xlsx** in Excel.
6. The values in the first two columns (**TcSE Date Format** and **TcSE Time Format**) represent all possible combinations of Architect/Requirements date and time formats.

The **Display Date** column represents the Excel date format corresponding to Architect/Requirements date and time format. For example, the **Long** date format in Architect/Requirements is represented by applying custom date formatting in Excel as **mmmm d, yyyy**.

For a given locale, modify the formatting on every cell of **Display Date** column, such that the date format matches with the Architect/Requirements date format for the given locale. To apply the date format, right-click a cell and select **Format Cells**. The **Custom** category can be used to format the cell.

Save and close the file after modifying it.

7. Run **dateStyleCreator.bat** from the utility folder. It prompts for the location of the template file. Enter the full path of the **.xlsx** file modified in step 6 (for example, **C:\TcSE_DateStyle_Generator\templates\DateFormat_Template.xlsx**).

8. The utility creates the property file (for example, **DateStyle_en_GB.properties**) in the **templates** folder.
9. Copy this file to the folder specified in the **Custom.Folder.Path** Web application configuration parameter.

Appendix D: Project Package Property

Architect/Requirements provides the following package options for use as the **Package** property of a project:

- **Version**

The majority of product development involves changing and improving an existing product, rather than starting from scratch. As a result, the product structures, requirements, and functions remain essentially the same as in the original design, but change slightly as the product matures. To accommodate this evolutionary process, Architect/Requirements employs versions and variants for requirements and building blocks.

Each version is an independent object, with its own relationships to other objects, including child objects and defining and complying objects.

For information on **Version**, see [Enabling the Versions Package for the Project](#).

- **Change Log**

A change log captures the history of modifications to an object of a given type definition.

For information on **Change Log**, see [Change Logs](#).

- **Diagramming**

Projects are created with three default stencils for the three basic diagram types:

- Ported
- Un-ported
- Static tree

Installing the diagramming package creates additional stencils for some common diagram types like UML, IDEF0, flow chart, ERD, block diagram, and data control flow.

- **Standard Reports**

Architect/Requirements provides built-in reports through the optional Standard Reports package.

For information on **Standard Reports**, see [Reports](#).

- **Synergy**

Installing the Synergy package updates the project's schema to support the Synergy interface.

For details on Synergy interface, see the chapter *Controlling Requirement Content Changes Through Telelogic Synergy* in the *Systems Architect/Requirements Management User's Manual*.

- **Power Users**

Power users are non-administrative users who are assigned with project-specific privileges to assist with the project administration.

For information on **Power Users**, see [Power Users](#).

You must have the **Project Administrator** privilege for the project to modify the Package property of a project.

To modify the Package property of a project:

1. In the navigation tree, select the **Projects** node.
The content table displays all projects to which you have access.
2. Select the project in the content table.
The **Properties** tab or window displays all viewable properties for the project.
3. In the **Value** column, double-click the **Packages** property value.
The Multi-Choice dialog window is displayed.
4. Select the packages that you want to apply and click **OK**.

Appendix E: Controlling Requirement Content Changes Through IBM Synergy

Overview of the Architect/Requirements Interface With Synergy



IBM Synergy product support is provided by IBM.

The Architect/Requirements interface with IBM Synergy™ 6.5 brings together primary capabilities of both applications:

- Create requirements in Architect/Requirements, and enter content in the database using Microsoft Office Word.
- Add requirements to Synergy and use its check in and check out functions, accessible in Architect/Requirements, to manage content changes. A Synergy session is not required for these actions.

A checked-in requirement is released by the check in user and is frozen in Architect/Requirements. It is unfrozen automatically when it is checked out.

A checked-out requirement becomes a new version in Architect/Requirements.

Each application maintains references for synchronization with the corresponding objects in the other application:

- In Architect/Requirements, the **SynergyID** property shows the full path to a requirement in a Synergy project work area.
-

In Synergy, the **TcSE_URL** property shows an object's URL in Architect/Requirements.



Architect/Requirements remains the authoring and editing tool. Do not enter content in Synergy. Otherwise, synchronization problems may occur.

From Synergy, you can open a requirement object and view the content in a browser window. This content automatically includes a hyperlink, named **Goto**, which you can click to navigate to the requirement in Architect/Requirements. The Architect/Requirements client is not required for viewing in Synergy.

The project administrator enables the interface by adding the **Synergy** package, together with the **Version** package, to the Architect/Requirements project in the Administration module.

Synergy Package Additions to Architect/Requirements Project Schema

The **Synergy** package updates the project schema as follows:

-

Adds the following subfolders to the **Activators** folder:

Synergy	Defines the Synergy custom menu in the Systems Engineering and Requirements Management module and contains a menu item for each menu command. This subfolder has the Menu Folder subtype.
Synergy Macros	Contains activators and macros that carry out various Synergy interface functions. This subfolder has the Admin Folder subtype.

For more information about using activators, see the *Systems Architect/Requirements Management API Reference* manual.

-

Adds the following properties to the **Property Definitions** folder:

Synergy Check-In Comment	Plain text information entered the last time the requirement was checked in to Synergy. The default comment is Check In from TcSE .
SynergyID	Full path to a requirement in a Synergy work area.
SynergyTask	Synergy task to which a checked-out requirement is assigned.

The Synergy properties are automatically applied to the **Requirement** type definition in the **Type Definitions** folder. Subsequent new subtypes based on the **Requirement** type definition inherit the Synergy properties. Before existing subtypes can be used with Synergy, these properties must be manually applied to the subtypes, including the **Paragraph** subtype. For more information about modifying the properties of a type definition, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

-

Adds the following templates to the **Reports and Formatting** folder:

Synergy Document Template	<ul style="list-style-type: none">o Applies the formatting specified in the Default Style Sheet to the Synergy object content.o Assigns the Synergy Object Template to the Requirement type definition. The project administrator is responsible for modifying the Synergy Object Template assignment to include any subtypes to be added to Synergy.
----------------------------------	--

Synergy Object Template Contains tags that specify the content that is inserted in the Synergy object. This data includes the **Goto** hyperlink in the Synergy object content.

For more information about templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

Enabling the Synergy Package for an Architect/Requirements Project

The Architect/Requirements **Synergy** package contains the components of the interface with Synergy. The project administrator installs these components by adding the **Synergy** value to the **Packages** property for the project. For more information, see [Synergy Package Additions to Architect/Requirements Project Schema](#), earlier in this chapter.



- The **Synergy** package cannot be removed from the project after it is added.
- Synchronization problems may occur if requirement content is entered directly in Synergy. Use only Architect/Requirements to author and edit the content of requirements that are added to Synergy.



- You must have **Project Administrator** privilege for the project.
- The **Version** package also must be added to the project. The **Version** package cannot be removed from the project after it is added.

1. On the module bar, click the **Administration** button to access the Administration module.
2. Select the project node in the navigation tree or the content table.

The **Properties** tab or window displays all viewable properties for the project.

3.

In the **Value** column, double-click the value for the **Packages** property.

The Multi-Choice dialog window is displayed.

4. Check the **Synergy** check box, and then click **OK** to close the dialog window.



If the **Version** check box is cleared, check it before you click **OK**.



The **Synergy** and **Version** packages cannot be removed if you continue this procedure.

5. Click **Yes**.



This action cannot be reversed.

The **Synergy** value is added to the **Packages** property and the Synergy package is enabled.

Adding Requirements to a Synergy Project

Requirements must be added to a Synergy project before their content can be managed through the check in and check out functions. A Synergy session is not required for this procedure.



- If a folder is selected, only its top-level requirements are added. Its nested folders, if any, are also selected and only their top-level requirements are added. Children of top-level requirements are not added.
- If a selected folder contains objects other than requirements and folders, those objects are not added.
- For parent requirements selected individually, direct children and lower-level descendants are not added.
- Requirement hierarchies in Architect/Requirements are not preserved in Synergy. They reside at the same level in the Synergy directory.
- The Synergy properties must be applied to the type definition of each requirement. For more information, see [Synergy Package Additions to Architect/Requirements Project Schema](#), earlier in this chapter.
- You cannot add requirements that are frozen.

1.

Select the requirements, pull down the **Synergy** menu, and choose **Migrate**.



The Synergy Login dialog window is displayed if you are not logged in to Synergy. Enter your Synergy user ID and password, and then click **OK**.

The Select a Synergy Project dialog window is displayed.

2.

Click the Synergy project, and then click **OK**.

The Select Folder for Files dialog window is displayed.

3. Navigate to and select the Synergy directory where you want to add the requirements, and then click **OK**.

At the bottom of the main window, the status bar displays a progress indicator. Then, an information message states that the migration is complete.

For each selected requirement:

- A check mark appears on the object type indicator, showing that the requirement is checked out to your Architect/Requirements user name.
- The **SynergyID** property shows the full path in the Synergy project.
- The **Synergy Check-In Comment** and **SynergyTask** values are blank.

Procedure Notes

Step 1: You can select individual requirements in the content table or the **Links, Where Used**, or **Versions** tabs or windows. You can select a folder in the navigation tree, the content table, or the **Links** tab or window.

Checking In Requirements to Synergy

A check mark is shown on the object type indicators of requirements that are checked out from Synergy. When changes are complete, check in the requirements to make them available to other users. A Synergy session is not required for this procedure.

You can select a folder the check in its top-level requirements simultaneously. Only the folder's top-level requirements are checked in. Its nested folders, if any, are also selected and only their top-level requirements are checked in. Children of top-level requirements are not checked in.

For parent requirements selected individually, direct children and lower-level descendants are not checked in.



- If a selected folder contains objects other than requirements and folders, those objects are not checked in.
- This procedure freezes the requirements. You cannot check in requirements that are already frozen.

1. Select the requirements, pull down the **Synergy** menu, and choose **Check-In**.



The Synergy Login dialog window is displayed if you are not logged in to Synergy. Enter your Synergy user ID and password, and then click **OK**.

The Check-In dialog window is displayed.

You can enter a comment by double-clicking the **Values** cell to open a text field. A default comment is entered if you leave this field blank.

2. Click **OK** to start the check in.

At the bottom of the main window, the status bar displays a progress indicator during this process. Then, an information message states that the check in is complete.

For each selected requirement:

- A lock symbol replaces the check mark on the object type indicator, showing that the requirement is frozen.
- The **Synergy Check-In Comment** property displays your comment or the default comment.
- The **SynergyTask** property is cleared of any previous value.

Procedure Notes

Step 1: You can select individual requirements in the content table or the **Links, Where Used**, or **Versions** tabs or windows. You can select a folder in the navigation tree, the content table, or the **Links** tab or window.

Checking Out Requirements From Synergy

A lock symbol is shown on the object type indicators of requirements that are checked in to Synergy. The lock symbol shows that the requirements are frozen. Check out the requirements to create a new version in Architect/Requirements and make changes to the content. A Synergy session is not required for this procedure.

You can select a folder to check out its top-level requirements simultaneously. Only top-level requirements are checked out; children are not checked out. Any nested folders are selected and only their top-level requirements are checked out.

For parent requirements selected individually, direct children and lower-level descendants are not checked out.



Use only Architect/Requirements to change the content of requirements that are checked out from Synergy. Do not edit such requirements directly in Synergy. Otherwise, synchronization problems may occur.



- If a selected folder contains objects other than requirements and folders, those objects are not checked out.
- The requirements must be frozen. This procedure unfreezes the requirements automatically.

1. Select the requirements, pull down the **Synergy** menu, and choose **Check-Out**.



The Synergy Login dialog window is displayed if you are not logged in to Synergy. Enter your Synergy user ID and password, and then click **OK**.

The Select a Synergy Task dialog window is displayed.

2. Click the Synergy task to which you want to assign the check out, and then click **OK** to start the check -out.

At the bottom of the main window, the status bar displays a progress indicator. When the check out is complete, an information message is displayed.

For each selected requirement:

- A new version is created in Architect/Requirements.

You can access all of the requirement's versions through the **Versions** tab and window. In the content table, you can use the **Effectivity** list to access versions by category. For more information, see *Unsatisfied xref title* in chapter *Unsatisfied xref number*, *Unsatisfied xref title*; and *Unsatisfied xref title* in chapter *Unsatisfied xref number*, *Unsatisfied xref title*.

- A check mark replaces the lock symbol on the object type indicator, showing that the requirement is unfrozen and checked out.
- The **SynergyTask** property shows the assigned task.

Procedure Notes

Step 1: You can select individual requirements in the content table or the **Links, Where Used**, or **Versions** tabs or windows. You can select a folder in the navigation tree, the content table, or the **Links** tab or window.

Synergy Interface Customization

The Synergy Interface Toolkit provides a mechanism to link Architect/Requirements requirement objects with Synergy CM. This linking allows Architect/Requirements to remain the content authoring tool, while Synergy CM is used as the configuration management tool. It is a toolkit in that it provides the necessary tools to allow the user to customize the Synergy interface and to create totally new applications and interfaces.

When the Synergy package is installed, several menu items are included, plus several activators and macros. Along with the delivered Synergy Interface Java code, these are the tools to be used in the Customization Toolkit. With these tools, custom behavior and custom applications can be developed.

Notable features added to support the Synergy interface are the createAction RunJava mechanism, the client interface ClientJavaAPI.runMacro() mechanism and the ability to customize icons based on a property being set on an object.

Program Flow

A typical flow of events would be to create a requirement in TcSE, then migrate it to Synergy and check it in. After it has been checked in to Synergy, the user must perform a check out operation in order to create a new version of the object in TcSE to make changes.

The Synergy Interface provides the ability to migrate requirements to Synergy, and to perform Check-In and Check-Out operations on the migrated requirements. These behaviors are accomplished by using customizable Tcl code plus customizable external Java code.

Initially, a Tcl macro is invoked on the TcSE client, which in turn executes external Java code, which then in turn calls the runMacro capability in TcSE to perform further TcSE behavior via Tcl.

A Synergy menu item is selected, which executes Tcl code. This code gathers the selected objects, performs necessary verification functions, gathers needed TcSE information, and then passes information along to the external Java code via the createAction RunJava mechanism, which is documented elsewhere.

The external Java code then verifies input parameters, logs into the Synergy application, performs Synergy queries, prompts the user for input, performs Synergy specific actions such as logging in and creating objects, gathers needed information, and then passes information back to TcSE via the ClientJavaAPI's runMacro() method, which runs a macro in TcSE.

The macro then completes the action by using the information passed to set properties, create versions, and set object status in TcSE, and then displays a completion message dialog.

Once the requirement objects are linked to Synergy, then in the interests of synchronization, certain user actions are discouraged via activators. These include deleting the requirement, manually freezing or unfreezing the requirement, manually modifying the Synergy properties and version creation.

Customization Points

The interface is customizable and customizations can be implemented in Java code or Tcl code. The user can also add new menu commands. Changes can be made to the Synergy menu item Tcl code, to the external Java code and to the finishing macros. The behavior can be changed and modified to more exactly match other processes or to change the behavior entirely, or to tweak the behavior a little.

New Applications and Interfaces

The Synergy interface can be used as a starting point for other interfaces. New applications and interfaces can be created following the same general processing.

The Synergy interface provides a number of new integration features that will facilitate interacting with TcSE. These include:

- createAction RunJava
- ClientJavaAPI class
 - runMacro() method
 - getTempDir() method
- Icon overlays

The Synergy interface can be modified, or it can be copied and changed, or it can be used as a learning tool to understand how the interoperability works.

Users can create custom Tcl code and custom Java code that will be run “on the client” in the client JVM. An example of this might be interfacing with other applications or creating custom interfaces, such as creating a Java form for user input. Also, the mechanism is put in place via the createAction RunJava to allow the external program to interface through the C# interface.

createAction RunJava

The createAction RunJava command is covered elsewhere in the documentation. Its purpose is to allow external Java code to be run from within TcSE via Tcl code. It allows identification of Java classes and execution of methods in those classes from the Tcl code of activators, macros or custom menu items. The location of the .jar file must be identified to TcSE via the Package.Location parameter set by the administrator. The method signatures must follow a prescribed format. The .jar file must always contain a method named connectTcSE with a prescribed signature.

Examples of the usage of the createAction RunJava command can be found in the Synergy interface.

ClientJavaAPI Class

The ClientJavaAPI class is the beginning of a client interface.

In order to call a TcSE macro from the external Java code, the class file must import the ClientJavaAPI class, which can be found in the tcrPackages.jar file. So the tcrPackages.jar file can be added to the classpath, or it can be unzipped and the ClientJavaAPI.class file can be pulled out to the development area.

runMacro() Method

The mechanism to call a macro from external Java code uses one of the ClientJavaAPI class’s runMacro() method.

A basic method will be the one that simply calls the macro by name, passing in a space delimited string of LOIDs for the macro to use as selected objects:

```
public String runMacro(String macroName, String objectIDs)
```

Another method uses the macro name and the macro LOID along with the object IDs. In this case, if the macroID is passed and is not null, then the macroName is not used:

```
public String runMacro(String macroName, String macroID,  
String objectIDs)
```

Another well used method is one where values are passed into the macro via the Form mechanism, passing in the form properties and the associated values. Again, if the macroID is passed and is not null, then the macroName is not used:

```
public String runMacro(String macroName, String macroID,  
String objectIDs, String[] formProps, String[] formValues)
```

getTempDir() Method

There is a mechanism to get a temporary directory if needed from TcSE. The getTempDir() method returns a string containing the system temporary directory that is used by TcSE.

ClientJavaAPI Usage

The ClientJavaAPI class must be available to the custom package. Then, the ClientJavaAPI must be imported with this statement:

```
import com.edsplm.tc.req.client.shared.api.ClientJavaAPI;
```

and instantiated with a statement similar to this:

```
protected ClientJavaAPI javaAPI = new ClientJavaAPI();
```

before it can be used like this:

```
result = javaAPI.runMacro(macroName, objectIDs);
```

or this:

```
result = javaAPI.runMacro(macroName, null, macroParameters,  
formProps, formValues);
```

Icon Overlays

The Synergy interface does, and new customizations may, use the hidden **Icon Overlay** property to provides a mechanism for adding custom overlays to object type indicator icons. If the **Icon Overlay** property value is the LOID of a type definition, the icon that is set for that type definition is used as an overlay to the current icon.



The **Icon Overlay** property applies only to folders, requirements, and building blocks and their subtypes. These object types are containers for other types of objects.

The custom overlay is the first one applied, followed by any possible out of the box overlays (such as subtype, variant, frozen, etc.) All out of the box icons are 16x16 pixels in size. In case the icon sizes are not the same, overlay icons are anchored at the top left of the icon.



If the overlay icon has transparent parts, then the original icon shows through. If the overlay icon is opaque, it appears to completely replace the existing icon.

A new **Icon** type definition has been created as a convenience. It is a non-instantiable type definition used solely for creating subtypes of type definitions for using custom icon overlays.

The hidden **Icon Overlay** property can be set via Tcl in a menu item, an activator, or a macro. For example, the following Tcl code sets the custom overlay icon to the icon identified as the icon for a type definition named **BigRedEx**:

```
# get the LOID of the icon overlay
set name "BigRedEx"

# find type def by iterating over all type defs
foreach def [getList $currentProject OBJECT_TYPE_LIST]
    {if {[getValue $def Name] == $name}
        {set iconOverlay $def
         break
        }
    }
}
# real code would ensure that icon was found before continuing
setValue $selected "Icon Overlay" $iconOverlay
```

Any overlay change is always a change on the actual object (involving the object's **Change Time** and **Change User** properties). Also, the change impacts not just the current user, but the overlay is saved in the database and shared with all users in the project.

Java Development Environment

Java knowledge is needed in order to take full advantage of this capability, and will help get the development environment to be set up properly. Different people use different tools and have different ways of doing things, so do what is necessary in order to set up a development environment.

The hard part is setting it all up, extracting the files to the proper location and getting the compiler set up to compile correctly and have it compiled in the right location so that TcSE can find it properly. Once everything is set up correctly, the rest is straightforward.

It is a good idea to create subclasses of the delivered classes, so that any potential changes made will not be overwritten when a new TcSE deployment is made.

These things must be accomplished:

- Set up development environment, preferably with an IDE (Eclipse is free at www.eclipse.org).
- Classpaths must be set up correctly.
- Packages must be set up correctly.
- Must be able to “jar up” the code for distribution.
- Understand how to extend the delivered class in order to create new customized behavior.

The goal is to create a .jar file that can be distributed, or to replace the **tcrPackages.jar** file. A good starting point would be to unzip the **tcrPackages.zip** file, which contains the two external Java files used in the Synergy Interface into the working directory.

Development Hints and Recommendations

New custom java code should be in its own package.

The Java package does not require a main() method, but is simply a collection of methods.

The methods must be called from TcSE in order to be an extension of the TcSE client.

Unzip the **tcrPackages.zip** file and become familiar with the Java source code for the **TcSECMInterface.java** and **SynergyInterface.java**, and how they work together, as well as become familiar with the out of the box Synergy macros. These are the parts that constitute the Synergy interface, and the parts that are the best example of the Customization Toolkit.

SynergySchemaSave

If the Synergy Interface files are being modified for use with a specific Synergy installation, there is a SynergySchemaSave macro delivered with the Synergy package that can be used to repackage the changes for redistribution. If the Synergy package is customized and saved into the tcrPackages.jar file, special care is needed during the next TcSE upgrade in order to preserve the customizations, as they will be lost with a client upgrade.

If a new interface is being created from scratch, then a new macro similar to the SynergySchemaSave macro can be created to package up the schema for redistribution.

Index

A

Access control	
Inheritance	
Overview.....	157
Setting.....	162
Introduction.....	155
Overview.....	155
Access privilege.....	146
Modifying.....	146
Overview.....	133
Access to projects, granting or revoking.....	145
Activators	
Change Approved.....	173
Change Rejection.....	173
Change Response.....	173
Change Submit.....	173
Deleting.....	58
Picklist subtype	
Applying to choice property definition.....	70
Creating.....	69
Description.....	69
Removing from choice property definition.....	70
References to, removing.....	52
Results, Excel template tags.....	108
Script, opening.....	69
Activators property.....	186
Activators, creating, running, and modifying ...See Systems Architect/Requirements Management API Reference	
Adding	
Requirements to Synergy.....	205
Synergy package to project.....	204
Additional Privilege property.....	146, 186
Admin Folder, schema object type.....	22
Administration folders	
Creating.....	22
Deleting.....	25
Modifying.....	23
Moving.....	23
Overview.....	21
Renaming.....	24
Security profiles, applying.....	24
Administrative tools, Web application configuration parameters.....	171
Apply Packing value, Excel Template Rules property....	106
Apply To New Descendents property.....	157, 162, 186
Applying a choice property definition to a type definition	72
Applying a security profile to a schema object.....	164

Approval process, change management package, setting up.....	171
Approved, change approval event.....	167
Approver, change approval user.....	165
Architect license	
Description.....	137
Menu options.....	137
Architect privilege.....	134, 146
Assigning display name	
Menu items.....	27
Menus.....	27
Attachments tab.....	30

B

Browser and dialog window examples.....	11
Building block subtypes	
Change log, setting up.....	175
Creating.....	83
Object templates, assigning.....	84
Object type indicators, customizing.....	86
Overview.....	81
Properties, modifying.....	84
Security profile, setting default.....	163
Security profiles, applying.....	164
TRAM subtype.....	81

C

Change approval	
Events	
Approved.....	167
Rejected.....	167
Response.....	167
Submit.....	165
Setting up.....	171
Users	
Approver.....	165
Notifier.....	166
Change Approved activator:.....	173
Change Approvers property.....	186
Change Log property.....	186
Change logs	
Enabling for project.....	175
Overview.....	174
Setting up for type definitions.....	175
Change Notifiers property.....	186
Change Password dialog window.....	149
Change Rejection activator:.....	173

Change Response activator:	173
Change Submit activator:	173
Change Time property	186
Updates	79
Change User property	187
Updates	79
Changing	
Access privilege	146
Additional privilege	146
E-mail address	146
Maximum privilege level	146
Project access	146
Type definitions, properties	84
User name	146
User passwords	149
Checking in requirements to Synergy	207
Checking out requirements from Synergy	208
Choice List property	187
Choice list, dynamic, choice property definition	69
Choice List, Picklist activator	69, 70
Choice property definition	
Choice list, dynamic, setting	69
Creating	63
Modifying	64
Requiring non-blank value	67
Choice property value	61
ClientJavaAPI class	211
Command line	
Export, running with tcradmin script	43
Import, running with tcradmin script	41
Complying Objects Count property	187
Configuring	
Change management package	
Change approval	165
Change logs	174
Mapping file, Microsoft Office Visio and live Visio interface	116
Connection point sections, live Visio stencil	130
Connection point sections, using	131
Connection subtypes	
Change log, setting up	175
Creating	83
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164
Consumer license	
Description	137
Menu options	137
Content property	187
Content Type property	187
Conventions	
Browser and dialog window examples	11
Names	12
Revisions	11
Values	12
Create Time property	187
Create User dialog window	144
Create User property	187
createAction RunJava	211
Creating	

Folders, Administration module	22
Menu items	27
Menus	26
Projects	34
Property definitions	63
Security profiles	159
Stencils, live Visio interface	125
Style sheets	100
Subtypes	83
Templates	
Document template	92
Excel template objects	103
Object template	97
Type definitions	83
User groups	150
Users	144
Current Access property	187
Current Value property	188
Customization, IBM Synergy interface	
ClientJavaAPI class	211
createAction RunJava	211
Customization points	210
getTempDir() method	212
Icon overlay	212
Icon overlays	213
Java development environment	214
New applications and interfaces	211
Overview	210
Recommendations	214
runMacro() method	211
SynergySchemaSave	215
Customizing	
Menus	26
Object type indicators	86
ROIN, requirement type definition	88
Stencils, live Visio interface	126

D

Data ElementTypes	
Deleting Objects	51
Property Values	50
Required Fields	50
Type specific fields	51
Database Class Name property	188
Date Format property	188
Date property definition	
Creating	63
Modifying	73
Date property value	61
Date Style Support	
Export to Microsoft Office Excel	115
Deactivating a user	153
Default Object Template	96
Default Text Object Template	96
Default View property	188
Default view, setting for folder type definitions	87
Defining Objects Count property	188
Deleting	
Folders, Administration module	25
Objects, schema	58
Deleting a project	60

Description property	188
Diagram subtypes	
Change log, setting up.....	175
Creating.....	83
Object type indicators, customizing	86
Overview.....	81
Properties, modifying.....	84
Security profile, setting default	163
Security profiles, applying	164
Dialog Window examples.....	11
Dialog windows	
Change Password.....	149
Create User	144
Document Template.....	94
Modify Choice Property Definition	64
Modify Date Property Definition	73
Modify Numeric Property Definition.....	74
Modify Text Property Definition	78
Multi-Choice.....	204
Additional Privilege property value, modifying	147
Applying property definition to type definition.....	72
Change Approvers property value, modifying	173
Change Log property value, modifying.....	176
Change Log, enabling for project.....	175
Change Notifiers property value, modifying	174
Power Users, enabling for project	135
Projects property value, modifying	148
Properties tab	30
Schema objects, removing references	52
Security profile, modifying	160
Standard Reports, adding to project	18
Type definition, modifying	84
User, project access, granting or revoking	145
Versions, enabling for project	169
Open.....	36, 39, 86, 125, 126, 127
Rename	65
Save	38
Select	125, 126
Select a Synergy Project	205
Select folder for files.....	205
Select Location to Copy	126
Single-Choice	
Document template, modifying.....	94
Input Required property value, modifying	67
Maximum Privilege property value, modifying	147
Properties tab	30
Schema objects, removing references	52
Security profile, Apply To New Descendents property	162
Security profile, applying, administration folders	24
Security profile, applying, schema objects.....	164
Security profile, applying, type definitions	163
Text property definition, modifying	76
Updating choice list dynamically.....	70
User Access property value, modifying	147
User, deactivating.....	153
Synergy Login.....	205, 207, 208
Document subtype, folders	81
Document Template dialog window	94
Document Template Rules property	188
Document templates	
Creating.....	92

Deleting.....	58
Modifying.....	93
Overview	91
References to, removing.....	52
Dynamic choice list, choice property definition.....	69

E

Editable properties	185
Editing	
Property definitions	
Choice definition	64
Date definition	73
Numeric definition.....	74
Text definition	76
Security profiles	160, 162, 164
Stencils, live Visio interface.....	126
Style sheets.....	101
Templates	
Document template.....	93
Excel template, modification concepts	107
Excel template, packing multiple objects in one row	106
Excel template, procedure steps.....	104
Object template.....	97
Type definitions, applying security profiles	164
Type definitions, properties.....	84
Effectivity Date property.....	188
Effectivity Label property	188
Effectivity Type property	189
E-mail Address property	146
E-mail address, user	144
Email property.....	189
E-mail server, IP address.....	171
Emptying a user's Recycle Bin.....	154
Enabling	
Architect/Requirements interface with IBM Synergy 204	
Change logs, for project	175
Promote References, for project	170
Versions, for project	169
Enterprise Administrator privilege	133
Error String property	189
Error String property, Excel templates	114
Events property	189
Excel Template Rules property	189
Excel Template Rules property, Apply Packing value ...	106
Excel templates	
Activator tag results.....	108
Creating template objects	103
Error String property	114
Modifying	
Concepts	107
Procedure steps.....	104
Overview	103
Packing data	112
Report output, Search module	103, 110
Rule table	105
Concepts	110
Conditions	114
Criteria matching	114
Data placement in export files	111
Level key field.....	110

Levels and relationships	111
Packing multiple objects in one row	106
Relationship key field	110
Subtype key field	110
Export or Import Objects	
Rule File Management	45
Exporting data from project	37

F

Filtering	
Menus by privilege	28
Menus by Security Profile	29
Filtering the Properties tab	32
First Name property	146, 189
Floating tab windows	31
Folder subtypes	
Change log, setting up	175
Creating	83
Document subtype	81
Object templates, assigning	84
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164
View, setting default	87
Folders, Administration module	
Creating	22
Deleting	25
Modifying	23
Moving	23
Overview	21
Renaming	24
Security profiles, applying	24
Form Values property	189
Format property	189
Formula property	189
Full Control property	160, 190

G

getTempDir() method	212
Group subtypes	
Change log, setting up	175
Creating	83
Object templates, assigning	84
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164

H

Has Trace Links property	190
HTML property	190

I

IBM Synergy	
-------------	--

Macros	202
Synergy Document Template	202
Synergy Object Template	202
Viewing requirement content	201
IBM Synergy interface	
Activators	202
Adding requirements	205
Architect/Requirements interface, overview	201
Architect/Requirements project schema, Synergy package	202
Checking in requirements	207
Checking out requirements	208
ClientJavaAPI class	211
ClientJavaAPI usage	212
createAction RunJava	
createAction RunJava	211
Customization	
Customization points	210
New applications and interfaces	211
Overview	210
Program flow	210
Customization, recommendations	214
Enabling Architect/Requirements Synergy package for project	204
getTempDir() method	212
Icon overlays	213
Java development environment	214
Property definitions	202
runMacro() method	211
SynergySchemaSave	215
TcSE_URL property	201
Icon overlay	212
Icon overlays	213
Icon property	190
Importing	
Microsoft Office Word styles to style sheet	101
Project	35
Schema objects to project	39
Inheritance, access control	
Overview	157
Setting	162
Input Required property	67, 190
Instance Security Profile property	190
Property definitions	158
Type definitions	163
IP address, E-mail server	171

J

Java, development environment, IBM Synergy interface	214
--	-----

L

Last Name property	146, 190
Launcher URL property	190
Layers, port master shapes	130
License types	
Architect	
Description	137
Menu options	137
Consumer	
Description	137

Menu options	137
Menu options	137
Requirements	
Description	137
Menu options	137
Script Authoring	
Description	137
LightWeight Properties property	190
Live Visio interface	
Mapping file, configuring	116
Static tree stencil	124
Stencils, creating	125
Stencils, editing	126
Live Visio stencils	
Connection point sections	130
Layers, port master shapes	130
Ports, master shapes	130

M

Macros, creating, running, and modifying	See Systems Architect/Requirements Management API Reference
Macros, deleting	58
Macros, Visual Basic, in Excel templates	108
Mapping file, configuring, Microsoft Office Visio and live Visio interface	116
Master shapes, ports, layers	130
Master shapes, ports, live Visio stencil	130
Maximum privilege level	
Assigning	144
Modifying	146
Overview	133
Maximum Privilege Level property	191
Maximum Privilege property	146, 191
Menu items	
Assigning display name	27
Creating	27
Rearranging	29
Setting module display	29
Setting visibility	28
Menu options, Systems Engineering and Requirements Management module	
License types	137
Shortcut keys	137
Menu Text property	191
Menus	
Assigning display name	27
Creating	26
Customizing	26
Filtering	28, 29
Rearranging	29
Setting module display	29
Setting visibility	28
MessageQueue.Start parameter	84
MHTML property	191
Microsoft Office Excel	
Excel templates	
Creating	103
Modifying, concepts	107
Modifying, procedure steps	104
Overview	103
Packing multiple objects in one row	106

Microsoft Office Visio	
Mapping file, configuring	116
Static tree stencil	124
Stencils, creating, live Visio interface	125
Stencils, editing, live Visio interface	126
Microsoft Office Word	
Object templates	
Creating	97
Modifying	97
Overview	96
Style sheets	
Creating	100
Modifying	101
Overview	100
Styles, importing to style sheet	101
Mnemonic property	191
Modify And Read Access property	160, 191
Modify Choice Property Definition dialog window	64
Modify Date Property Definition dialog window	73
Modify Numeric Property Definition dialog window	74
Modify Text Property Definition dialog window	78
Modifying	
Folders, Administration module	23
Property definitions	
Choice definition	64
Date definition	73
Numeric definition	74
Text definition	76
Security profiles	160, 162, 164
Stencils, live Visio interface	126
Style sheets	101
Templates	
Document template	93
Excel template, modification concepts	107
Excel template, packing multiple objects in one row	106
Excel template, procedure steps	104
Object template	97
Type definitions, properties	84
User groups	151
User permissions	160
User profiles	146
Module property	191
Moving	
Folders, Administration module	23
Schema objects	23
Multi-Choice dialog window	204
Additional Privilege property value, modifying	147
Applying property definition to type definition	72
Change Approvers property value, modifying	173
Change Log property value, modifying	176
Change Log, enabling for project	175
Change Notifiers property value, modifying	174
Power Users, enabling for project	135
Projects property value, modifying	148
Properties tab	30
Schema objects	
Removing references	52
Security profile, modifying	160
Standard Reports, adding to project	18
Type definition, modifying	84
User, project access, granting or revoking	145

Versions, enabling for project	169
Multiple Choice property	191

N

Name conventions	12
Name property	146
No Access privilege	134
Note subtypes	
Change log, setting up	175
Creating	83
Object templates, assigning	84
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164
Notebook pane	
Attachments tab	30
Floating tab windows	31
Preview tab	31
Properties tab	30
Where Used tab	31
Notifier, change approval user	166
Numeric property definition	
Creating	63
Modifying	74
Numeric property value	61

O

Object Template property	191
Object templates	
Assigning to type definitions	84
Creating	97
Modifying	97
Overview	96
References to, removing	52
Object type indicators, customizing	86
Objects	
Deleting	58
Exporting	37
Folder type definitions	
Default view, setting	87
Importing	39
Requirement type definitions	
ROINs, customizing	88
Subtypes	
Creating	83
Object templates, assigning	84
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164
System-defined properties	185
Open dialog window	36, 39, 86, 125, 126, 127
Open Icon property	191
Open Property property	191
Output Template property	192
Output Type property	192
Overlay, object icon	212

Overlays, object icons	213
Override Object Template property	192
Owner property	192

P

Packages	
Synergy package, adding to Architect/Requirements project	204
Synergy package, project schema	202
Packages property	192
Packing data, Excel export templates	106, 112
Packing multiple objects in one row	106
Paragraph subtype, requirements	81
Pass Owner to TeL Context property	192
Passwords, resetting	149
Path property	192
Pending, Shared State property value	
Reports	18
Views	19
Picklist, activator subtype	
Applying to choice property definition	70
Choice List activator	69, 70
Creating	69
Description	69
Removing from choice property definition	70
User activator	69, 70
User Group activator	69, 70
Port master shapes, live Visio stencil	130
Power User privilege	134
Power Users	
PU_Activators_Access	136
PU_Property_Definitions_Access	136
PU_Reports_and_Formatting_Access	136
PU_Security_Profiles_Access	136
PU_Type_Definitions_Access	136
PU_User_Access	136
Preview tab	31
Primary folders, Administration module	21
Private, Shared State property value	
Reports	18
Views	19
Program flow, IBM Synergy interface customization	210
Project Administrator privilege	133
Projects	
Adding Synergy package	204
Creating	34
Deleting	60
Enabling change logs	175
Enabling Promote References	170
Enabling versions	169
Importing	35
Schema, Synergy package	202
User access, granting or revoking	145
Projects property	146, 192
Promote References, enabling for project	170
Properties property	193
Properties tab	
Description	30
Filtering properties	32
Properties, system-defined	185
Property definitions	

Choice, applying to type definition	72
Creating	63
Deleting	58
Instance Security Profile property	158
Modifying	
Choice definition	64
Date definition	73
Numeric definition	74
Text definition	76
Overview	61
References to, removing	52
System-defined properties	185
Property does not exist, Microsoft Office Excel export files	114
Property property	193
Public, Shared State property value	
Reports	18
Views	19

R

Read Access property	160, 193
Read and Write privilege	134
Read Only privilege	134
Read-only properties	185
Rearranging	
Menu items	29
Menus	29
Recycle Bin, emptying for a user	154
Reference links	
Displaying as hyperlink URLs, Preview tab and window	95
Promote References	170
Properties, removing from type definitions	84
Reference Settings	170
Registering users	144
Rejected, change approval event	167
Removing	
Picklist activator from choice property definition	70
References to a schema object	52
Rename dialog window	65
Renaming, folders, Administration module	24
Reports	
Deleting	58
Orphan Requirements	17
Output, Search module	110
Overview	17
Requirement Approval Status	17
Rule table, Excel templates	
Concepts	110
Modifying	105
Show Impact Downstream	17
Show Impact Upstream	17
Trace Report Deep	18
Trace Report Deep Table Format	18
Unallocated Requirements	18
Requirement Object Identification Number (ROIN)	See ROIN
Requirement subtypes	
Change log, setting up	175
Creating	83
Object templates, assigning	84

Object type indicators, customizing	86
Overview	81
Paragraph subtype	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying	164
Requirement type definitions	
ROINs, customizing	88
Requirements	
Adding to Synergy	205
Checking in to Synergy	207
Checking out from Synergy	208
Requirements license	
Description	137
Menu options	137
Resetting a user password	149
Response, change approval event	167
Revision conventions	11
ROIN Counter property	193
ROIN, customizing	88
Routing Status property	193
Rule File Management	
Default AP233 Rule File	45
Export or Import Objects	45
Identifying the Rule for Objects	45
Identifying the Rule for Properties of Objects	46
Including and Excluding Property	47
Specifying the Unique Property for Updating Objects	
During Import	48
Updating Objects During Import	47
Using the Rule File	45
Rule table, Excel templates	
Concepts	110
Conditions	114
Criteria matching	114
Data placement in export files	111
Key fields	
Level	110
Relationship	110
Subtype	110
Levels and relationships	111
Modifying	105
Packing multiple objects in one row	106
runMacro() method	211

S

Save dialog window	38
Schema objects	
Creating	
Document templates	92
Excel templates	103
Object templates	97
Property definitions	63
Security profiles	159
Style sheets	100
Type definitions	83
User groups	150
Users	144
Deleting	58
Exporting	37
Importing	35, 39

Moving.....	23	Security profile	
Overview.....	16	Apply To New Descendents property.....	162
References to, removing	52	Applying, administration folders	24
Security profile, setting default for type definitions...	163	Applying, schema objects	164
Security profiles, applying	164	Applying, type definitions	163
Schema, Architect/Requirements projects		Text property definition, modifying	76
Synergy package	202	Updating choice list dynamically	70
Script Authoring license		User Access property value, modifying.....	147
Description.....	137	User, deactivating.....	153
Script Authoring privilege	134, 146	Snapshot Filename property	194
Script property	193	Snapshot property.....	194
Script Type property.....	193	Spreadsheet subtypes	
Script, activator, opening	69	Change log, setting up	175
Search module, Excel template for report output....	103, 110	Creating	83
Security profiles		Object type indicators, customizing	86
Applying		Overview	81
Folders, Administration module.....	24	Properties, modifying	84
Objects, Systems Engineering and Requirements		Security profile, setting default	163
Management module	167	Security profiles, applying.....	164
Schema objects, Administration module	164	Standard reports	
Creating.....	159	Adding to project.....	18
Default, setting for type definitions.....	163	Descriptions.....	17
Deleting.....	58	Static tree stencil, live Visio interface	124
Inheritance		Stencil object	
Overview.....	157	Uses Port property	127
Setting	162	Stencils, live Visio	
Introduction.....	155	Connection point sections.....	130
Overview.....	155	Port master shapes	130
References to, removing	52	Port master shapes, layers.....	130
Setting user permissions.....	160	Stencils, live Visio interface	
System-defined properties.....	185	Creating	125
Select a Synergy Project dialog window.....	205	Editing	126
Select dialog window.....	125, 126	Style Sheet property	194
Select folder for files, dialog window	205	Style Sheet Source property	194
Select Location to Copy dialog window	126	Style sheets	
Setting		Creating	100
Default view, folder type definitions.....	87	Deleting	58
Dynamic choice list, choice property definition.....	69	Modifying.....	101
Module display		Overview	100
Menu items.....	29	References to, removing.....	52
Menus.....	29	Styles, Microsoft Office Word	
User permissions, security profiles	160	Importing to style sheet	101
Visibility		Modifying in style sheet	101
Menu items.....	28	Submit, change approval event.....	165
Menus.....	28	Subtypes	
Setting up		Applying choice property definition to type definition	72
Change approval process	171	Change log, setting up.....	175
Change logs, for type definitions	175	Creating	83
Shared State property.....	193	Folders, setting default view.....	87
Menu option visibility	28	Object templates, assigning	84
Reports.....	18	Object type indicators, customizing	86
Views	19	Overview	81
Shortcut keys, menu options, Systems Engineering and		Picklist, activator subtype	
Requirements Management module.....	137	Applying to choice property definition.....	70
Show Non-Effective property	194	Creating	69
Single-Choice dialog window		Description	69
Document template, modifying.....	94	Removing from choice property definition.....	70
Input Required property value, modifying	67	Properties, modifying	84
Maximum Privilege property value, modifying	147	ROINs, customizing	88
Properties tab	30	Security profile, setting default	163
Schema objects		Security profiles, applying.....	164
Removing references.....	52	Synergy Check-In Comment property definition	202

Synergy Document Template	202
Synergy folder, Administration module.....	202
Synergy Login dialog window	205, 207, 208
Synergy Macros folder, Administration module.....	202
Synergy Object Template	203
Synergy Task property definition	202
SynergyID property definition	202
SynergySchemaSave.....	215
System-defined properties	185

T

Tabs	
Attachments	30
Floating windows.....	31
Preview	31
Properties	30
Where Used	31
Tags	
Activator results, Excel template	108
Excel template.....	104
Object template	99
Tags, object template	96, 97
TBD button, Modify Date Property Definition dialog window	73
tcradmin script	
Data export.....	43
Data import	41
TcSE XML Format	49
Data Element Types	50
Order of Objects.....	52
Schema XML Format	52
XML File Types.....	49
TcSE_URL property, interface with IBM Synergy.....	201
Templates	
Document templates	
Creating	92
Modifying	93
Overview.....	91
Excel templates	
Creating	103
Modifying, concepts.....	107
Modifying, procedure steps.....	104
Overview.....	103
Rule table	106
Object templates	
Assigning to type definitions.....	84
Creating	97
Modifying	97
Overview.....	96
Text property definition	
Creating.....	63
Modifying	76
Text property value.....	61
Time Flag property	194
Time Format property	194
Today button, Modify Date Property Definition dialog window	73
Trace link subtypes	
Change log, setting up.....	175
Creating.....	83
Object templates, assigning.....	84

Object type indicators, customizing	86
Overview	81
Properties, modifying	84
Security profile, setting default	163
Security profiles, applying.....	164
Tracking changes	
Change approval.....	165
Change logs	174
TRAM subtype, building blocks	81
Transitional mapping.. See TRAM subtype, building blocks	
Tree view, live Visio interface	124
Type definitions	
Change log, setting up	175
Creating	83
Deleting	58
Folders, setting default view.....	87
Object templates, assigning	84
Object type indicators, customizing	86
Overview	81
Properties, modifying	84
ROINs, customizing	88
Security profile, setting default	163
Security profiles, applying.....	164
System-defined properties	185
Type Name property.....	194
Type property.....	194

U

UNC path format.....	76
Update Choice List Dynamically property	70, 194
URL, text property definition.....	76
Usage property	195
User Access Level property	
Project	195
User Access property	146
Project	195
User	195
User Group, Picklist activator	69, 70
User groups	
Creating	150
Modifying.....	151
Object type indicators, customizing	86
References to, removing.....	52
Subtypes	
Creating	83
Overview	81
Properties, modifying	84
Security profile, setting default.....	163
Security profiles, applying.....	164
User, Picklist activator	69, 70
Users	
Access privilege	
Modifying	146
Overview	133
Access to projects, granting or revoking	145
Changing	
Access privilege.....	146
Additional privilege.....	146
E-mail address	146
Maximum privilege level.....	146
Project access.....	146

Creating.....	144
Deactivating.....	153
E-mail addresses	144, 146
Emptying Recycle Bin	154
Licensing.....	137
Maximum privilege level	
Modifying	146
Overview.....	133
Overview.....	133
Permissions, modifying.....	160
References to, removing	52
Resetting passwords.....	149
System-defined properties.....	185
User name, modifying.....	146
Uses Port property, stencil object	127
Using connection points.....	131

V

Value conventions	12
Values of system-defined properties.....	185
Versions, enabling for project.....	169
Views	
Default, setting for folder subtypes	87
Overview.....	19
References to, removing	52

Visio.....	See Microsoft Office Visio
Visual Basic macros, in Excel templates.....	108

W

Web application configuration parameters, administrative tools.....	171
Where Used tab	31
Word	See Microsoft Office Word
Word URL property	195

X

XML File Types	
Data	49
Project	50
Schema	49
XML files	
Exporting data	37
Importing projects	35
Importing schema objects.....	39
Mapping file, configuring, Microsoft Office Visio and live Visio interface.....	116

Teamcenter 11.1 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management System Administrator's Manual

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	3.0.1	May 2003
B	4.0	December 2003
C	4.1	February 2004
D	5.0	July 2004
E	6.0	March 2005
Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
F	2005	September 2005
G	2005 SR1	June 2006
H	2007	December 2006
I	2007.1	April 2007
J	2007.2	September 2007
K	2007.3	January 2008
L	8	January 2009
M	8.0.1	June 2009
N	8.1	October 2009
O	8.2	October 2010
P	9	July 2011
Q	9.1	May 2012
Q1	9.1.5	January 2014
R	10.0	January 2015

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	10.1	September 2016
T	11.1	March 2018

This edition obsoletes all previous editions.

System Administrator's Manual Contents

Manual History	4
Contents	7
Preface.....	9
Audience	9
Conventions	9
Revision Marks	9
Names and Values	9
Command Line Entries, File Contents, and Code.....	10
Submitting Comments.....	10
Proprietary and Restricted Rights Notice.....	11
Chapter 1: Introduction to System Administration.....	13
Systems Architect/Requirements Management Architecture	14
Systems Architect/Requirements Management Components	16
Client Components	17
Web Components.....	17
Database Server Component.....	18
Chapter 2: Configuring Architect/Requirements	19
Installing Architect/Requirements Client in Silent Mode	19
Using the Configuration Web Page	20
Configuring the Office Live Interface and Teamcenter Linking	22
Configuring License Information.....	24
Customizing the Architect/Requirements Server.....	25
Configuring Security Services	31
Configuring for Performance Improvement in WAN Environments.....	32
Configuring Password and Login Settings.....	32
Setting Other Configuration Options	34
Configuring Log Server Parameters	34
Naming Convention.....	36
Naming Process	36
Managing Message Queues	36
Viewing items in the message queue	37
Stopping the message queue	37
Removing tasks.....	37
Using Log Files for Troubleshooting.....	37

Chapter 3: Tuning the Database.....	39
Setting Versant Database Parameters.....	40
System File	40
Log Files	40
profile.be File.....	40
Running the Versant reorgdb Utility.....	42
Chapter 4: Maintaining the System	43
Cleaning up a Project	43
Maintaining the Database.....	43
Backing Up the Database.....	44
Restoring the Database	45
Backing Up and Restoring the osc-dbid File	45
Increasing the Size of the Database	46
Running the Database Maintenance Utility	47
Running the database maintenance utility in delta mode.....	55
Stopping the Database	57
Starting the Database	57
Deleting the Database	57
Cleaning Up a Project in the Database	57
Running System Utilities	59
Chapter 5: Managing Licenses	61
Architect/Requirements License Key Files.....	61
Viewing License Information	62
Accessing the License Management Utility.....	63
Adding a License Key	64
Removing License Keys	65
Troubleshooting Server Errors.....	66
Appendix A: List of Class Names for Checking Database Objects	67
Appendix B: Modifying the Client Memory Allocation	69
Appendix C: Configuring Security Services Logging.....	71
Appendix D: Reverting to Security Services 11.2 or Previous Versions	74
Index.....	75

Preface

This manual is a system administrator's reference for Teamcenter Systems Architect/Requirements Management 10.1. Systems Architect/Requirements Management belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual is for Systems Architect/Requirements Management system administrators. The manual provides both conceptual information and step-by-step instructions for specific tasks.

This manual assumes that you are familiar with the general system administration tasks of operating systems.

Conventions

This manual uses the conventions described in the following sections:

Revision Marks

Technical changes are marked by a bar adjacent to the changed text.

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **initsid.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
<i>Italic</i>	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In initsid.ora , <i>sid</i> identifies a varying portion of the name (a unique system ID). For example, the name of the file might be:

	initBlue5.ora
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Command Line Entries, File Contents, and Code

This manual represents command line input and output, the contents of system files, and computer code in fonts that help you understand how to enter text or to interpret displayed text. For example, the following line represents a command entry:

```
msqlora -u system/system-password
```

The conventions are:

Monospace	<p>Monospace font represents text or numbers you enter on a command line, the computer's response, the contents of system files, and computer code.</p> <p>Capitalization and spacing are shown exactly as you must enter the characters or as the computer displays the characters.</p>
<i>Italic</i>	<p>Italic font represents text or numbers that vary. The words in italic text describe the entry.</p> <p>The words are shown in lowercase letters, but the varying text may include uppercase letters. When entering text, use the case required by the system.</p> <p>For the preceding example, you might substitute the following for <i>system-password</i>:</p> <pre>KLH3b</pre>
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

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

© 2018 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Chapter 1: Introduction to System Administration

This chapter presents an overview of system administration in Systems Architect/Requirements Management.

A Systems Architect/Requirements Management system administrator has the following main responsibilities:

- Configuring Systems Architect/Requirements Management in a Web application server environment.
- Tuning the Systems Architect/Requirements Management database.
- Maintaining the Systems Architect/Requirements Management database in the back end.
- Managing the Systems Architect/Requirements Management license.

For information about Systems Architect/Requirements Management project administration, such as creating projects, maintaining project security, customizing objects and properties, and managing users, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

For information about installing Systems Architect/Requirements Management, see the *Systems Architect/Requirements Management Server Installation Manual*.

Systems Architect/Requirements Management Architecture

Designed for multitier Web deployment, the Systems Architect/Requirements Management architecture employs four logical tiers:

- The *client* tier, which presents the Systems Architect/Requirements Management user interface on a local computer.
- The *Web* tier, which handles the HTTP/HTTPS traffic between the client and enterprise tiers.
- The *enterprise* tier, where the Systems Architect/Requirements Management business logic resides.
- The *resource* tier, where the Versant object database resides.

Although the Web and enterprise tiers are logically separate, both tiers run in the same Java™ environment of the J2EE Application Server, or Web application server with servlet engine. Therefore, only three independent Systems Architect/Requirements Management components are deployed:

- Client workstation (where the client runs). For example, a Windows XP computer.
- Web application server (Requires servlet engine only. Does not require or use EJB support.) For example, WebLogic, Tomcat.
- Database server (Versant database application).

In enterprise production deployments, the Web application server and database server typically run on different hosts. In smaller installations, both components can run on the same host. For standalone demonstration environments, all three components can be installed on one workstation or laptop computer.

The components in all tiers are multithreaded and are able to maximize performance on hardware with multiple CPUs.

Figure 1-1 shows two examples of multitier Web deployment.

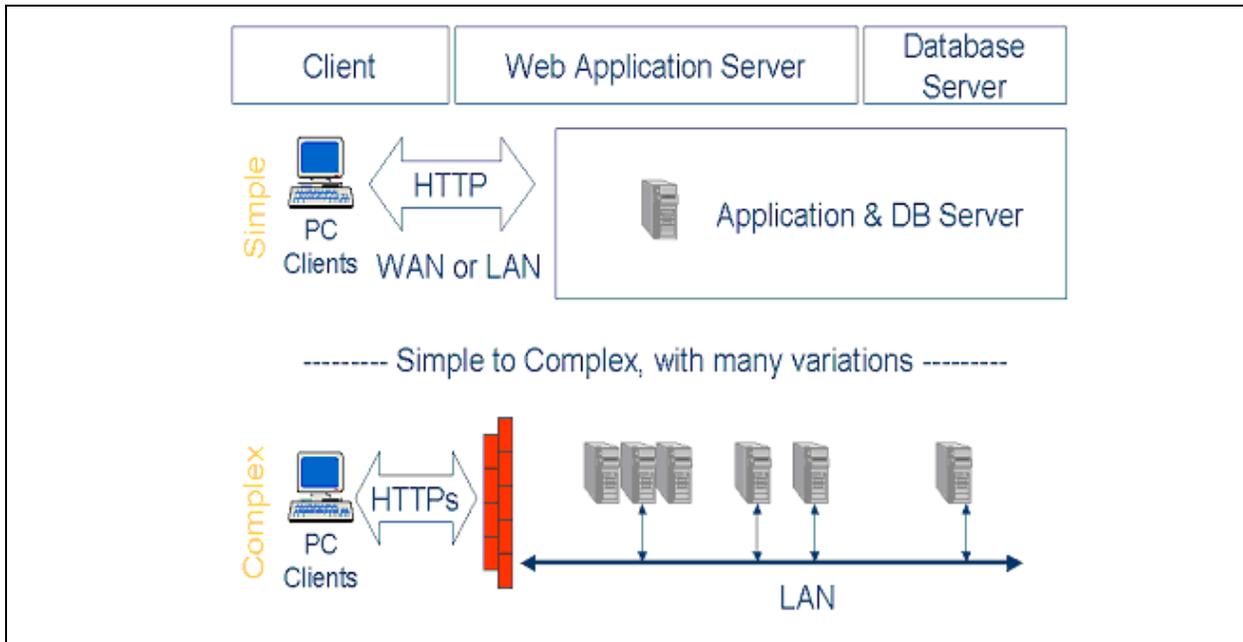


Figure 1-1. Systems Architect/Requirements Management Multitier Web Deployment

Systems Architect/Requirements Management Components

The Web application server and database server components are initially installed from the Systems Architect/Requirements Management installation package. The files installed on the Web application server include the client components. These client files are automatically downloaded to a user's workstation the first time the user logs in to Systems Architect/Requirements Management.

The components installed from the installation package fall into three categories:

- Client components, which are unique to Systems Architect/Requirements Management.
- Web components, which are common to other Teamcenter products.
- Database component, the Versant database server supplied by Siemens PLM Software.

Figure 1-2 shows the components included in the Systems Architect/Requirements Management installation package.

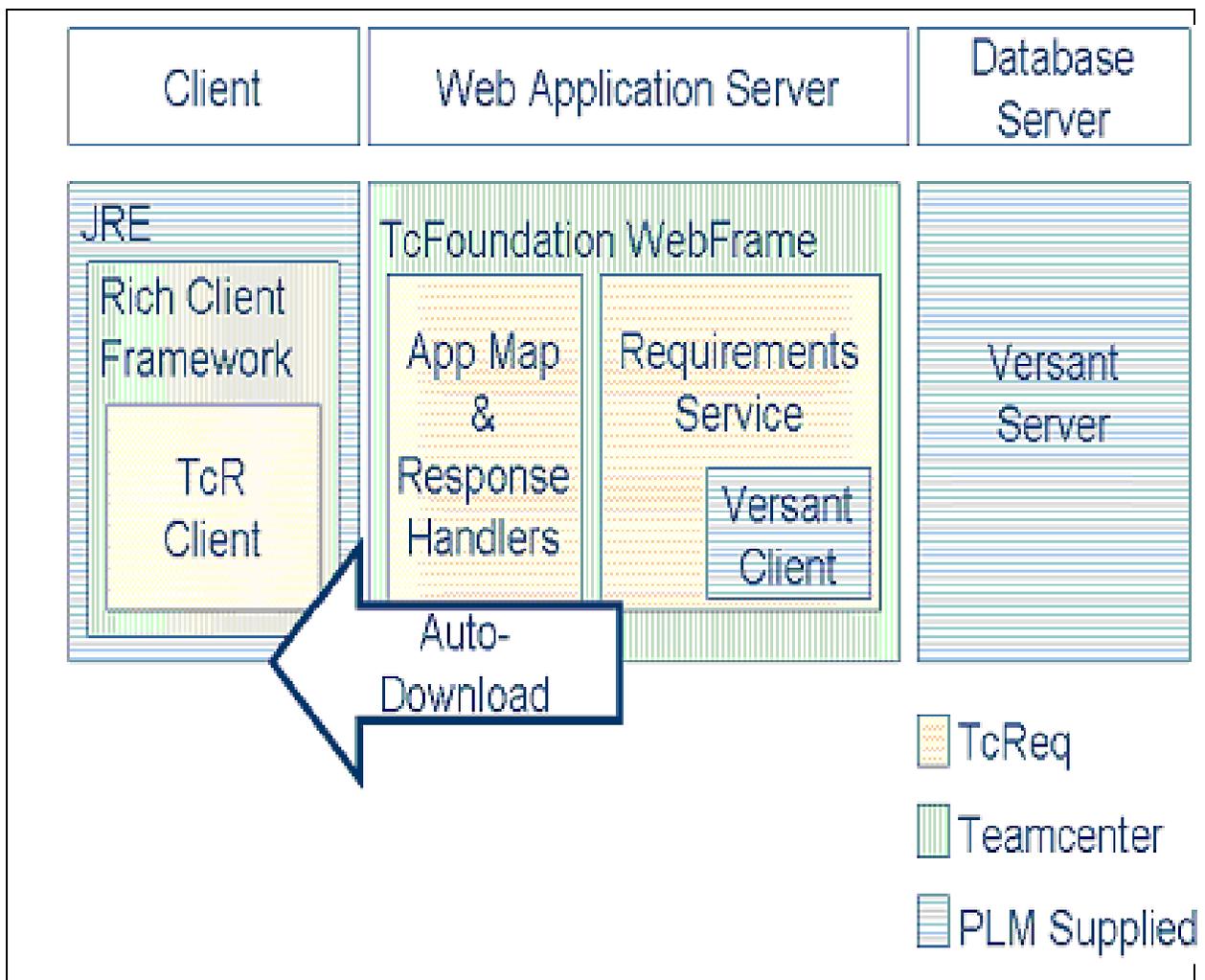


Figure 1-2. Components Included in the Systems Architect/Requirements Management Installation Package

Client Components

The Systems Architect/Requirements Management installation package contains the following client components:

-

Systems Architect/Requirements Management client

This component presents the Systems Architect/Requirements Management user interface on a local computer, and communicates with the Web tier via HTTP/HTTPS. This Java client is downloaded automatically from the Web application server when a user logs in to Systems Architect/Requirements Management the first time. It is also downloaded whenever a user logs in and a newer version is found at the server.

-

Rich Client Framework

The Rich Client Framework (RCF) is shared by certain other Teamcenter products. This Java client authoring framework and the Systems Architect/Requirements Management client are installed and run together as one application. At this time, there is no need for users or administrators to deal separately with RCF, but in the future more than one Teamcenter product may run within the same RCF instance and certain RCF customization points will be exposed.

-

Java Runtime Environment (JRE)

The Systems Architect/Requirements Management client requires a Java Runtime Environment. If a compatible version is already present on the client workstation, that existing JRE installation is used. Otherwise, the JRE version delivered with Systems Architect/Requirements Management is automatically downloaded to the client and installed.

Web Components

The Systems Architect/Requirements Management installation package contains the following Web components:

-

TcFoundation WebFrame

This is a model-view-controller Java environment that manages incoming HTTP/HTTPS requests and dispatches them according to the *application map*. *TcFoundation WebFrame* is also used by Teamcenter Enterprise, but at this time Systems Architect/Requirements Management installs and runs its own copy.

-

Application map and response handlers

The application map tells TcFoundation WebFrame which *response handlers* to execute for each incoming URL. The response handlers then communicate the incoming user request to the requirements service and format the results for return to the client via the HTTP/HTTPS response.

-

Requirements service

The Systems Architect/Requirements Management business logic resides in the *requirements service*. Incoming requests are verified against access rights of the user, actions are performed on objects residing in the database, and results assembled for return to the client. Each request to the requirements service is bounded by a database transaction that is rolled back if any part of the request fails to complete successfully.



Versant client

The requirements service is a client to the Versant database, managed through a pool of open database sessions. Each incoming request from a user is temporarily allocated one of those sessions to perform its work, and then releases the session. This pool avoids the overhead of establishing a new database connection for each user request, and avoids the database server load of maintaining a dedicated open connection for each user.

Database Server Component

The Versant database server is supplied with Systems Architect/Requirements Management. The database server can be installed on the same host as the Web application server, or another host within the same local area network segment. In large installations, more than one Web application server can connect to the same database server.



For best performance, avoid firewalls or routers between the database server and the Web application servers connected to it. Wide-area network segments between the database server and the Web application servers must never be used.

Chapter 2: Configuring Architect/Requirements

This chapter contains instructions for configuring Architect/Requirements in a Web server environment.

Configuring Architect/Requirements involves the following:

- Configuring for linking to other Teamcenter products
- Configuring for license
- Customizing the Architect/Requirements server
- Configuring for Security Services

Use the Configuration Web page to configure Architect/Requirements parameters. The remaining sections in this chapter provide details on the specific parameters to update.

Installing Architect/Requirements Client in Silent Mode

The Architect/Requirements client installation program provides an option to install the client in silent mode. The IT departments or enterprise administrators can use the silent mode option to roll out command line installations of Architect/Requirements Client with Office Integration on end-user machines.



- The installation must be performed as a user in the Power Users or Administrators group.
- The client installer does not check for the user privilege level. If the installation is not performed as a privileged user, the registry entries are not created and the client fails to run when launched by a normal user.

To install the Architect/Requirements client in silent mode:

1. The client installation program is located in the Architect/Requirements Web component (the **tcr.war** file).

The **tcr.war** file is located in the **war_file** directory. The typical location of the **war_file** folder is *C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file*.

Extract the **tcr.war** file in a folder. You can use Winzip, or run the command: `jar -xf tcr.war`

2. Locate the client installer (**setup.exe**) in the **/ugs/tc/req/installs** folder and run the following command:

```
setup.exe -i silent
```

The client is installed in the default location.

For example, in the **C:\Program Files\SiemensPLM\Teamcenter\SystemEngineering\Release_9** folder.

Using the Configuration Web Page

A Configuration Web page provides a graphical user interface for you to configure server parameters. This section describes how to access the Configuration Web page and explains how to use it to configure Architect/Requirements parameters.

To access the Configuration Web page:

1.

In Internet Explorer, open the Architect/Requirements home page, and then click the **Administrative Tools** link.

Depending on whether Security Services is enabled on the Architect/Requirements server, one of two login pages appears:

- The Architect/Requirements login page appears if Security Services is not enabled.
- The Teamcenter Login page appears if Security Services is enabled.

2. Do one of the following:

- On the Architect/Requirements login page, enter your Architect/Requirements user name and password, select a language, and click **Log In**.
- On the Teamcenter Login page, enter your Security Services user name and password, select a language, and click **Log In**.

3. From the Administrative Tools page, select **Web Application Configuration**.

This brings you to the following URL, which can also be accessed directly:

```
http://.../tcr/ugs/tc/req/configtcr.jsp
```

For example:

```
http://myhost:8080/tcr/ugs/tc/req/configtcr.jsp
```

The Architect/Requirements Configuration Web page is displayed.

To enter information in the Configuration Web page:

1. Under **Parameter Name**, select the parameter you want to update.

If a parameter value exists in the database, it appears in bold in this column.

The **Description** column provides a description of the corresponding parameter.

2. Under **Current Database Value**, enter the proposed value for the parameter.

If a parameter value exists in the database, that value appears in this column.

3. After you make all your updates, you have the following choices:



For information about installing the Architect/Requirements security files, see the *Systems Architect/Requirements Management Server Installation Manual*.

- To reset the value of a parameter that has been changed to its default value, before clicking the **Update** button, click **Clear**.
The default value appears in the **Default Value** column.
- To reset the value of selected parameters to their default values, click **Reset to Default**.
- To update your changes and save them to the database, click **Update**.
- To cancel your changes, click **Back**.

4. Click **Update**.

The Confirm Deployment Descriptor Update page is displayed, with the proposed changes in a table.

5. From the Confirm Deployment Descriptor Update page, click **OK** to accept the changes and update the database.

If you need to change the information, click **Back** to return to the Configuration Web page.

6. Restart your Web server after modifying a value.

Configuring the Office Live Interface and Teamcenter Linking

To support Office Live interface configuration and object linking between Architect/Requirements and other Teamcenter products, modify the following parameters using the Web Application Configuration page (<http://.../tcr/ugs/tc/req/configtcr.jsp>):



If there is a problem, run the Office Live diagnostic utility, which is located at **Systems Architect/Requirements Management Home Page|Administrative Tools|Diagnostic Tools|Live Office diagnosis**.

WOLF.AppIP (Office Live Interface and Teamcenter Linking)

The application address, used when registering with the application registry service.

Change to the DNS name or IP address of the server where the Architect/Requirements Web application is running. For example:

```
server1.ugs.com
```

WOLF.AppPort (Office Live Interface and Teamcenter Linking)

The application port, used when registering with the application registry service.

Change to the port where the Architect/Requirements Web application is running. For example:

```
8080
```

WOLF.AppProtocol (Office Live Interface and Teamcenter Linking)

The Architect/Requirements Web application protocol, used when registering with the application registry service.

The value is **http** unless you are running Architect/Requirements under Security Services, in which case the value may be **https**.

WOLF.AppRegistryURL (Teamcenter Linking only)

The URL of the application registry service.

WOLF.AppGUID (Teamcenter Linking only)

The Architect/Requirements unique global unique identifier (GUID) for the Architect/Requirements database being registered. Enter a new value. The GUID must be unique.

WOLF.ChooserURL (Teamcenter Linking)

URL for the WOLF object chooser. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.IconURL (Teamcenter Linking)

URL for an icon that represents the Architect/Requirements application. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.LauncherURL (Teamcenter Linking)

URL that identifies the Architect/Requirements Object Launcher. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.LinkHandlerURL (Teamcenter Linking)

URL to transfers the control to Architect/Requirements link handler during WOLF linking. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

WOLF.SoapURL (Teamcenter Linking)

Web Service URL for SOAP communication in PROXY linking. If omitted, this is constructed using the WOLF AppIP, AppPort and AppProtocol, which yields the correct value for most sites.

Configuring License Information

When you register your Architect/Requirements application, you receive the following from Siemens PLM Software Customer Support:

- Your Architect/Requirements license file, named **tcr.lic**.
- Your Architect/Requirements customer number.

To configure a license from the Configuration Web page:

- Locate the **LIC. Customer Number** parameter and enter your customer number in the corresponding **Current Database Value** box. You are given a customer number with your Architect/Requirements license.

For more information, see [Managing Licenses](#).

Customizing the Architect/Requirements Server

You can customize the Architect/Requirements server by modifying the following parameters on the Web Application Configuration page (<http://.../tcr/ugs/tc/req/configtcr.jsp>):

DB.LockTimeOut

The length of time (in minutes) during which reservations on requirement objects are held after a timeout occurs before being released. The first user to open a requirement for editing obtains a reservation, or lock, on that requirement. No other user can change the requirement until the reservation is released. The existing reservation is renewed each time the user saves his or her work, extending the time the object is held.

The reservation is released when the user closes the requirement or when the HTTP session is terminated.



If your HTTP session terminates (for example, if the client session times out) your reservation is lost. Another user can open the requirement for editing during this time. If you then try to edit the requirement, the client session reestablishes a connection with the database and sets a new reservation if another user has not opened the requirement for editing.

If the Web server is not aware that your HTTP session has terminated (for example, if the Web server crashes), the reservation on the requirement is released at the end of the lock timeout period, allowing a user to access the requirement.



If your system is configured so that multiple Web servers access a single database, the lock timeout period is checked as soon as a second user (on a different Web server from the first user) tries to access the requirement. This can cause the first user to lose his or her reservation. For such a system configuration, Siemens PLM Software recommends that you set a higher lock timeout period (for example, 60 minutes) to avoid having multiple users accessing the requirement.

You can change the reservation duration in the **Changed to Value** field for this parameter.

DB.MaxSession

The maximum connections to the database. You must restart the Web server after changing this parameter.

DB.MaxSessionAge and DB.MaxSessionSize

Architect/Requirements uses a session pool to manage Versant (database) sessions. These sessions are used by Versant for any database operations. Sometimes, the sessions may grow large and continue to hold the resources, such as memory, even after they are released to the session pool. In such cases, it is better to close the sessions to release the resources they hold to the system. Closing Versant sessions has no effect on the active user sessions.

Two Web Application Configuration parameters enable you to control how and when these sessions are closed. The parameters are:

- **DB.MaxSessionAge** specifies the maximum time in hours that a Versant database session may exist before it is closed. The default value for this parameter is 4. Any unused Versant session is closed by the system if it is found to be more than 4 hours old. You should avoid using a value less than 20 because it may lead to premature session closure and extra overhead for starting new sessions. If you do configure a value less than 20, then monitor session closures in the Architect/Requirements server log. A value of 0 (zero) turns off this behavior, and the session is not closed, irrespective of how old it gets.
- **DB.MaxSessionSize** specifies the maximum size in mega bytes (MB) that a Versant database session may grow to before it is closed. The default value for this parameter is 20. Any unused Versant session that has grown to be more than 20 MB is closed. A value of 0 (zero) turns off this behavior, and the session is not closed irrespective of how big it grows.

The following general formula is suggested for adjusting the **DB.MaxSessionSize** parameter for your site. Divide the total cache size by the maximum number of sessions, with a reservation for especially large actions.

Let us assume:

C is the memory configured for the Versant front end object cache. This is controlled by the Versant tuning parameter **swap_threshold**. The default value is 128 MB, which is sufficient for basic installations. Higher values may be recommended for high user counts and 64-bit servers. **swap_threshold** can be modified in the Versant front end profile.

R is the reserved memory for large transactions (use 100 MB for this value).

N is the number of Versant sessions (the likely maximum concurrent server load, typically 10 to 20% of the number of logged in users).

Then, **DB.MaxSessionSize** = (C minus R) divided by N

For example, let the Versant sessions have 500 MB to work with, and let the server typically need 20 concurrent sessions. This example uses 100 MB for the reserved memory.

Then, **DB.MaxSessionSize** = (500-100)/20 = 20 MB (which is the default setting).

If your calculation yields a value significantly less than 20 MB, you may be at risk of poor performance by expecting to service too many active users with insufficient memory.

DB.UndoLevels

The number of undo levels.

DB.RecycleBinTimeout

Determines whether a user's recycle bin is automatically emptied when the user logs out. This parameter is also used to specify how long a deleted item can remain in the recycle bin before being automatically deleted. Set this value to **0** to enable the automatic emptying of the recycle bin. Set this value to **-1** to disable it. Setting this value to a positive integer indicates the number of days after which the items in the recycle bin are automatically deleted.

DB.FastNumberRefresh

Determines the speed of the operations that modify a hierarchy. **DB.FastNumberRefresh** can be set to **True** or **False**. Operations that modify a hierarchy are faster if this parameter is set to **True**. In some cases, the number property displayed for an object may not be updated immediately. The number properties are always updated on setting this parameter to **False** but the operation takes longer.

Project.Cleanup.Now

Determines whether the database space should be recovered upon deleting a project in the Architect/Requirements client. Note that after a project is deleted, it is immediately removed from the list of projects.

By default, **Project.Cleanup.Now** is false. If **Project.Cleanup.Now** is true, after a project is deleted, the database space is cleaned up immediately by spawning an external process on the Application Server. If **Project.Cleanup.Now** is false, the project cleanup does not happen immediately. A subsequent administrative **projectCleanup** action is required to destroy the project completely, or upon the next complete run of **maintainDB**.

ExternalImportExport

When **True**, data imports and exports in the Architect/Requirements client run in separate Java processes outside the Web application server's JVM. Each import and export request from the client generates a process with its own JVM.

Imports and exports of large files can consume significant amounts of memory in the Web server. Running separate processes can help to avoid performance problems and failures due to out-of-memory errors.



External processes consume the Web server machine's memory. Set the **ExternalImportExport** parameter to **True** only if the server machine has sufficient memory for importing and exporting large files.



Not all imports and exports use a separate process when the value is **True**. Because the overhead can be significant, the server may determine that an import or export is too small to benefit from running in a separate process. For example, when importing a file smaller than two megabytes, an external process is not used.

Each external process automatically runs the **tradmin** script to perform the import or export. The **ImportExportScript** configuration parameter controls the location of the **tradmin** script file.

When **false**, the default value, imports and exports run within the Web server.

ImportExportDir

Complete path to the directory on the server where Architect/Requirements temporarily stores import and export files. The directory must exist and be writable by the Web server process.



If this parameter is not set, document export and import, XML export and import, and some features of the live Office interface do not work.

Some temporary files remain in the directory. The application server deletes temporary files created by Architect/Requirements that are more than two hours old. Architect/Requirements temporary files have the **TcR-** prefix.

ImportExportScript

Complete path to the **tradmin** script file. The script file is created automatically by the Architect/Requirements installer. The **ImportExportScript** parameter is initialized to the file's location.



If the **tradmin** script file is moved or you want to use a customized version, you must update the **ImportExportScript** parameter to point to the new location.

By editing the **tradmin** script, you can configure Java command line arguments such as *maximum heap size (-Xmx)*.

Custom.Folder.Path

Complete path to the directory on the server where user wants to place files related to Architect/Requirements configuration, such as locale-specific date styles. The directory must exist and be writable by the Web server process. The default value of this is blank.

A permanent location can be used for **Custom.Folder.Path** because its value is stored in the database and does not change during an upgrade.

MailServerIP

This is the IP address of the Mail server that is used to send E-mail. If this IP address is not set, send E-mail functionality does not work. Additionally, the change approval feature does not work.

MessageQueue.Master

Determines if this application server Instance is the master for processing objects in the message queue. The message queue must be set as **Master** on one server to integrate with Engineering.

Process Management. Setting this to **Master** also enables background processing, which results in improved performance when adding or removing properties for a type definition.

For more information, see *Modifying the Properties of a Type Definition* in the *Systems Architect/Requirements Management Project Administrator's Manual*.



Property updates may take a few seconds to many minutes, depending on the number of objects of that type, and how often the background queue is set to run. Also, the objects are collected for update using a query, and not all objects in a folder are processed at the same time. So, you may open a folder and see some objects that have the property changes applied and some objects where the changes are not applied yet.

Only one server (instance) should be designated as **Master**. The Master server runs a thread to process all tasks submitted to the message queue.

The possible values are **True**, **False**, or **port** number.

- **True**: Indicates that the machine is a single application server instance and is the master server for managing the message queue.
- **port**: Indicates that the machine is also the master server and has multiple instances running. The port designates the master instance.
- **False**: Indicates that message queue thread not run on this machine.

MessageQueue.Start

Determines if an application server instance participates in message queuing or not.

If background task processing is enabled (see **MessageQueue.Master**) all application server instances (including **MessageQueue.Master**) must have **MessageQueue.Start** set to **True**.

Setting **MessageQueue.Start** to **True** allows the application server instance to submit tasks to the message queue.

MessageQueue.Thread.SleepTime

The Message Queue server checks the queue at interval specified in this parameter.

PortRangeStart

Start of the port range the client searches when trying to find an available port on which to start its socket service.

PortRangeEnd

End of the port range the client searches when trying to find an available port on which to start its socket service.

In most installations the **PortRangeEnd** can be ~5 greater than the **PortRangeStart** value. But, when Citrix or Terminal Server are in use, the range should be 2x the expected maximum number of concurrent users running Architect/Requirements on that shared Microsoft Windows environment.

EnableAdditionalSecurity

If **True**, Architect/Requirements validates all JSP parameters against cross site scripting attack.

Monitor.PollTime

Amount of time (in seconds) that the client waits before it polls the server for a monitored task. 0 indicates no polling.

RegeditEnabled

Administrator should set this parameter to **True** if users have access to read/write registry entries using **regedit.exe**. Architect/Requirements then gets and sets registry entries using **regedit.exe**.

The administrator must set this parameter to **false** if users do not have access to read/write registry entries using **regedit.exe**. Architect/Requirements does not use **regedit.exe** to get and set the registry. Instead it, it uses a third party utility, **ICE_JNIRegistry.dll**.

The default is **True**.



If you are using Internet Explorer as your browser, and if **RegeditEnabled** is set to **false**, and you want to perform multiple installations, you must start a new session of Internet Explorer for each installation. This is because while loading **ICE_JNIRegistry.dll**, a security exception is thrown by Internet Explorer when same browser window is used for second installation.

DB.greadSize

Integer threshold value for when to use a group read while reading a list of objects from the database. If a list is larger than the **greadSize** a group read is used. When reading lists of objects, such as the members of a folder, group reads can improve performance by reducing the number database operations (chattiness). However, group reads hurt performance when reading small lists. The threshold list size where group reads begin to improve performance depends on factors such as network latency between the database and application server. This value does not normally need to be adjusted.

WOLF.Remote.Trace.Deep

A boolean value that indicates if trace reports for integrations show a single level or all levels of linked objects. Setting **WOLF.Remote.Trace.Deep** to **True** will display all levels of trace links for a trace report request from a remote application.

Configuring Security Services

You can customize Architect/Requirements to allow Security Services by modifying the following parameters using the Configuration Web page. Security Services allows users of multiple Teamcenter applications to use a single login window to enable access to all Teamcenter modules.



Security Services 11.3 and higher includes a logging feature that may be useful for support and troubleshooting. For information about the feature, see Appendix C, [Configuring Security Services Logging](#).



It is possible to use Security Services 11.2, or previous versions, by updating the Architect/Requirements war file prior to deployment. For instructions on that procedure, see Appendix D, [Reverting to Security Services 11.2 or Previous Versions](#).

SSO.AppID

The ID that the Security Services server uses to identify Architect/Requirements.

SSO.Enabled

If this value is **True**, Architect/Requirements redirects login requests to the Security Services login URL. Users must have a Security Services user id and password.

SSO.LoginURL

The URL of the Security Services logon service.

SSO.ServiceURL

The URL of the Security Services identity service.

Configuring for Performance Improvement in WAN Environments

The Architect/Requirements system administrator can use certain configuration parameters to significantly improve performance for deployments in WAN environments. The http response from the Architect/Requirements server to the client is compressed using Java's API for zip.

Click the **Web Application Configuration** link on the Administrative Tools page to display the configuration parameters.

Filter.compression Threshold

Number of bytes in response above which the compression of response is enabled. The default value is **128**.

Filter.debug

Debug level for the compression filter. Zero is the minimum. Zero (**0**) is the minimum and is recommended for production.

Filter.doCompression

Indicates whether the http responses are compressed using GZIP compression. If all users are LAN-connected, there may not be any net gain to using compression, but it is a big benefit for WAN users. If compression is provided by application servers, Siemens PLM Software recommends that you set the **Filter.doCompression** parameter to **false** to avoid redundant layers of compression. The default value is **True**.

Configuring Password and Login Settings

You can customize user password settings to determine password length, password character requirements, and password expiration. The default value for each password parameter is 0, meaning the parameter is not enforced. If a user violates a password parameter, an error message appears. You can also set the number of failed logins before a user is suspended.

The password expiration and grace period rules do not apply to enterprise administrators, including the default **tradm** account. If they did apply to these users, there would be a risk that all administrators could be denied access.

Password.Minimum Alpha Chars

The minimum number of required alpha characters in a password.

Password.Minimum Length

The minimum length for a password.

Password.Minimum Numeric Chars

The minimum number of required numeric characters in a password.

Password.Expiration Days

The number of days before a password expires. When this number is reached, users are given a warning message, instructing them to change the password.

Password.Grace Days

When a password is set or reset by the system administrator, the user has this number of days to create a new password. If the user fails to create a new password within the time allowed, he or she loses access to the system.

Password.Grace Expired Logins

Once a password expires, the user has this number of *logins* before losing access to the system. Each time a user logs in after password expiration, a warning message appears telling the user that he or she is about to lose access to the system.

Password.Account Lockout Threshold

The number of login attempts that are allowed before a user is suspended. Zero (0) indicates that the rule is not configured. Once the system administrator resets the password, the user must change it with the grace period defined by **Password.Grace Days**.

Setting Other Configuration Options

You can customize other Architect/Requirements options by modifying parameters using the Configuration Web page (<http://.../tcr/ugs/tc/req/configtcr.jsp>).

JRE.Version

This option allows the Administrator to supply a list of client side Java Run Time Environments (JRE). The list contains all “supported” JREs for the current release of Architect/Requirements.

The option list is configurable since new JRE versions released after Architect/Requirements/Requirements Management may be found to be compatible, and can be added to this list to enable their use. But, do not add JRE versions to this list unless instructed to do so by UGS Customer Support. Adding JREs that are not known to be compatible can lead to unpredictable client errors.

When you log in, Architect/Requirements looks at this list (contained in the database) and makes sure you have one of the JRE’s that are on the list installed on your computer. If not, you are asked if you would like to install a compatible JRE.

The **JRE.Version** configuration option’s default value includes only the 1.6 version. However, sites upgrading from previous Architect/Requirements releases should edit their **JRE.Version** option to have reference to the correct JRE version. Select the **Reset to Default** check box and click **Update**. Add the required JRE version to the list.

For information about version of Java Runtime Environment (JRE) and Java Plug-in supported, see the Siemens PLM Software Certification Database:

http://support.ugs.com/online_library/certification/

Package.Location

The path on the client machine to the optional **.jar** files containing methods that can be run via **createAction RunJava** to be included in the client's **classpath**. This parameter value must include the complete path.

If there is more than one jar file, the paths must be delimited by semicolons. For example:

```
C:\apps\sample1.jar;C:\apps\sample2.jar
```

The path name can accept environment variables in the **%ENVVAR%** notation.

For more information about **createAction RunJava**, see the *Systems Architect/Requirements Management API Reference* manual.

Configuring Log Server Parameters

You can customize the Architect/Requirements log server parameters by modifying the following parameters:

Log.Server.Level

Logging level for Architect/Requirements application server log file. This determines the types of messages written to the log file. Following types of messages can be written to the log file:

- **INFO:** It is the default mode. Setting **Log.Server.Level** to **INFO** helps record informational and warning messages.
- **WARN:** Setting **Log.Server.Level** to **WARN** helps record warning messages only.
- **DEBUG:** Setting **Log.Server.Level** to **DEBUG** helps record all messages including the debug messages.

The log file size increases rapidly when **DEBUG** mode is set, hence, the **DEBUG** option must be used for short durations to diagnose specific problems.

Log.Server.Location

The directory where Architect/Requirements application server log file is saved.



You must restart the application server after updating **Log.Server.Location**.

Log.Server.MaxBackupIndex

The maximum number of backups of the Architect/Requirements application server log files that are maintained. When a new log file is created, a copy of the old log file is made. This setting determines the number of old log files to be preserved. After the maximum is reached, the oldest log file is deleted when a new log file is created. Log files are created when the application server is restarted or when the current log file reaches the maximum size as specified in

Log.Server.MaxFileSize. Backup log files have a numerical extension added to give them a unique name.



You must restart the application server after updating **Log.Server.MaxBackupIndex**.

Log.Server.MaxFileSize

Maximum size (in MB) of the Architect/Requirements application server log file. When a log file reaches the maximum size, a backup of the log file is made and a new file is created.



You must restart the application server after updating **Log.Server.MaxFileSize**.

Log.Server.MemoryMonitorInterval

Time interval, in seconds, that the application server memory usage is logged to Architect/Requirements log file. The information is useful for investigating problems with excessive memory use.

Log.Server.Tcl.Level

Logging level for Tcl script logging in the Architect/Requirements application server log file. The setting behaves the same as `Log.Server.Level` except that it applies when Tcl scripts are running. This can help with debugging Tcl problems by recording only Tcl related debug messages rather than all debug messages.

Naming Convention

A `TcrServerLog_<port>.html` file is created in the location specified in the `Log.Server.Location` parameter. Replace `<port>` with the port number on which the application server instance is running. The naming convention is to support multiple instances/profiles of application server running on the same machine and saving the logs in the same location as specified in the `Log.Server.Location` parameter. The `TcrServerLog_<port>.html` file has the following columns:

- **Time:** Time when the event being logged occurred. It is the time when the log statement in the code is executed.
- **Thread:** Name of the thread that executed the log statement in the code.
- **UserName:** Name of the user logged on when the log was created. The log event created for the logged on user.
- **SessionName:** Name of the Versant session used when the log event occurred.
- **className:** Name of the class from where the event being logged occurred. This information is helpful for the debugging process.
- **Level:** Level at which the event was logged.
- **Message:** Message that was logged.

Naming Process

When the log file reaches the size specified in the `Log.Server.MaxFileSize` parameter then it is renamed to `TcrServerLog_<port>.html.1` and a new `TcrServerLog_<port>.html` is created. All subsequent messages are logged to the new file. The rollover behavior happens as follows:

1. `TcrServerLog_<port>.html.<N>` gets deleted or rolled off the list.
Here `N = Log.Server.MaxBackupIndex`.
2. `TcrServerLog_<port>.html.<N-1>` gets renamed to `TcrServerLog_<port>.html.<N>`.
3. `TcrServerLog_<port>.html.1` gets renamed to `TcrServerLog_<port>.html.2`.
4. `TcrServerLog_<port>.html` gets renamed to `TcrServerLog_<port>.html.1`.
5. A new `TcrServerLog_<port>.html` is created.

Managing Message Queues

Architect/Requirements can perform some operations in the background. These operations include emptying the recycle bin, adding or removing properties from a type definition, and synchronizing information for Teamcenter Engineering and Teamcenter Enterprise integrations. When an operation is done in the background, a task object is created in the database and is added to the message queue. Tasks in the queue are processed in the order in which they are received. When a task operation completes successfully, the task is removed from the queue. If an error occurs, the task can be re-queued and run again later.

Viewing items in the message queue

You can see the pending tasks in the message queue from the **Message Store Administration** page. To view the **Message Store Administration** page, click **Administrative Tools** → **Configure Message Queuing** on the main Architect/Requirements page. A list of tasks waiting to be run is displayed.

Stopping the message queue

You can stop the message queue temporarily from the **Configure Message Queuing** page by clicking the **STOP** button. You can restart the queue processing by clicking the **START** button.

Removing tasks

An error condition can prevent a task from completing successfully. These tasks can remain in the queue. You can remove such tasks from the message queue.



You can delete tasks only when the queue is stopped.

To delete tasks from message queue:

1. Stop the queue.
2. Select the check box for the tasks that you want to delete and click **Delete Selection**.
3. Restart the queue.

Using Log Files for Troubleshooting

You can use Systems Architect/Requirements Management log files for debugging purposes.

To view the location of the Client log file:

1. Select **Tools** → **System Information**.
2. Click **Client Log**.

The client log tab displays the path for the log file. You can navigate to the log path from Windows Explorer. The following log files are present in the log path location:

- **tcr_log.txt**

This is the only file with up to date client log information that you must use for debugging.

- **tcr_log_old.txt**

This is a backup of the log file. You can use this file for debugging a problem that happened earlier than the first item shown in **tcr_log.txt**.

- **temp_tcr_log.txt**

This is a snapshot of the log file created when you click the **Open In Notepad** button. This file is not kept up to date and you must not use it for collecting client log information.

Chapter 3: Tuning the Database

This chapter discusses how the Systems Architect/Requirements Management system administrator tunes the Versant database.

Tuning the Systems Architect/Requirements Management database involves the following:

- Setting Versant database parameters
- Running the Versant reorgdb Utility

Setting Versant Database Parameters

The Versant database can be tuned by changing the database parameters. All databases on a machine are created under the database root directory, which is created when Versant is installed.

Each Versant database is a collection of files located in a directory with the same name as the name of the database. For example, if your Systems Architect/Requirements Management database, **TCR_db**, is on a machine whose database root directory is **/files/versant/db**, that root directory contains a directory named **TCR_db**.

To find the exact location of the database root directory, enter the following command in a command prompt:

```
oscp -d
```

The following sections describe the files contained in each database directory.

System File

The database system volume, **system**, stores class descriptions and object instances for each database. There is one system volume for each database. However, additional volumes can be added through the **addvol** utility.



This file is never edited or manipulated directly. It is documented here only as general information.

Log Files

The physical log volume, **physical.log**, and the logical log volume, **logical.log**, record transaction activities and provide information for rollback and recovery.



These files are never edited or manipulated directly. They are documented here only as general information.

profile.be File

This is the Versant server process profile file. When a database starts, the database server process reads the **profile.be** file to determine the location of the database volumes and to set the database operating environment. If this file does not exist, you cannot start the database.



Changes to **profile.be** do not take effect until the database is restarted. For more information, see [Stopping the Database](#) in chapter 4, *Maintaining the System*.

These are some of the **profile.be** parameters that you can modify:

transaction

Specifies the number of concurrent transactions for this database. The default value is **100**. For example:

```
transaction 100
```

You must increase the **transaction** parameter value if:

- Several hundred users use the Systems Architect/Requirements Management database concurrently.
- You see the following error message:

```
DB_TOO_MANY_USERS: (Too many users have logged into the database.)
```

For more information, contact Systems Architect/Requirements Management Customer Support.

lock_wait_timeout

Specifies the number of seconds the server waits for an object lock to be released before giving up. The default value is **5**. For example:

```
lock_wait_timeout 5
```

max_page_buffs

Specifies the maximum number of 16 KB buffers for caching disk pages from data volumes. This parameter strongly influences database performance. The default value is **2048**. For example:

```
max_page_buffs 2048
```

The default value should work well for most applications. For a large database, however, you may be able to improve performance by increasing the value. To conserve memory for small databases, you may want to decrease the value.

Running the Versant reorgdb Utility

The purpose of the **reorgdb** utility is to pack data so that space is used more efficiently. It reorganizes all data volumes for the database dbname by rearranging objects to remove gaps in physical space. The sequence of steps involved in reorganizing a database is:

1. Set the database to single-connection mode. For example:

```
dbinfo -l TCR_db
```

Available options are:

- `Dbinfo -m TCR_db`: This sets the database in multi user mode.
- `Dbinfo -p TCR_db`: This prints a message that the database is in multi-user mode. It is just for visual confirmation that the database is in multi-user mode.

2. Stop the Versant database using stopdb. For example:

```
stopdb TCR_db
```

3. Run the **reorgdb** *[options] dbname* utility. For example:

```
reorgdb TCR_db
```

The available option is noprint, which suppress display messages while running.

4. Place the database back into Multi-User mode:

```
dbinfo -m TCR_db
```

5. Restart the Versant database using startdb.

```
startdb TCR_db
```

Chapter 4: Maintaining the System

This chapter describes the Systems Architect/Requirements Management system administrator's responsibilities in regular database and system maintenance, with instructions for specific tasks.

Cleaning up a Project

In previous releases, project deletion was performed in a single, long running operation, thus blocking the use of the Architect/Requirements client during the entire transaction. Now, the project deletion has been broken down into two phases.

In the first phase, a project is selected in the Architect/Requirements client and deleted. At this time, all the users are removed from the project, all the external references to the project are removed, and all the project's objects are removed from the Recycle Bins of all users. The cleanup operation can still take some time; however, after the cleanup, the project is completely inaccessible to the users. Control is now returned to the client. Users can continue working in Architect/Requirements.

All the objects in the project are now orphaned and take up space in the database. These objects do not interfere with any other Architect/Requirements transactions.

In the second phase, the project is cleaned up in the database.

For more information on cleaning up a project in the database, see [Cleaning Up a Project in the Database](#).

Maintaining the Database

The Systems Architect/Requirements Management system administrator is responsible for regular database maintenance. This maintenance includes the following operations:

- Backing up the database
- Restoring the database
- Backing up and restoring the **osc-dbid** file
- Increasing the size of the database
- Running the **Database Maintenance** utility
- Stopping the database
- Starting the database
- Deleting the database

Backing Up the Database

You should have a backup strategy in place to protect the mission-critical information assets in your Systems Architect/Requirements Management database. Backups should already be a part of your organization's standard disaster recovery plans. Systems Architect/Requirements Management must be added to these plans to ensure the capability to recover from a failure.

Generic operating system utilities for backing up file systems or disk images are not adequate for protecting the Systems Architect/Requirements Management database. When the database is running, some of its information is in memory. Therefore, that information is not captured by disk backup utilities, whether Systems Architect/Requirements Management clients are connected to the database or not.

The only way to ensure a completely recoverable backup is to use the Versant **vbackup** utility. **vbackup** can save a database reliably while users are connected to the database, and provides a binary backup. The binary file can be used to restore the database. **vbackup** is intended primarily for routine disaster recovery. It is included with the Systems Architect/Requirements Management installation package.



Siemens PLM Software strongly recommends that you use **vbackup**. General purpose file backup utilities do not capture all the information for databases that are running.

vbackup can perform incremental backups. A level 0 backup writes the entire database to the backup device. A level 1 backup writes all of the changes since the last level 0 backup. A level 2 backup writes all of the changes since the last level 1 backup.

To back up the database to a file, enter the following command:

```
vbackup -level level -device fileName -backup dbname
```

Enter variable values as follows:

Replace *level* with the type of backup you want to perform (0,1,2). If *level* is not specified, **vbackup** assumes a level 0 backup.

Replace *fileName* with the fully qualified path name of the backup file.

Replace *dbName* with the name of the database. For example, **TCR_db**.

To back up the database to a tape, enter the following command:

```
vbackup -level level -device tape-device -position position -backup dbName
```

Enter variable values as follows:

- Replace *level* with the type of backup you want to perform. If *level* is not specified, **vbackup** assumes a level 0 backup.
- Replace *tape-device* with the name of the tape device.
- Replace *position* with the tape position to which you want to back up the database. If *position* is not specified, **vbackup** assumes the current position.
- Replace *dbName* with the name of the database. For example, **TCR_db**.

Restoring the Database

If you used the Versant **vbackup** utility to back up the database, you can use **vbackup** to restore the database from a backup file or tape. You must first restore backup level 0. If there are more backup levels, restore level 1 and then level 2.

To determine the number of levels necessary to restore the database to its last backup state, enter the following command:

```
vbackup -info dbName
```

Replace *dbName* with the name of the database. For example, **TCR_db**.

To restore the database from a backup file, enter the following command:

```
vbackup -device fileName -restore dbName
```

Enter variable values as follows:

Replace *fileName* with the fully qualified path name of the file from which you want to restore the database.

Remember to restore the latest good level 0 backup and then, the latest good level 1 backup. Then, restore the latest good level 2 backup that goes with the level 1 backup.

Replace *dbName* with the name of the database. For example, **TCR_db**.

vbackup responds with a question about saving or applying log file contents. Although the default reply is **Yes**, Siemens PLM Software recommends that you reply **No**.

Backing Up and Restoring the osc-dbid File

A Versant file named **osc-dbid** tracks all databases on the network. Every time a database is created or deleted, the **osc-dbid** file is updated. Versant consults this file before it assigns a unique number to each database.

Versant uses this unique database number to assign a unique ID (called LOID) to each object on the network. Therefore, the **osc-dbid** file must be backed up regularly.

Ask your system administrator to include the **osc-dbid** file in whatever backups are performed on your system. The **osc-dbid** file is usually located on the Versant server.

To determine the file's complete path name, run the following command:

```
oscp -o
```

To determine the system where the **osc-dbid** file is located, run the following command:

```
oscp -n
```



The **osc-dbid** file can be restored only if no Versant database was created since the previous backup of the **osc-dbid** file. If a new database was created after the last backup of the **osc-dbid** file, do not restore the **osc-dbid** file. Contact Systems Architect/Requirements Management Customer Support for assistance.

Increasing the Size of the Database

The initial size of the Systems Architect/Requirements Management database is 2 GB by default. This initial allocation can be exceeded as a project progresses.

You must increase the size of the database if you see the following exception message:

```
SM_E_OUT_OF_VOL_SPACE: all volumes exhausted.
```

To increase the size of the database:

1. Log in using the account designated as the Systems Architect/Requirements Management administrator.
- 2.

Add a volume by entering the following command:

```
addvol [-i] -n volName -p path -s size dbName
```

-i is optional. It causes the specified disk space (*size*) to be allocated and initialized immediately. That disk space is withheld from other applications, ensuring that the space is available when the Systems Architect/Requirements Management database grows enough to require it.

Enter variable values as follows:

- Replace *volName* with the name of the new volume, for example, **volume2**.
- Replace *path* with the same value as *volName* if you are adding the volume on the same disk as the original database. If you are adding the volume on another disk, replace *path* with the full path name of the new volume.



Using an absolute path name presents difficulties if the database is moved later. Siemens PLM Software recommends that you keep all volumes on the same disk. In either case, verify that sufficient space exists on the disk.

- Replace *size* with the quantity of disk space (in megabytes using **M** after the number, or in gigabytes using **G** after the number) that you want to add to the database.
- Replace *dbName* with the name of the database. For example, **TCR_db**.

For example, if you want to increase the size of the **TCR_db** database by 300 MB, and the name of the new volume is **vol2**, enter the following command on the machine where the database is located:

```
addvol -i -n vol2 -p vol2 -s 300M TCR_db
```

Running the Database Maintenance Utility

The database maintenance utility (**tcradmin**) performs diagnostics on the Systems Architect/Requirements Management database. This utility's **analyzeDB** option can check for inconsistencies that may arise, and can correct many of them when the **maintainDB** option is in effect. The utility makes two passes through the database. The physical pass looks for and reports low level database exceptions, if any. The logical pass detects and reports inconsistencies in Systems Architect/Requirements Management objects. The database maintenance utility (**tcradmin**) can examine the entire database, a specific project, all objects of a class, or a specific object.



Siemens PLM Software recommends that you schedule a regular job or task to run the database maintenance utility, perhaps weekly. If you have questions about scheduling, consult your system administrator.

Database maintenance can be run while the database is in use. However, it is preferable to run it when no users (or few users) are connected to the database, because inconsistencies could be mistakenly reported for objects involved in active transactions.

To use the database maintenance utilities (**maintainDB** and **analyzeDB**), you must be logged in to the system with the user ID that has a matching enterprise administrator user object in the Architect/Requirements database. You can use the `-user` command line option to identify the enterprise administrator account that you can use.

The **maintainDB** utility does the following:

- Creates a new **TcrAdminLog.html** file every time the **tcradmin** script is run.
If you do not want a new file to be created, run **maintainDB** with the **-startNewLogFile false** option.
- When verbose mode is set to **false**, **maintainDB** prints out error versus all thread activities such as:
 - o Begin first pass physical check
 - o 10000 objects

If you do not want the error versus all thread activities to be printed, run **maintainDB** with the **-verbose true** option.

The database maintenance utility (**tcradmin**) is enhanced with each release to detect and repair additional inconsistencies. So, a number of new messages may appear the first time you run it after upgrading. They are likely inconsistencies that have been present for a long time, but were never reported before. If they were not evident in using Architect/Requirements before installing this release, there is no need to be concerned about them now that they are detected and corrected.



The **tcradmin** command accepts a maximum of 18 command line arguments. Also, the operating system limits the amount of data that can be entered on the command line. If the command line exceeds these limits, the **tcradmin** utility does not run.

To avoid these limitations, **tcradmin** also accepts arguments in a file.

- The `-argFile` argument specifies the file name.
- The file must have only one argument per line.

To run **maintainDB** using an argument file:

```
Tcradmin -argFile arguments.txt
```

Replace *arguments.txt* with the name of the file that contains the arguments.

For example, the argument file can contain:

```
-action  
maintaindb  
  
-mode  
database  
  
-logToStdOut
```

Maintenance utility logging and summary

Information about the progress of the utility and problems detected in the database is captured in a log. Output from **analyzedb** and **maintainDB** is written to the Architect/Requirements administrative log file **TcrAdminLog.html**, by default. The **TcrAdminLog.html** file is created in the current directory. You can change the location of the **TcrAdminLog.html** file using the `-logdir <directoryname>` option.

Use the `-logToStdOut` option to redirect the output to **stdout**. To capture the information as plain text in a file, use the operating system mechanism for redirecting **stdout** to a file.



You can mail the summary file to the Architect/Requirements administrator as part of a maintenance script.

Running analyzeDB from a Command Prompt

analyzeDB examines database content for consistency, but does not modify anything. It cannot be run from a client. It is run from a command prompt on the Systems Architect/Requirements Management server where the database is located.

analyzeDB examines the content of each object and all relationships between objects. It can take a long time to check a large database, perhaps several hours to check millions of objects.

Running maintainDB From a Command Prompt

maintainDB does the same checks as **analyzeDB**, but then takes appropriate corrective action on most anomalies that it finds. It cannot be run from a client. It is run from a command prompt on the Systems Architect/Requirements Management server where the database is located.

Database Maintenance Options

The following table show options for the database maintenance utility (**tcradmin**).

Table 4-1. Database Maintenance Options

Command Line Option	Possible Value	Comments
-action (Required)	analyzeDB maintainDB	analyzeDB reports only errors, does not repair. maintainDB reports errors, repairs errors, and reports repair failure.
-mode (Optional) If no -mode is passed, the default is database. If -mode option is provided, you need to provide correct -name option.	<i>project name</i>	Checks only a specified project. If the project is not found, an error is reported. Enter the project name next to the <i>-name</i> option. <code>-mode project -name myProject</code>
	<i>file name</i>	Checks only those LOIDs provided in the LOIDs file. If a problem occurs, an error is reported. The LOIDs file should be a .txt file with one loid per line. The pathname needs to be provided with path name in quotes. <code>-mode file -name "C:\Temp\ TcR_LOIDs.txt"</code> Note that the LOID file can be created by running maintainDB with the -generateLooidsOnly option.
	<i>loid</i>	Check a specified LOID. You need to provide that LOID in quotes next to

Table 4-1. Database Maintenance Options

Command Line Option	Possible Value	Comments
		<p><i>-name</i> option. If the LOID is invalid, an error is reported.</p> <pre>-mode loid -name "403.0.12530"</pre>
<p><i>-name</i> (Depends on mode option)</p> <p>If -mode option is database and -name is not passed, default database is used. If default database not found, error message is reported.</p> <p>For all other -mode options, you need to provide correct -name option of either Project/file/LOID. Otherwise, error message is reported.</p>	<p><i>project name</i></p> <p><i>LOID file name</i></p>	<p>If -mode is selected as project, Project name.</p> <p>If -mode is selected as loid, LOID.</p> <p>If -mode is selected as file, fully qualified path name of the file.</p>
<p><i>-generateLoidsOnly</i> (Optional)</p>		<p>The generateLoidsOnly option gathers the objects to check, as specified by other options, and writes the LOIDs to a temporary file. The objects are not checked, just saved for later use. To check these objects run <code>maintaindb</code> again with the -file option.</p>
<p><i>-logToStdOut</i> (Optional)</p>		<p>Redirects the output of the tcradmin command to stdout. By default, most of the output of tcradmin is written to the Architect/Requirements administrative log file, TcrAdminLog.html.</p> <p>The Architect/Requirements server log is written in a tabular HTML format that is convenient to view. When redirected to stdout, the Architect/Requirements log information is written as plain text. To capture the information as plain text in a file, use the operating system mechanism for redirecting stdout to a file.</p>

Table 4-1. Database Maintenance Options

Command Line Option	Possible Value	Comments
<code>-class</code> (Optional) If any wrong class name is passed, check stops and reports an error message.		Checks objects with the specified class type. All other class are ignored. <code>-class RequirementDB</code> Checks RequirementDB objects only. If no -class detail provided, the default class used is AbstractKernelDB . For a list of all class names that can be used with the -class option, see appendix A, List of Class Names for Checking Database Objects .
<code>-verbose</code> (Optional)	<code>true</code> <code>false</code>	true: detailed report of the result. false: short summary of the result.
<code>-blocksize</code> (Optional) If no blocksize is passed, or if nothing is passed next to <code>blocksize</code> , or given as empty string, the default value is 10,000.		When there is multiple thread checking the database, the -blocksize option tells how many LOIDs per thread. <code>-blocksize 1000</code>
<code>-threads</code> (Optional) If no threads is passed, or if nothing is passed next to <code>threads</code> or given as empty string, the default value is 1		Indicates the number of threads to use for checking the database. <code>-threads 5</code>
<code>-user</code> (Optional)	<i>enterprise administrator name</i> For example, tcradm	Required only if the operating system account name running tcradmin does not match an Architect/Requirements enterprise administrator name.

The **-blocksize** and **-threads** settings help manage how the `analyzeDB` and `maintainDB` utilities take advantage of the memory and processors in the server.

Objects are processed in batches. The number of objects in each batch is determined by the **-blocksize** value. Each batch is processed by a thread. Each thread starts up, connects to the database, processes its batch of objects, disconnects from the database, and shuts down. A new thread is started and repeats the process for the next batch. This clean start for each thread insures that memory does not creep up as

processing continues. Smaller batch sizes use less memory, but the increased thread startup and shutdown can make the total process take longer. The default value is a reasonable starting place, but won't be optimum for all servers. Larger values can improve total processing time on servers with adequate RAM, and especially UNIX servers with a full 4 GB of address space in 32-bit processing mode. Smaller values may be necessary in special situations, but increase processing time.



The memory address space available for object checking is limited by Java runtime parameters within the tcradmin script. To allow the checking to use more (or less) memory, edit the script to change the **-Xmx** value, or Java maximum heap size. For example, set `-Xmx 1024M` for 1 GB of address space. The tcradmin script is in the location specified by the **ImportExportScript** configuration parameter.

Multiple batches can be processed concurrently by setting **-threads** to a value greater than 1. Multiple threads running in parallel can shorten the total processing time, up to the point where some system resource (CPU, RAM, database I/O) becomes saturated. Running multiple threads uses more CPU cycles and require more memory. The optimum number of threads will depend on the number of processors on the host, the total memory available, and consideration for any concurrent workload from TcSE users or other processing. Setting **-threads** to the number of physical processors in the server is a reasonable starting place. But, if other loading allows it, try 1.5 to 2 times the number of processors. Even servers with only a single processor show some benefit of using `-threads 2`.

Working from the defaults and these guidelines, the best values to use depends on local conditions; number of processors, physical RAM, database size, set of objects to be processed at different times, and the target run time window.

analyzeDB Examples

The following is a list of examples using the analyzeDB utility:

- The following example checks the complete database with only **RequirementDB** objects, and it performs only a logical check. With **-verbose** set to true, a detail report is generated. The report is created as **TcrAdminLog.html** file in the current directory. You can change the location of the **TcrAdminLog.html** file using the `-logdir <directoryname>` option.

There are maximum five live threads to examine and each thread takes 100 LOIDS to examine.

```
TcrServerDir\schemata\tcradmin -action analyzeDB
-mode database -class RequirementDB -type logical -verbose true
-blocksize 100 -threads 5
```

- In this example, you provide the LOIDS in a text file and only those LOIDs are examined. In this example, the **-logToStdOut** option is used so the output is seen as plain text in the console. The check is performed on those LOIDs you provide in the **-loid_file** option. Both a physical and logical check are performed because the **-type** option is not provided.

```
TcrServerDir\schemata\tcradmin -action analyzeDB
-mode file -name "C:\Documents and Settings\user\
Local Settings\Temp\Tcr_LOIDs.txt" -verbose true
-blocksize 100 -threads 5
```

- To examine the entire database, enter the following command:

```
TcrServerDir\schemata\tcradmin -action analyzeDB
```

Where *TcrServerDir* is the directory where the Systems Architect/Requirements Management server is installed. In Windows, it is the value of **USER_INSTALL_DIR** of the **HKEY_LOCAL_MACHINE\SOFTWARE\SIEMENS**

Teamcenter\SystemEngineering\Server registry key. In UNIX database, while logged in as the Systems Architect/Requirements Management owner, run the command:

```
cat $HOME/TcSE_Server.properties
```

- To examine a specific project, enter the following command:

```
TcrServerDir\schema\tcradmin -action analyzeDB -mode project  
-name projectName
```

Replace *projectName* with the name of the project.



On Solaris platforms, an error results if the *projectName* variable contains spaces, even if you place quotation marks around the project name. Instead of entering the project name directly, do the following:

- . Set the project name as an environment variable. For example:

```
setenv PROJECT_NAME "Project Name"
```

- . Enter the following command:

```
tcradmin -action analyzeDB -mode project -name $PROJECT_NAME
```

- To check a specified database class:

```
TcrServerDir\schema\tcradmin -action analyzeDB -class classname
```

The default is **all**, analyze all database classes

- To specify the type of test to perform:

```
TcrServerDir\schema\tcradmin -action analyzeDB -type  
[physical | logical | both]
```

Select an option as follows

- . **physical:** performs physical analysis only
- . **logical:** performs logical analysis only
- . **both:** performs both physical and logical checks.

maintainDB Examples

The following is a list of examples using the **maintainDB** utility:

- To examine and correct the entire database, enter the following command:

```
TcrServerDir\schema\tcradmin -action  
maintainDB -logToStdOut > "logFile"
```

Replace *logFile* with the name of the file in which you want to record the **maintainDB** results. The result is saved as a plain text file.

- To examine and correct a specific project, enter the following command:

```
TcrServerDir\schema\tcradmin -action maintainDB -mode project  
-name projectName
```

Replace *projectName* with the name of the project.



On Solaris platforms, an error results if the *projectName* variable contains spaces, even if you place quotation marks around the project name. Instead of entering the project name directly, do the following:

- Set the project name as an environment variable. For example:

```
setenv PROJECT_NAME "Project Name"
```

- Enter the following command:

```
tcradmin -action maintainDB -mode project -name $PROJECT_NAME
```

Null owner errors may be reported when running **maintainDB**. To correct the errors, **maintainDB** moves the objects with no owners to the user's Recycle Bin. The Recycle Bin used will be for the Architect/Requirements user whose name matches the account name of the user running **maintainDB**.

For example, if you are running **maintainDB** logged into the system with the **tcradm** user ID, the objects that report null owners are moved to the Recycle Bin of the corresponding **tcradm** user in Architect/Requirements.

Objects moved to the Recycle Bin can then be manually examined and either destroyed, by emptying the Recycle Bin, or restored, by moving them to a valid owner.

Running the database maintenance utility in delta mode

With large databases, running a complete **maintainDB** as part of regular database maintenance takes a lot of time. Delta **maintainDB** solves this problem by checking only the portion of the database that is likely to have issues. Inconsistencies are normally introduced when objects are modified. In delta mode, **maintainDB** checks only the recently modified database objects. Running the maintenance in delta mode detects almost all issues and takes much less time than a full **maintainDB**.

Recently modified LOID capture

For delta **maintainDB** to function, the **LOIDs** of modified database objects must be captured. To capture the **LOIDs** of the modified database objects, the delta capture mode must be enabled by setting the web application parameter **DB.DeltaCapture** to **true**. In the delta capture mode, the **LOIDs** of database objects modified in each transaction are written to the database.



Enabling the delta capture mode can cause a slight degradation in performance and a small increase in database size. You must enable the delta mode only if you intend to use delta **maintainDB**.

Running delta maintainDB from a command prompt

To run the delta **maintainDB** from the command prompt, execute the `tcradmin` command with the `-action maintainDB -mode delta` option. In the delta mode, **maintainDB** collects the **LOIDs** of recently modified database objects that are stored in the database. These objects are then checked as regular **maintainDB**. After delta **maintainDB** runs, the list of recently modified **LOIDs** is automatically removed from the database.

Delta maintainDB usages

You can use delta **maintainDB** in two ways. You can run delta **maintainDB** on a daily basis to check all database objects modified on that day. Delta **maintainDB** can be used as a replacement for a full **maintainDB** to improve performance. No special options are required in this mode. You can also run delta **maintainDB** incrementally during the day. This mode requires setting up a cron job or other mechanism to execute delta **maintainDB** on a regular basis. Running delta **maintainDB** more frequently spreads the maintenance load throughout the day and detects issues sooner. In this mode, the checks are done while the database is in use. Delta **maintainDB** provides options to reduce conflicts with other users. To reduce the conflicts, you must run delta **maintainDB** with the options mentioned in the following section.

Delta maintainDB options

There are **tcradmin** command line options that work only with delta **maintainDB**. The following table shows the options for delta **maintainDB**.

Command Line Option	Possible Value	Comments
<code>-action</code> (Required)	maintainDB	
<code>-mode</code> (Required)	delta	To check the recently modified objects only.

Command Line Option	Possible Value	Comments
-delaytime (Optional)	Time	To skip checking an object if it has been modified within the specified time in minutes. This helps prevent checking an object that is actively being modified by another user.
-objectlimit (Optional)	Count	To limit the maintainDB run to check only a specified number of objects. This helps prevent maintainDB from over consuming resources such as CPU and memory. Objects not checked remain in the recently modified LOID list and are checked in a future delta maintainDB run.



All options of **maintainDB** work for delta **maintainDB** also . However, **-class** and options specific to a particular mode do not work for delta **maintainDB**.

Delta maintainDB examples

- To examine and correct all recently modified database objects, enter the following command:

```
TcrServerDir\schemata\tradmin -action maintainDB -mode delta -logToStdOut > "logFile"
```

- To use delta **maintainDB** while lessening the impact on other database users, enter the following command:

```
TcrServerDir\schemata\tradmin -action maintainDB -mode delta -delaytime 30 -objectlimit 10000 -logToStdOut > "logFile"
```

A 30 minute delay avoids checking objects that are actively getting modified by other users. A 10000 object limit prevents **maintainDB** from overusing system resources.

Clearing the recently modified LOID list

When the modified object capture mode is on (the web application parameter **DB.DeltaCapture** is set to **true**), the **LOIDs** of the modified database objects are written to the database. This increases the database size if the list is not cleared regularly. The list is cleared automatically when delta **maintainDB** is run. The list can also be manually cleared with the following command:

```
TcrServerDir\schemata\tradmin -action dropDelta
```

After a full **maintainDB** is run, the modified object list is not required. The list can be cleared using the **-dropDelta** option. For example:

```
TcrServerDir\schemata\tradmin -action maintainDB -mode database -dropDelta
```

Temporarily disabling modified LOID capture

For operations such as a large project import, capturing modified **LOIDs** is not required. Capturing **LOID** is disabled by setting the **DB.DeltaCapture** parameter to **false**. **LOID** capture mode is temporarily enabled or disabled in a transaction with the **setEnvironment** command. For example:

```
setEnvironment deltaMode false
```

Stopping the Database

You must stop the Systems Architect/Requirements Management database before you reboot the machine on which the database resides. In addition, you must stop the database to perform some maintenance and tuning operations.



- There cannot be any client connections to the database when it is stopped.
- The Systems Architect/Requirements Management Web server application must be shut down.

To stop the database named *DBname*, run the following command:

```
stopdb DBname
```

For example, `stopdb TCR_db`.

Starting the Database

The Systems Architect/Requirements Management database need not be started explicitly. It is started automatically when a request for connection comes up. Sometimes, however, it may be necessary to start the database for diagnostic purposes.

To start the database named *DBname*, enter the following command:

```
startdb DBname
```

For example, `startdb TCR_db`.

Deleting the Database

To delete the Systems Architect/Requirements Management database named *DBname* completely from your system, enter the following command:

```
removedb -rmdir DBname
```

For example, `removedb -rmdir TCR_db`.



This command does not ask for confirmation.

Cleaning Up a Project in the Database

To recover the database space upon deleting a project in the Architect/Requirements client, a database cleanup operation is required. The cleanup operation can be performed in either of the following methods:

- **Default (Project.Cleanup.Now=false):**

Recover the database space by the **tcradmin** script on the database server machine, using the **projectCleanup** option. Run the following command to use this option:

```
tcradmin -action projectCleanup
```

The **projectCleanup** option searches the database for projects that are deleted. Then, it runs **maintainDB** with specific options that destroy all the objects in the deleted projects, therefore, recovering the database space.

The **tcradmin -action maintainDB** command (with no other options, therefore, running complete maintenance on the entire database) also performs the **projectCleanup** operation as the first step of **maintainDB**.

- **Optional (Project.Cleanup.Now=true):**

Set the Architect/Requirements Web application parameter **Project.Cleanup.Now** to **true**, which cleans up the database space immediately by spawning an external process on the application server. Then, run the **tcradmin -action projectCleanup** command.

Project.Cleanup.Now uses the **schema** directory on the application server machine; therefore, the required setup must be in place. **Project.Cleanup.Now** runs on the application server, which adds significantly to the network load during the cleanup operation. Therefore, the default cleanup method is to use **maintainDB** on the database server.

For more information on configuring project cleanup, see [Project.Cleanup.Now](#).

The **tcradmin -action maintainDB** command (with no other options, therefore, running complete maintenance on the entire database) also performs the above **projectCleanup** operation, as the first step of the **maintainDB**.

Running System Utilities

Several utilities display information that can be helpful in diagnosing network and web server installation problems. They are detailed in this section.

In Internet Explorer, open the Systems Architect/Requirements Management home page, and then click the **Administrative Tools** link. From the Administrative Tools page, click the **Diagnostic Tools** link.

From the Diagnostic Tools page, you can access the following options:

- **TcSE Home:** Return to the Systems Architect/Requirements Management home page.
- **Server Properties:** Monitor Java environment properties.
- **Server Information:** View Web server information including server name and IP address, protocol, server port, and application server.
- **Web Application Configuration:** Access the Web Application Configuration page where you can configure Systems Architect/Requirements Management.
- **Office Live Diagnosis:** Access the Office Live Diagnosis page. This page tells users if their local machine is properly configured to run Office Live, which lets you create, edit, view, and manipulate objects in Systems Architect/Requirements Management through the Microsoft Office suite of products.
- **Server Installation Diagnosis:** View information regarding the server installation in the event of an installation problem.
- **Server Integration:** View information regarding Application Registry installation in the event of an Application Registry installation problem.
- **List Current Users:** View a list of all users logged in to Systems Architect/Requirements Management. Shows each user's full name, email, the time he or she logged in, and the status of the session. The status is based on the duration that the session is left idle. If session is idle longer than the applications **session-timeout** value as set in **web.xml** file, it is flagged as **Dormant**.



Systems Architect/Requirements Management allows for enterprise deployments of more than one Application server. This utility works by detecting sessions at the Application server level. So, if you have more than one Application server deployed, you need to connect to each Application server individually to get the list of *all* users logged into Systems Architect/Requirements Management.

Chapter 5: Managing Licenses

This chapter describes Architect/Requirements license key files and contains instructions for working with license keys. A table of possible server errors is included for troubleshooting.

Architect/Requirements License Key Files

To use the Architect/Requirements server, a license key file (**tcr.lic**) must be obtained from Siemens PLM Software. You receive this file when you register your Architect/Requirements application. With the license key file, you also receive a customer number for your organization.

When you obtain the license key file and customer number, you use the Web Application Configuration page to configure license information. For more information, see [Configuring License Information](#) in chapter 2, *Configuring Architect/Requirements*.



The Architect/Requirements server must be installed before you can configure license information. For more information, see the *Systems Architect/Requirements Management Server Installation Manual*.

A license key file is an encrypted binary file and consists of the following:

- - **Customer Number**

Specifies the number assigned to your organization by Siemens PLM Software Customer Service. The number should be entered into the Configuration Web page (<http://.../tcr/ugs/tc/req/configtcr.jsp>).
 - **Expiration Date**

Specifies the date on which the license becomes invalid. After this date, the Architect/Requirements server will not start.
 - **IP Address**

Specifies the IP address at which the Architect/Requirements server is licensed to run.
 - **Number of Seats**

A **Read/Write** license is consumed if a user's maximum privilege property is **Read-Write**, **Project Administrator**, **Enterprise Administrator**, or **Database Administrator**.

A **Read/Only** license is consumed if a user's maximum privilege property is **Read Only**. However, if a **Read/Only** license is not available, then a **Read/Write** license is consumed.

A **Scripting** license is consumed when a user is given **Script Authoring** privilege for at least one project.



Users need the **Script Authoring** privilege to create, edit, and run activators in each project. To give this privilege to a user for a particular project, change the user object in the project's **Users** folder by setting the **Additional Privilege** property to include **Script Authoring**.

Because enterprise administrators have access to all projects in the system, their user objects appear only in the **Users** folder of the **TcSE Administration** project. However, a user does not need **Script Authoring** privilege if an activator is triggered as a result of another user's actions.

•

Version Number

Specifies the license version that corresponds to the version of the Architect/Requirements application.

On startup of the Architect/Requirements server, each data item in the license key is checked for validity.

Viewing License Information

For each type of Architect/Requirements license, you can view the following information:

- The total number of licenses for your customer number.
- The number of licenses that are in use.

To view license information:

1. On the Teamcenter 11.1 systems engineering and requirements management home page, click the **Administrative Tools** link.
2. On the Administration Tools page, click the **TcSE Licensing** link.

The License Information page displays the license count from all current license keys.

Also on this page, you can click the **Manage Licenses** link to access the license management utility. For more information, see [Accessing the License Management Utility](#), later in this chapter.

Accessing the License Management Utility

You access the license management utility through the License Management Page. On this page, you manage licenses by adding and removing license keys. For more information, see [Adding a License Key](#) and [Removing License Keys](#), later in this chapter.

A license key is an encrypted text string that controls a certain number of licenses. Each license key contains information such as your customer number, an expiration date, and the number of seats for each license type.



- Your valid customer number must be entered before you start this procedure. For more information, see [Configuring License Information](#) in chapter 2, *Configuring Architect/Requirements*.
- **Enterprise Administrator** privilege is required for this procedure.

To access the license management utility:

1. On the Teamcenter 11.1 systems engineering and requirements management home page, click the **Administrative Tools** link.
2. On the Administrative Tools page, click the **TcSE Licensing** link.
3. On the License Information page, click the **Manage Licenses** link.
4. On the Teamcenter 11.1 systems engineering and requirements management login page, enter your enterprise administrator user name and password, select a language, and click **Log In**.

The License Management Page is displayed.



An alternate page is displayed if your database is new and if your customer number is not entered. This page contains a link to the Web Application Configuration page. Click this link, enter your customer number in the **LIC.CustomerNumber** parameter, and click the **Update** button. Then repeat this procedure.

Adding a License Key

You can add a new license key without replacing existing licenses. The licenses in the new key are automatically included in the total license count for the Architect/Requirements server.

You can view the server's total license count on the License Information page. For more information, see [Viewing License Information](#), earlier in this chapter.



Enterprise Administrator privilege is required for this procedure.

To add a license key:

1. Open the license key file in a text editor (for example, Microsoft Notepad), and copy the encrypted string to the clipboard.
2. In the text field at the bottom of the License Management Page, delete the words “Enter Key Here,” and then paste the encrypted string into the field.

For more information, see [Accessing the License Management Utility](#), earlier in this chapter.



You can reverse this action by clicking **Clear**.

3. Click **Add Key**.

The license management utility checks the license key. When the license key is validated, a confirmation page displays the license key information in unencrypted format.



If the license key is invalid, expired, or a duplicate, an error message is displayed. Click **Back** to return to the License Management Page, where you can enter the key again or enter another key.

4. Click **Add**.

The License Management Page displays the new license key in the table of encrypted strings.



If you have two or more Web application servers that point to the same database, and if you manage each server separately through the **Installation Key** parameter in the web.xml file, you must add this license key on each server.

Removing License Keys

When a license key expires, its licenses are automatically subtracted from the total license count for the Architect/Requirements server. However, expired license keys remain in the database until you remove them.

You can view the server's total license count on the License Information page. For more information, see [Viewing License Information](#), earlier in this chapter.



Enterprise Administrator privilege is required for this procedure.

To remove license keys:

1. In the table of encrypted strings on the License Management Page, check the check box for each license key that you want to remove.

For more information, see [Accessing the License Management Utility](#), earlier in this chapter.

2. Click **Remove Key**.

A confirmation page displays the key number and encrypted string for each selected license key.



To return to the License Management Page without removing any licenses, click **Back**. You can click **Cancel** to abort this operation and exit the license management utility.

3. Click **Remove** to remove all licenses in each displayed license key.

The License Management Page is displayed with the license keys removed from the table of encrypted strings.



If you have two or more Web application servers that point to the same database, and if you manage the servers separately through the **Installation Key** parameter in the web.xml file, you must remove these same license keys from each server.

Troubleshooting Server Errors

Table 5-1 describes the types of server errors that can occur because of incorrect license configuration.

Table 5-1. Possible Server Errors in Systems Architect/Requirements Management License Configuration

Type of Error	Reason for Error
<code>InvalidLicenseException: Invalid License</code>	<p>Either of the following:</p> <ul style="list-style-type: none">• The expiration date has passed.• The version number in the Configuration Web page does not match the Systems Architect/Requirements Management version number.
<code>InvalidLicenseException: Given final block not properly padded.</code>	<p>Any of the following:</p> <ul style="list-style-type: none">• The Configuration Web page does not contain the customer number.<ul style="list-style-type: none">• The customer number in the database is invalid.•• The version number on the Configuration Web page does not match the Systems Architect/Requirements Management version number.
<code>License Management not configured.</code>	<p>Security libraries are not located in the appropriate directories.</p>

Appendix A: List of Class Names for Checking Database Objects

This appendix lists the class names that can be used with the **-class** option of the database maintenance utility (**tcradmin**). The **tcradmin** options are described in table [Database Maintenance Options](#) of chapter *Maintaining the System*.

The **-class <className>** option of the **tcradmin** command checks objects in the database with the specified class name. All other classes are ignored.

Table A-1. Class Names for -class Option

Class Name	Class Name	Class Name
AbstractApplicationDB	ConnectionDB	ParamDB
AbstractAttributeDB	CounterDB	PortDB
AbstractAttributeDefinitionDB	DateAttributeDB	ProjectDB
AbstractContainerDB	DateDefinitionDB	ProxyDB
AbstractDefinedPropDB	DiagramDB	RequeueProcessInfoDB
AbstractDesignDB	DiagramLinkDB	RequirementDB
AbstractGroupDB	DiagramOwnerLinkDB	SchemaUpdateTaskDB
AbstractHandleDB	DiagramStencilLinkDB	SearchDB
AbstractKernelDB	DocumentTemplateDB	SecurityProfileDB
AbstractLinkDB	EmptyTrashTaskDB	ServerRunningCheckRuleDB
AbstractNamedDefinitionDB	EventLinkDB	SessionDB
AbstractOutputTemplateDB	ExcelTemplateDB	ShortCutDB
AbstractOwnedObjectDB	ExportedProxyDB	ShortCutLinkDB

Table A-1. Class Names for -class Option

Class Name	Class Name	Class Name
AbstractProcessInfoDB	ExportedProxyLinkDB	SpreadsheetDB
AbstractQueryDB	FillinAttributeDB	StencilDB
AbstractQueuedTaskDB	FillinDefinitionDB	StringDB
AbstractRuleDB	FolderDB	StyleSheetDB
AbstractShortCutDB	GenericLinkDB	SymbolicLinkDB
AbstractUsesLinkDB	GroupDB	TcIntegrationSOAPTaskDB
AccessLinkDB	GroupLinkDB	TemplateDB
AccessProfileDB	HierShortCutDB	TestTaskDB
ActivatorDB	LinkDB	TimerRuleDB
AdminFolderDB	MapDB	TraceLinkDB
AttachmentDB	MarkerDB	TransactionDB
BuildingBlockDB	MatlabDB	TrashCanDB
ChangeApprovalDB	MessageQueueDB	UserDB
ChangeLogDB	MessageStoreDB	UserGroupDB
ChoiceAttributeDB	NoteDB	UserTypeDefinitionDB
ChoiceDefinitionDB	NumericAttributeDB	WolfHandleDB
ChoiceStringDB	NumericDefinitionDB	

Appendix B: Modifying the Client Memory Allocation

Architect/Requirements is configured for optimum memory allocation for machines with **4 GB** RAM. However, depending on your computers, you may need to increase or decrease the client memory allocation.

To modify the client memory allocation, you need to update the **login.jsp** file in the **tcr.war** file.

To modify the client memory allocation

1. Create a temporary folder and extract the **launch.jsp** file from the **tcr.war** file.

Open the command prompt, change to the directory location of the **tcr.war** file, and run the following command:

```
jar xvf tcr.war ugs/tc/req/launch.jsp
```



You must have the JDK installed to run the **jar** command.

You can download JDK from

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

2. Open the **ugs/tc/req/launch.jsp** file in a text editor like **Textpad** or **Notepad++**.
3. Search for the following string:

```
<PARAM name="vm_args" value="-Xms64m -Xmx1024m"> <!-- vm args for launching client -->
```

Xmx1024m in the string denotes the maximum heap space. In this case, the heap space is set to **1024 MB** or **1 GB**.

Change it to the desired value.

For information about the optimum heap space, see the Java documentation at docs.oracle.com.



There are two instances of the string. One set of values is for Internet Explorer and other standard browsers. The second set of values are targeted towards the Firefox browser.

Repeat the search and update the value of **Xmx1024m** in both the instances.

4. To prevent adding the existing **tcr.war** file to the new WAR file, delete or move it (the existing **tcr.war** file) from the temporary folder.
5. Add the updated **launch.jsp** file to the **tcr.war** file. Run the following command at the command prompt:

```
jar uvf tcr.war ugs/tc/req/launch.jsp
```

6. Deploy the war file on your Web server.

Appendix C: Configuring Security Services Logging

Security services has a logging system separate from the TcSE server log. Security services log files are located in the directory `./logs/TcSS`. The logs will contain any errors or warnings detected.



By default the log file is created relative to the current directory for the application server. This is commonly the application servers root directory. But you may need to determine what the current directory is to locate the logs.

If you need to change the log file location or get more detailed information for debugging there is an XML file used for configuration. The Configuration file is in the war file at this at this location, **WEB-INF\classes\log4j2.xml**.

To modify the configuration:

1. Create a temporary folder and extract the **log4j2.xml** file from the **tcr.war** file.

Open the command prompt, change to the directory location of the **tcr.war** file, and run the following command:

```
jar xvf tcr.war WEB-INF/classes/log4j2.xml
```



You must have the JDK installed to run the **jar** command.

You can download JDK from

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>.

2. Edit `WEB-INF/classes/log4j2.xml` and locate the security services logger entries:

```
<Logger name="com.teamcenter.ss" level="WARN" additivity="true">
```

```
  <AppenderRef ref="javaClientFileWriter"/>
```

```
</Logger>
```

```
<Logger name="com.teamcenter._ss" level="WARN" additivity="true">
```

```
  <AppenderRef ref="javaClientFileWriter"/>
```

```
</Logger>
```

3. Change the log level by replacing **WARN** with **INFO** or **DEBUG**.
4. Change the log file location by changing the `AppenderRef` entry. If you want the messages in the application servers console change **javaClientFileWriter** to **STDOUT**.

5. Add the updated **launch.jsp** file to the **tcr.war** file. Run the following command at the command prompt:

```
jar uvf tcr.war WEB-INF/classes/log4j2.xml
```

6. Deploy the war file on your Web server.

Appendix D: Reverting to Security Services 11.2 or Previous Versions

By default, Architect/Requirements will integrate with Teamcenter Security Services version 11.3 or higher for the purpose of adding single sign-on (SSO) functionality. It is also possible to integrate with Teamcenter Security Services 11.2 and previous versions by modifying the Architect/Requirements war file.



Teamcenter Security Services 11.2 and previous versions requires the use of Java applets. Reverting to any of these versions will limit users to versions of web browsers and Java 8 updates that support applets.



Teamcenter Security Services 11.2 and previous versions do not provide the logging feature described in Appendix C [Configuring Security Services Logging](#). That feature would therefore no longer be available following completion of the procedure in this appendix.

To integrate Architect/Requirements with Security Services 11.2, or previous versions, you must replace the Security Services client interface jar files, **WEB-INF\lib\teamcenter_sso_applib.jar** and **WEB-INF\lib\teamcenter_sso_common.jar**, within the **tcr.war** file with corresponding files of the same name from the version of Teamcenter you intend to use.

To revert to a previous version of Security Services

1. Create a temporary folder on a Architect/Requirements application server.
2. Copy the **tcr.war** file to newly created temporary folder.
3. Copy the Security Services redistributable archive corresponding from the version of Security Services you intend to use to the temporary folder. For example, the client interface jar files for Security Services 11.2 can be extracted from the following archive located in the Architect/Requirements 10.1 distribution archive folder :

Root-Folder\TcServices\TeamcenterSecurityServices\11.2.2\TcSecurityServices11.2.2_20160331.zip

4. Open a command prompt from the temporary folder and extract the the Security Services client interface jar files from the Security Services redistributable archive. Continuing the example, execute the following at the command prompt::

```
jar xvf TcSecurityServices11.2.2_20160331.zip TcSecurity\default\TEAMCENTER_SSO_APPLIB.jar
jar xvf TcSecurity\default\TEAMCENTER_SSO_APPLIB.jar WEB-INF\lib\teamcenter_sso_applib.jar
jar xvf TcSecurityServices11.2.2_20160331.zip TcSecurity\default\TEAMCENTER_SSO_COMMON.jar
jar xvf TcSecurity\default\TEAMCENTER_SSO_COMMON.jar WEB-INF\lib\teamcenter_sso_common.jar
```

5. Finally, update the **tcr.war** file with the newly extracted the Security Services client interface jar files. Completing the example:

```
jar uvf tcr.war WEB-INF\lib\teamcenter_sso_applib.jar WEB-INF\lib\teamcenter_sso_common.jar
```

6. Deploy the war file on your Web server.

Index

addvol command.....	46	Web.....	17
addvol utility.....	40	ConfigTcr.xml file.....	66
analyzeDB		Configuration options	
analyzeDB.....	49	Setting.....	34
Application map component.....	17	Configuration Web page.....	20, 66
Application registry service		Configuring Log Server Paramters.....	34
Application address.....	22	Configuring Message Queues.....	36
Application port.....	22	Configuring separate Java processes for data	
Application Registry URL.....	22	imports and exports.....	27
GUID.....	22	Configuring Systems Architect/Requirements	
Web application protocol.....	22	Management	
Architect/Requirements Client		License.....	24
Installation in Silent Mode.....	19	Object linking, Teamcenter products.....	20
Architecture.....	14	Conventions	
-argFile argument, tcradmin utility.....	48	Code.....	10
Argument file, tcradmin utility.....	48	Command line entries.....	10
Backing up		File contents.....	10
osc-dbid file.....	45	Names.....	9
Systems Architect/Requirements Management		Revisions.....	9
Database.....	44	Values.....	9
Backup levels.....	44, 45	Current users	
Check Database.....	47	List Current Users.....	59
Check Database utility.....	47	Custom.Folder.Path parameter.....	28
Checking Systems Architect/Requirements		Customer number.....	61, 66
Management database.....	47	Customizing	
Cleaning Up a Project.....	43	Export	
Cleaning Up a Project in the Database.....	57	Separate process.....	27
Client component.....	17	Import	
Client tier.....	14	Separate Java process.....	27
Code conventions.....	10	Customizing Systems Architect/Requirements	
Command line entry conventions.....	10	Management server.....	25
Commands.....	49	Customizing Systems Architect/Requirements	
addvol.....	46	Management sign on.....	31
Database Maintenance.....	47	Database	
oscp.....	40, 46	Backing up.....	44
projectCleanup.....	57	Checking.....	47
removedb.....	57	Cleaning Up a Project.....	57
startdb.....	57	Deleting.....	57
stopdb.....	57	Increasing size.....	46
vbackup.....	44, 45	Maintenance.....	43
Components.....	16	osc-dbid file, backing up and restoring.....	45
Client.....	17	Parameters	
Database server.....	18	lock_wait_timeout.....	41

max_page_buffs.....	41	ImportExportScript parameter	28
transaction	41	Increasing size of Systems	
reorgdb utility.....	42	Architect/Requirements Management	
Restoring.....	45	database	46
Root directory	40	Incremental backups	44, 45
Starting	57	Installing Architect/Requirements Client in	
Stopping	57	Silent Mode.....	19
Database Maintenance command.....	48	IP address, Systems Architect/Requirements	
Options	49	Management server.....	61
Database server component	14, 18	J2EE Application Server.....	14, 61
DB.FastNumberRefresh parameter.....	27	Java process	
DB.greadSize parameter	30	Export	27
DB.LockTimeOut parameter	25	Import	27
DB.MaxSession parameter	25	Java Runtime Environment component.....	17
DB.MaxSessionAge parameter	26	Java Server page	20
DB.MaxSessionSize parameter.....	26	JRE.Version configuration option	34
DB.UndoLevels parameter	27	Kernel configuration file.....	43
Deleting Systems Architect/Requirements		LIC. Customer Number	24
Management database	57	License	
Deployment tiers.....	14	Configuring Systems Architect/Requirements	
Directories		Management	24
etc.....	43	File	20, 24
system	43	Installing	61
tcr	61	tcr.lic	61
TCR_db.....	40	Troubleshooting server errors.....	66
Versant root.....	40	License key file.....	61
EnableAdditionalSecurity parameter	30	License management utility	64
Enterprise tier.....	14	lock_wait_timeout parameter	41
etc directory	43	Log files.....	40
Expiration date	61, 66	Log.Server.Level parameter	34
Export		Log.Server.Location parameter	35
Customizing		Log.Server.MaxBackupIndex parameter.....	35
Separate Java process.....	27	Log.Server.MaxFileSize parameter	35
ExternalImportExport parameter	27	Log.Server.MemoryMonitorInterval parameter	
File contents conventions.....	10	35
Files		Log.Server.Tcl.Level parameter	35
logical.log	40	logical.log file	40
osc-dbid.....	45	LOID.....	45
physical.log	40	MailServerIP parameter.....	28
profile.be	40	maintainDB	
system	40	maintainDB.....	49
tcr.lic	24, 61	max_page_buffs parameter.....	41
UNIX kernel configuration	43	Maximum privilege levels	61
Filter.compression Threshold parameter.....	32	Message Queues	
Filter.debug parameter	32	Configuring.....	36
Filter.doCompression parameter.....	32	MessageQueue.Master parameter.....	28
Hosts	14, 18	MessageQueue.Start parameter	29
Import		MessageQueue.Thread.SleepTime parameter .	29
Customizing		Monitor.PollTime parameter	30
Separate Java process.....	27	Multitier Web deployment.....	14
ImportExportDir parameter	28	Name conventions	9

Number of seats	61	SSO.LoginURL	31
Object identifier (LOID)	45	SSO.ServiceURL	31
Object linking.....	20	transaction.....	41
Object Linking parameters.....	22	WOLF.AppGUID	22
osc-dbid file	45	WOLF.AppIP	22
oscp command	40, 46	WOLF.AppPort	22
Package.Location configuration option	34	WOLF.AppProtocol	22
Parameters		WOLF.AppRegistryURL	22
Custom.Folder.Path.....	28	WOLF.ChooserURL.....	22
DB.FastNumberRefresh.....	27	WOLF.IconURL.....	23
DB.greadSize	30	WOLF.LauncherURL.....	23
DB.LockTimeOut	25	WOLF.LinkHandlerURL.....	23
DB.MaxSession	25	WOLF.Remote.Trace.Deep.....	30
DB.MaxSessionAge.....	26	WOLF.SoapURL.....	23
DB.MaxSessionSize.....	26	Password setting	32
DB.UndoLevels	27	Performance.....	14
EnableAdditionalSecurity	30	physical.log file	40
Enabling Security Services	31	PortRangeEnd parameter.....	29
ExternalImportExport	27	PortRangeStart parameter.....	29
Filter.compression Threshold.....	32	profile.be file	40
Filter.debug	32	Project.Cleanup.Now parameter.....	27
Filter.doCompression.....	32	projectCleanup command.....	57
ImportExportDir	28	projectCleanup feature.....	43
ImportExportScript	28	RCF..... See Rich Client Framework component	
LIC. Customer Number	24	RegeditEnabled parameter.....	30
License	24	removedb command	57
lock_wait_timeout.....	41	reorganize data volumes	42
log server.....	34	Repairing Systems Architect/Requirements	
Log.Server.Level.....	34	Management database.....	47
Log.Server.Location	35	Requirements service component.....	17
Log.Server.MaxBackupIndex	35	Reservations.....	25
Log.Server.MaxFileSize	35	Resource tier	14
Log.Server.MemoryMonitorInterval	35	Response handler component	17
Log.Server.Tcl.Level	35	Restoring	
MailServerIP	28	osc-dbid file	45
max_page_buffs.....	41	Systems Architect/Requirements Management	
MessageQueue.Master	28	Database.....	45
MessageQueue.Start.....	29	Revision conventions.....	9
MessageQueue.Thread.SleepTime.....	29	Rich Client Framework component.....	17
Monitor.PollTime.....	30	SAM utility	43
object linking	22	Seats.....	61
PortRangeEnd	29	Security Services	
PortRangeStart	29	Customizing.....	31
Project.Cleanup.Now	27	Server	
RegeditEnabled.....	30	Customizing.....	25
Security Services.....	31	Errors, license	66
Security Services login URL	31	IP address.....	61
Security Services service URL	31	Setting UNIX kernel parameters	43
server.....	25	Setting Versant database parameters	40
SSO.AppID	31	Shared memory	43
SSO.Enabled.....	31	SSO.AppID parameter	31

SSO.Enabled parameter	31	Version number	62, 66
SSO.LoginURL parameter.....	31	Web deployment.....	14
SSO.ServiceURL parameter	31	Tags	
startdb command	57	License.....	24
Starting Systems Architect/Requirements		log server	34
Management database.....	57	object linking	22
stopdb command	57	Security Services	31
Stopping Systems Architect/Requirements		server	25
Management database	57	TcFoundation WebFrame component	17
System Administration		tcr directory.....	61
projectCleanup feature	43	tcr.lic file.....	61
system directory	43	TCR_db directory	40
system file	40	tcradmin script	
Systems Architect/Requirements Management		External Java process, data import and export	
Architecture.....	14	27
Components	16	Script file location	28
Client.....	17	tcradmin utility	
Database server	18	command	
Web	17	Arguments, limitations on	48
Configuring		options	49
License	24	Teamcenter Linking.....	22
Object linking, Teamcenter products	20	Tiers.....	14
Customer number.....	61, 66	transaction parameter.....	41
Database		Troubleshooting server errors, license.....	66
Backing up	44	UNIX kernel	
Checking	47	Configuration file	43
Deleting.....	57	Setting parameters	43
Increasing size.....	46	User passwords	
osc-dbid file, backing up and restoring....	45	Configuring.....	32
Repairing.....	47	User privilege levels	61
Restoring.....	45	Utilities	
Starting.....	57	addvol	40
Stopping	57	Check Database	47
Expiration date	61, 66	License management	63, 64
Hosts	14, 18	List.....	59
License		Office Live diagnosis.....	59
Configuring	24	running.....	59
Server errors.....	66	Running	59
License file.....	20, 24	SAM	43
License key file	61	Server information	59
Performance	14	Server installation diagnosis	59
Reservations.....	25	Server integration	59
Seats	61	Server properties.....	59
Security Services		TcR Home.....	59
Customizing	31	vbackup.....	44, 45
Server		Web Application configuration	59
Configuring log	34	Value conventions	9
Customizing	25	vbackup command.....	44, 45
Errors, license	66	vbackup utility	44, 45
IP address	61	Versant	
User privilege levels	61	Client component.....	18

Database parameters		Web tier	14
lock_wait_timeout.....	41	WOLF. parameters	
max_page_buffs.....	41	AppGUID	22
transaction.....	41	AppIP.....	22
Database server component	18	AppPort.....	22
Object identifier (LOID)	45	AppProtocol.....	22
Root directory	40	AppRegistryURL.....	22
Version number.....	62, 66	WOLF.ChooserURL parameter.....	22
Web Application		WOLF.IconURL parameter.....	23
Configuration	59	WOLF.LauncherURL parameter.....	23
Web application server component.....	14, 18	WOLF.LinkHandlerURL parameter.....	23
Web deployment	14	WOLF.Remote.Trace.Deep parameter.....	30
Web page		WOLF.SoopURL parameter.....	23
Configuring Systems Architect/Requirements			
Management.....	20		

Teamcenter 11.1 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management API Reference

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	4.0	December 2003
B	4.1	February 2004
C	5.0	July 2004
D	6.0	March 2005

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
E	2005	September 2005
F	2005 SR1	June 2006
G	2007	December 2006
H	2007.1	April 2007
I	2007.2	September 2007
J	2007.3	January 2008
K	8	January 2009
L	8.0.1	June 2009
M	8.1	October 2009
N	8.2	October 2010
O	9	July 2011
P	9.1	May 2012
P1	9.1.5	January 2014
Q	10.0	January 2015
R	10.1	September 2016

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	11.1	March 2018

This edition obsoletes all previous editions.

API Reference Contents

Manual History	4
Contents	7
Preface.....	11
Audience	11
Conventions	11
Names and Values	11
Command Line Entries, File Contents, and Code.....	12
Submitting Comments.....	12
Proprietary and Restricted Rights Notice.....	13
Chapter 1: Introduction	15
Chapter 2: API Overview.....	17
Standard Input Parameters	17
Listing of API Methods.....	18
Schema Object Display Name and their Actual Name	22
Chapter 3: Using the Java API Client Library to Access the API	23
Using the API.....	23
Logging In to the API.....	23
Describing the API Functions.....	24
Describing ResultBeans	24
Internal vs External Java API	25
Describing DataBeans.....	29
Java API Methods	29
Java API Examples	30
Chapter 4: Using C# API from COM (VBA)	63
Introduction.....	63
Configuring COM Client (VBA)	64
Prerequisites.....	64
Adding Reference to TcR.tlb	64
Connecting to Systems Architect/Requirements Management.....	65

VBA Examples for Calling C# API Methods	67
Chapter 5: Using the Tcl Scripting API to Access the API.....	77
Introduction.....	78
Executing Tcl Scripts	79
Transaction Management.....	79
Parameter Types.....	79
Listing of Tcl Methods.....	79
Chapter 6: Using Activators.....	141
Introduction.....	141
Access Privileges for an Activator.....	141
Describing Activator Objects.....	142
Creating Activators	143
Using Activators in Excel and Object Templates	143
Defining Events.....	145
Defining Object Modify Events.....	145
Defining Session Modify Events	148
Change Approval Routing Events	149
Import Events.....	150
Storing Event Context.....	150
Defining the Change List.....	151
Defining Flags.....	152
Defining Relation Flags	152
Defining Modify Flags.....	153
Defining Delete and Create Flags	153
Using A Pick List Activator.....	154
Pass Owner to Tcl Context	154
When A Pick List Activator Gets Called.....	155
Implementing Transaction Control	158
Creating a Tcl Where Clause	159
Creating a Where Clause	159
Searching With a Tcl Where Clause.....	159
Tcl Where Clause Example	160
Tcl Global Variables in Where Clause Activators.....	162
Writing Activators When Objects are Deleted, Restored, or Discarded.....	163
TC_XML Export Activator.....	164
Assigning Teamcenter Item IDs	164
Excluding data from export	165
Adjusting type and property names	165
Specifying the versions to export.....	166
LOID Property	167
Activator Examples.....	167
Determining Requirements With The Same Name.....	167
Creating a Note	167
Creating An After Delete Activator Event.....	169
Using createAction FileDownload.....	170

Using createAction RunJava.....	171
Running an Activator From the Command Line With the teradmin Script.....	179
Using Macros	180
Creating a Macro	180
Running a Macro	181
Macro Examples	181
Working with Shortcut Objects.....	183
Working with Reference Links	185
References to Versioned Objects.....	185
Chapter 7: Using Icon Overlay	187
Chapter 8: Using Generic Links.....	189
Introduction.....	189
Support for Generic Links.....	189
Examples of API Methods	190
Chapter 9: Cross-Product Messaging	191
proxyAction	191
Setting Up proxyAction in Systems Architect/Requirements Management.....	191
Incoming proxyAction Requests.....	191
Sending proxyAction Requests.....	192
Setting Up proxyAction in Teamcenter Engineering.....	192
Remote Proxy Objects and Tcl API.....	192
Get Remote Proxy Properties.....	192
Appendix A: Word Content Activators	193
Word Edit Pre-Processor.....	193
Word Edit Post-Processor	194
Word Export Pre-Processor	194
Word Export Post-Processor.....	194
Appendix B: Examples for using C# API's.....	197
Accessing Using VBA	197
Accessing Using C#.....	199
Index.....	200

Preface

This manual is an API reference for Teamcenter Systems Architect/Requirements Management 11.1. Systems Architect/Requirements Management belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual contains information for Systems Architect/Requirements Management developers. It assumes that you understand the structure of the Systems Architect/Requirements Management product and are familiar with the installation and maintenance of Systems Architect/Requirements Management.

Conventions

This manual uses the conventions described in the following sections:

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **initsid.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
<i>Italic</i>	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In initsid.ora , <i>sid</i> identifies a varying portion of the name (a unique system ID). For example, the name of the file might be: initBlue5.ora
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Command Line Entries, File Contents, and Code

This manual represents command line input and output, the contents of system files, and computer code in fonts that help you understand how to enter text or to interpret displayed text. For example, the following line represents a command entry:

```
msqlora -u system/system-password
```

The conventions are:

Monospace	Monospace font represents text or numbers you enter on a command line, the computer's response, the contents of system files, and computer code. Capitalization and spacing are shown exactly as you must enter the characters or as the computer displays the characters.
<i>Italic</i>	Italic font represents text or numbers that vary. The words in italic text describe the entry. The words are shown in lowercase letters, but the varying text may include uppercase letters. When entering text, use the case required by the system. For the preceding example, you might substitute the following for <i>system-password</i> : KLH3b
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

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

© 2018 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Chapter 1: Introduction

This chapter contains introductory information about the Systems Architect/Requirements Management application programming interface (API).

The Systems Architect/Requirements Management API provides developers with a consistent and stable interface allowing programmatic access to the information within Systems Architect/Requirements Management. The developer has the option of implementing the API at a high level using Java™ or by using Tcl scripting.

- **Java Access:** The functions in the **RequirementService** class are available for use when writing programs in Java or JavaScript. The database contents can be read, modified, or updated programmatically without using the client GUI (graphical user interface). All functions that are available from the GUI can be accessed using Java, without the GUI interface.
- **Tcl Scripting Access:** The database can be read, modified or updated programmatically using Tcl (Tool Command Language) scripts while using the GUI. You can add activators if you have project administrators rights. These activators can be triggered on the occurrence of one or more events such as Create, After Delete, After Modify, Before Delete, and Before Modify on the BuildingBlocks, Folders, Notes, Requirements object types and their subtypes.
- **C# Access:** Server calls can be made from a COM client such as VBA (Visual Basic for Applications) macro using the C# API.

While both Java and Tcl provide full access to the Systems Architect/Requirements Management API, there are differences in how database transactions are handled. In Java, each API call runs in a separate transaction. In Tcl, all the API calls made within a Tcl script run in the same transaction. Because Tcl can run within a single transaction, it is the preferred method to use for iterative or large batch applications.

The two APIs can be used together very effectively in certain situations. A Java API caller can (through the `runActivator` function) run an activator's Tcl code. The Java program can remain in charge of an overall process, while handing off its iterative actions to Tcl running within the scope of a single transaction. This is much, much more efficient than making repetitive calls through the Java API.

The list of Java API functions is provided in the Javadocs. The Javadocs describe each function along with the response expected from the server. To browse a full list of functions, see the Javadoc HTML documentation, installed with Systems Architect/Requirements Management on the Web server. The exact URL may vary, depending on how Systems Architect/Requirements Management was installed in your web server, but the Javadoc URL will be similar to this:

[http:// ... /tcr/ugs/tc/req/apidoc/index.html](http://.../tcr/ugs/tc/req/apidoc/index.html)

The C# API provides an interface that allows making server calls from .Net code and an interface that exposes the API methods to COM.

Chapter 2: API Overview

This chapter contains standard input parameters and lists common API functions.

Standard Input Parameters

Each function takes a set of input parameters unique to the function. There are also several input parameters that apply to a wide range function calls. Table 2-1 lists the standard input parameters.

Table 2-1. Standard Input Parameters

Input	Description
User	Specifies the user session ID. This is the first parameter for almost all Java API methods. It uniquely identifies an existing (already logged in) user session. It consists of the user's login name concatenated with the HTTP session ID.
Properties	Specifies the property values to populate in returned data beans. This can consist of actual property names or one of the constants in com.edsplm.tc.req.databeans.DataBean (USER_PROPERTY, SYSTEM_PROPERTY, ALL_PROPERTY or FULL_PROPERTY).

Listing of API Methods

The complete list of API methods is shown in the table below. The table describes each method and indicates if the method is available to the Java API or the Tcl Scripting API.

Table 2-2. Available API Methods

Method	Description	Java	C#	Tcl
authenticateSSOUser	Validates a user name against single sign-on server using an SSO session key.	X		
authenticateUser	Validates a user name and password.	X		
calculateProperties	Calculates the numeric properties in objects.	X		X
changeApproval	Updates the status of change approval objects.			X
copyObjects	Copies the list of objects to the given destination.	X		X
createAction	Instruct Systems Architect/Requirements Management client to launch a URL or run a Macro.			X
createActionFileDownload	Downloads a file from the server to the Systems Architect/Requirements Management client workstation, and optionally, open it in an application. The C# method name is downloadfileCOM .		X*	X
createBaseline	Creates a baseline containing the specified objects.	X		X
createExternalLink	Creates a Teamcenter Interface (WOLF) link from Systems Architect/Requirements Management to an object in an external application such as Teamcenter Engineering or Teamcenter Enterprise.	X		X
createLinks	Creates Trace Links between fromID object and to each object in toIDs.	X	X	X
createObject	Creates a new object in the database as a LAST_MEMBER or FIRST_MEMBER or LAST_SIBLING or NEXT_SIBLING.	X	X	X

Table 2-2. Available API Methods

Method	Description	Java	C#	Tcl
createProject	Creates a new project in the database.	X		X
createShortcuts	Creates a new shortcut to the existing object. The C# method name is createShortcut .	X	X*	X
createUser	Creates a user object given user name. Also, sets the password and the user's maximum privilege for the database.	X		X
createVariant	Creates a variant of a versionable object.	X		X
createVersion	Creates a version of a versionable object.	X	X	X
deleteLinks	Deletes complying or defining traceLinks between fromID object and objects in toIDs.	X	X	X
deleteObjects	Deletes the list of objects and moves it to trash can of the caller.	X	X	X
displayMessage	Displays a message in the Systems Architect/Requirements Management client.			X
emptyTrashcan	Empties this users trash can by actually destroying the objects from the database.	X		X
export2Excel	Export an Microsoft Excel file.			X
exportDocument	Writes objects from the database to a Word, Excel, AP233, or XML file.	X	X	X
exportXML	Exports Systems Architect/Requirements Management schema objects or projects to an XML file.			X
getAdminMap	Gets the specified administration module folder from the database	X		
getEnvironment	Retrieves one of Architect/Requirements's configuration parameters.	X		X

Table 2-2. Available API Methods

Method	Description	Java	C#	Tcl
getList	Returns a list of Systems Architect/Requirements Management objects related to the given object.	X	X	X
getObject	Gets the given object (one) from the database with set of properties.	X	X	X
getObjects	Gets the given objects (one or more) from the database with set of properties.	X	X	
getProjects	Returns the list of projects you have access to and returns a specified set of properties for each object on the list.	X		X
getProperties WithFormula	Gives NumericPropertyDefinitions that have a formula attached for the passed objects.	X		X
getProperty Definition	Gets the property definition with the name propName from the project objID .	X	X	X
getProperty Definitions	Gets property definitions from the database.	X	X	X
getRemote ObjectTrace Report	Retrieves the trace report for the remote object.	X		X
getType Definition	Gets the type definition with the name propName from the project objId .	X		
getValue	Gets the value of a property from the given object.	X		X
importDocument	Imports a document of type XML, AP233, Excel, or Word. The C# method name is importFile .	X	X*	X
logoutSSOUser	Logs out the given user and closes the Systems Architect/Requirements Management and SSO sessions.	X		
logoutUser	Logs out the user whose user session ID is passed in.	X		

Table 2-2. Available API Methods

Method	Description	Java	C#	Tcl
moveObjects	Moves the list of objects to a destination object.	X	X	X
releaseObject	Releases reservation on an object.	X	X	
restoreFrom Trashcan	Restores objects from this user's trash can to the old owner.	X	X	X
runActivator	Runs the specified activator specified by Activator ID.	X	X	X
runActivator	Runs the specified activator, activator specified by activator name in the specified project.	X		X
runReport	Searches the database for objects that match the specified criteria and output them in a report file.	X	X	X
runScript	Runs a Tcl script.	X		
search	Return a list of objects matching the search parameters.			X
search	Searches the database for objects that match the specified criteria. The search is not case sensitive.	X	X	X
sendEmail	Tcl command class to send E-Mail to List of Email Ids.	X	X	X
setEnvironment	Sets one of Systems Architect/Requirements Management â€™s configuration parameters.			X
setObject	Sets the given object properties and the passed in value in the database the setObject requires NO response.	X		
setObject	Sets the given object properties in the database. If setObject requires a response, set a response.	X		X
setObjects	Sets the given objects' properties in the database.	X	X	
	Sets password for a user.	X		X

Table 2-2. Available API Methods

Method	Description	Java	C#	Tcl
setPassword				
setUser Preferences	Allows a user specify their location so information can be formatted appropriately.	X		X
setValue	Sets the value of a property for a given object. This function can be used to set one value of the object (unlike setObject that sets multiple values).			X
undo	Undoes the last database modification.	X	X	
uncoupleShortcuts	Uncouples a shortcut from the master object, creating a copy of the master object.	X		X
writeLog	Writes a message to the server log file.			X

* Indicates the C# method name is different. See description for the method.

Schema Object Display Name and their Actual Name

In some instances, the name displayed in the Systems Architect/Requirements Management client for out of the box schema objects, such as property and type names, does not match the name stored in the database. This is because the Systems Architect/Requirements Management client can map the actual name to something else. This mechanism is primarily intended for mapping the names to different languages. But some names are also mapped in English.

For example, the **Data Definition** and **Data Dictionary** type names are stored in the database with no space, **DataDefinition** and **DataDictionary**. However, for API calls, you must use the actual schema object name as the display name does not work. This can cause confusion when schema object names are used in API methods such as **createObject** and **setValue**. You can determine the actual name of a schema object by creating a macro with the content:

```
displayMessage [getValue $selected Name]
```

This displays the actual name of the selected schema object.

Chapter 3: Using the Java API Client Library to Access the API

This chapter contains information on using the Java API client library to access the API. If you are developing your client application using Tcl (Tool Command Language) scripting, you can use the Tcl client library included in the toolkit.

Using the API

The **RequirementService** class serves as Systems Architect/Requirements Management Enterprise API. **RequirementService** contains methods that provide complete access to the Systems Architect/Requirements Management server and database. All methods in **RequirementService** are static, so no instance of the class is required. The **RequirementService** class is located in the **com.edsplm.tc.req.enterprise** package. It is recommended that the Javadocs for **RequirementService** as well as for the package **com.edsplm.tc.req.databeans** be reviewed thoroughly.

A typical single transaction involves, obtaining a session ID, making a call using one of the API functions listed in **RequirementService**, obtaining a **ResultBean** from the returning API call, and checking to see if your transaction was successful by inspecting the **ResultBean** objects.

Logging In to the API

To access the Systems Architect/Requirements Management API you must login using the **AuthenticateUser** or **AuthenticateSSOUser** method. A session ID is constructed from the unique ID that is passed to a successful call to **AuthenticateUser** or **AuthenticateSSOUser**. The session ID takes the form of the login user name concatenated with the unique ID that was passed to **AuthenticateUser** or **AuthenticateSSOUser**. The unique ID is usually obtained by asking the J2EE server for the HttpSession ID. Once a successful login occurs, Systems Architect/Requirements Management keeps track of the user session identified by the session ID. From this session ID, all subsequent API calls must include the session ID as the first argument.

User sessions can be closed by calling **logoutUser** or **logoutSSOUser**. If a logout is not expressly called, the user session times out.

```
// Login to Teamcenter Requirements
ResultBean aBean = RequirementService.authenticateUser("username",
    "password", HTTPSessionID, null);
// Build session ID to use for each API call
String SessionID = "username" + HTTPSessionID;

// API calls...

RequirementService.logoutUser("username", HTTPSessionID);
```

Describing the API Functions

The API functions range from creating objects to creating users to searching. To browse a full list of functions, see the Javadoc HTML documentation, installed with Systems Architect/Requirements Management on the web server. The exact URL may vary, depending on how Systems Architect/Requirements Management was installed in your web server, but the Javadoc URL will be similar to this:

[http:// ... /tcr/ugs/tc/req/apidoc/index.html](http://.../tcr/ugs/tc/req/apidoc/index.html)

Describing ResultBeans

All the **RequirementService** API methods return a **ResultBean**. The result bean contains a flag indicating if the call succeeded. If no error occurred, the **ResultBeans** contain the desired return value for the API call, typically a **DataBean** or vector of **DataBeans** that represent Systems Architect/Requirements Management objects or strings. The **ResultBean** class is located in the **com.edsplm.tc.req.databeans** package.

In addition a result bean contains a message list, a change list, and a schema list.

Message List and Error Handling

The **MessageBean** class is located in the **com.edsplm.tc.req.databeans** package. A message bean contains the information for displaying a single error, warning, or information message to the user. The bean contains a unique tag name for the message which can be retrieved with the **getTag** method. The tag names are brief descriptions of the issue. For example, **PASSWORD_INCORRECT** tag is used when the wrong password is entered during login. The tag is converted to a readable message on the client using a language bundle. The message lookup service is not available to server APIs.

In addition to the tag, a message bean can contain substitutions and appended text. Substitutions are variable information included within a message. They can be retrieved with the **getSubstitutions** method. Appended text is displayed after the message. Substitutions are typically used to show the database objects involved. The **getAppendedText** method is used to get the information on the appended text.

A **ResultBean** contains a list of any messages generated during the API call. If an error occurs, the message list contains an error message to indicate the problem. In addition to the error message, the entire transaction is nullified.

```

// Get a list of all the TCR projects
ResultBean resBean = RequirementService.getProjects(SessionID,
    null, null);
If (resBean .isSuccess()) {
    // call succeeded, get the result
    Vector result = resBean.getResult();
} else {
    // call failed, get the error messages
    Collection messages = resBean.getMessageList();
    for (Iterator i = messages.iterator(); i.hasNext(); ) {
        MessageBean msg = (MessageBean)i.next();
        System.out.println(msg.getTag());
    }
}
}

```

Change List

For API calls that modify Systems Architect/Requirements Management objects, a change list is returned to indicate the new state of any modified objects. A data bean for each modified object is contained in the change list. The change list is useful for refreshing objects in a user interface. Because API callers commonly do not need the change list, performance can be improved by turning this feature off during login.

```

Properties properties = new Properties();
properties.setProperty(DataBean.SESSION_PROPS.RETURN_CHANGES, "false");
RequirementService.authenticateUser("username", "password",
    HTTPSessionID, properties);

```

Schema List

ResultBeans also may contain schema information describing the returned objects. The schema list is only returned for **getObject** and **getObjects** methods. The schema list contains data beans for property definitions. A property definition for each requested property is included. Property definitions give detailed information about a property such as the complete choice list for a choice property.

Database Transactions

Every requirement service call uses its own database transaction. When an error occurs, the transaction is cancelled and any changes are rolled back. If the transaction succeeds, changes are committed to the database. The undo method can be used to roll back the changes from prior requirement service calls. Since each requirement service call is a database transaction, transaction management can cause performance problems when large numbers of API calls are made.



When calling a large number of API methods, you should consider using the Tcl API. The Tcl API is an internal API so it can run as a single transaction.

Internal vs External Java API

There are two types of Java APIs, external and internal. Both external and internal APIs have almost exactly the same set of methods. However, their arguments differ slightly as every external call has to carry **sessionID** information whereas the internal calls occur inside a transaction where the session is already known and authenticated.

The **RequirementService** class is the external Java API. Every call is like a request coming from a Systems Architect/Requirements Management client. For each **RequirementService** call such as **getValue**, or **deleteObject**, the Systems Architect/Requirements Management server performs the following:

1. Validate the user **sessionID** passed.
2. Find and connect to an available session in the Versant session pool.
3. Run any before activators that are appropriate as required by the incoming call.
4. Perform the action of the **RequirementService** call.
5. Run any after activators, based on the objects that have changed.
6. Assemble a result bean with its change beans.
7. Commit the Versant transaction.
8. Release the Versant session.

You do not need to run steps 3, 5, and 7 for calls that don't modify anything; such calls also do not take much execution time. If the incoming **RequirementService** call is **runActivator**, or **runScript**, then that Tcl runs within step 4. Hence, many Tcl calls and actions are carried out within the scope of one server request and one transaction.

You can execute many actions in one transaction by using the Internal Java API, specifically the JavaAPI class. It is invoked through the **RequirementService.tcrEvaluate** method. It crosses the **RequirementService** boundary once and runs the evaluate method of the passed **TcrExecutable** object within the transaction. From a transaction standpoint, it is similar to calling the **runActivator** or **runScript**, except that it runs Java instead of Tcl. The internal Java API is essential for optimal performance when accessing a large volume of fine grained data with multiple **getValue** or **getList** calls and/or following relations to visit multiple objects. If you are performing a single action, like Export or Import, then calling the **RequirementService** directly is acceptable.

To use the Internal Java API you must define your own class that implements the **TcrExecutable** interface, which means that it must have an evaluate method. Your evaluate method can call any of the JavaAPI class methods. You must write one class with an evaluate method for each unique task that you want Systems Architect/Requirements Management to accomplish from Java. This is quite similar to writing individual activators for specific tasks in Tcl.

When you want to run one of these unique tasks, you must create an instance of the class and pass it in a **RequirementService.tcrEvaluate** call. In this way, all JavaAPI calls of the evaluate method occur inside one transaction. In addition to avoiding the overhead of crossing the **RequirementService** boundary multiple times, these calls also benefit from the cache built up as you touch different objects.

If the JavaAPI calls are modifying objects, you have the safety of the entire action completing successfully and committing the transaction, or rolling back the entire transaction if it fails. While using the **RequirementService** API to perform a set of interrelated actions, you have the risk of a later step failing that could leave the incomplete results from the first steps in the database.

The decision to use the Internal Java API or the Tcl API for carrying out a specified task depends on the language that the developer is most comfortable with. The Internal Java API is always the most efficient choice. Regardless of whether the JavaAPI class or Tcl API is called, most of the execution time is the same for the Systems Architect/Requirements Management code under both APIs that are called to do the object-level work.

Both the JavaAPI and RequirementService classes are covered in the Systems Architect/Requirements Management JavaDoc.

Figure 3-1 is an example of an internal Java API.

```
package com.teamcenter.example;

import com.edsplm.tc.req.databeans.DataBean;
import com.edsplm.tc.req.enterprise.JavaApi;
import com.edsplm.tc.req.enterprise.TcrExecutable;
import com.edsplm.tc.req.enterprise.TcrObject;
import java.util.Vector;

/**
 * Internal Java API Example
 *
 */
public class InternalJavaAPIExample implements TcrExecutable
{

    public InternalJavaAPIExample()
    {
        super();
    }

    public void evaluate( JavaApi javaApi, Vector aVector ) throws Exception
    {
        TcrObject project = null;
        project = javaApi.createProject("ProjectOne");
        if(project == null)
        {
            return;
        }

        //Create a folder and requirement
        TcrObject folder = null;
        TcrObject requirement = null;
        try {
            folder = javaApi.createObject("Folder1", project,
DataBean.TYPE.FOLDER, "");
            requirement = javaApi.createObject("Requirement1", folder,
DataBean.TYPE.REQUIREMENT, "");
        } catch (Exception e) {
            e.printStackTrace();
        }

        //Rename the folder and requirement
        javaApi.setValue(folder, DataBean.PROPERTY.NAME, "InternalJavaAPIs");
        javaApi.setValue(requirement, DataBean.PROPERTY.NAME, "FirstTask");
    }
}
```

Figure 3-1. InternalJavaAPIExample.java

```

//package com.teamcenter.example;

import com.edsplm.tc.req.databeans.MessageBean;
import com.edsplm.tc.req.databeans.ResultBean;
import com.edsplm.tc.req.enterprise.RequirementService;
import com.edsplm.tc.req.database.dbutils.*;
import com.edsplm.tc.req.enterprise.Config;
import java.util.Iterator;
import java.util.Vector;
import java.util.HashMap;

/**
 * Class for testing Internal Java API
 *
 */
public class TestInternalJavaAPI
{

    public static void main(String args[])
    {

        HashMap argMap = DefineSystem.parseArguments(args);
        String dbName = (String) argMap.get("-db");
        if ((dbName == null) || (dbName.equals(""))) {
            dbName = "TCR_db"; // deaefault value
        }

        DBManager dbm = new DBManager(dbName);
        System.setProperty(Config.DBNAME, dbName);

        InternalJavaAPIExample example = new InternalJavaAPIExample();
        Vector tcrObjects = new Vector();

        // As with all RequirementService calls there must be an active session for
        "username"
        //when making the tcrEvaluate call. Ordinarily that is done once at the beginning
        of a
        //user's session.
        // It is not necessary to have a separate authenticateUser call for each
        tcrEvaluate call.

        ResultBean loginResult =
        RequirementService.authenticateUser("username", "passwd", "");
        ResultBean result =
        RequirementService.tcrEvaluate("username", tcrObjects, example);
        if (! result.isSuccess())
        {
            // Get error message
            Iterator i = result.getMessageList().iterator();
            while (i.hasNext())
            {
                MessageBean msg = (MessageBean)i.next();
                System.out.println("Failed with error: " + msg.getTag());
            }
        }
    }
}

```

```
        }
        else
        {
            System.out.print("Successfully executed test");
        }
    }
}
```

Figure 3-2. TestInternalJavaAPI.java

Describing DataBeans

Systems Architect/Requirements Management objects are represented by **DataBeans**. A **DataBean** contains a table of the property values for the Systems Architect/Requirements Management object. The **DataBean** class also contains the constant values used in many API calls. When referring to these values, it is important to use the constants, instead of the actual values.

Java API Methods

The list of Java API functions is provided in the Javadocs. The Javadocs describe each function along with the response expected from the server.

Java API Examples

Figures 3-3, 3-4, and 3-5 work together to provide an example of selecting a project, a folder in the project, and creating a requirement in that folder. Figure 3-3 displays a drop down list of project names and allows you to select one.

```

<!-- This page is used for creating a requirement -->
<!-- This is the first page to be displayed -->
<!-- import statements -->
<%@ page import = "java.io.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.http.*" %>

<%@ page import = "com.edsplm.tc.req.enterprise.*" %>
<%@ page import = "com.edsplm.tc.req.databeans.*" %>
<!-- Select a project for the list box -->
<HTML>
  <HEAD>
    <H1><B>Select Project</B></H1>

  </HEAD>
<BODY>
<!-- select Folder is the second page to be displayed -->
<FORM METHOD=POST ACTION="/tcr/custom/selectFolder.jsp" >
<%
    HttpSession aSession = request.getSession();
/* if the user is not logged-in, the log-in page will be displayed */
if ( aSession.getAttribute( "loggedIn" ) == null ) {
    response.sendRedirect( "/tcr/custom/login.jsp?originalURL=" +
    request.getRequestURI() );
}
else {
    String aUserName = (String) aSession.getAttribute( "userplusid" );
%>
<%
    /*
    * Returns the list of objects the user has access to and a specified
    * set of properties for each object on the list.
    * To Gets the list of all projects in the database and their Name
    * property <BR> <CODE>
    * getList(userId,"",DataBean.LIST.PROJECT,new String[]
    * {DataBean.PROPERTY.LOID}); </CODE> <BR>.
    */
    ResultBean resPro = RequirementService.getList(aUserName, "",
        DataBean.LIST.PROJECT, new String[]
        {DataBean.PROPERTY.LOID}); %>
<% if (resPro.isSuccess() == false) { %>
    <!-- If no project is found -->
    <br> Did not find any projects.
    Successfully created the object= <%= resPro.isSuccess()
%>

<% } %>
<% /* Found at least one project */

    Vector dbProject = (Vector)resPro.getResult();
    Iterator idbProj = dbProject.iterator();

%>

```

```
Select a project from the list.  
<br>
```

Figure 3-3. SelectProject.jsp (Continued)

```

<Select Name="project" SIZE = 1 onChange="form.submit()">
<option VALUE = ""> </option>
<%
        /*      Gets the list of projects
                select the LOID and Name
                Populate the drop down list box
        */
        while (idbProj.hasNext()) {
            DataBean dbProj = (DataBean) idbProj.next();
                %>
            <option VALUE = <%= dbProj.getObjectId()%>>
            <%= dbProj.getName() %></option>

        <% }
        }%>
</Select>
<br>

</FORM>
</BODY>
</HTML>

```

Figure 3-3. SelectProject.jsp

Figure 3-4 displays a selectable list of folders from the project specified in figure 3-3. It also opens an editable text area to enter a requirements text.

```

<!-- This is the second page to be displayed to Creates a requirement -->
<!-- import statements -->
<%@ page import = "java.io.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.http.*" %>

<%@ page import = "com.edsplm.tc.req.enterprise.*" %>
<%@ page import = "com.edsplm.tc.req.databeans.*" %>

<HTML>
  <HEAD>
    <H1><B>Select Folder</B></H1>
  </HEAD>
<BODY>
<!-- createObject will be the next page to be displayed -->
<FORM METHOD=POST ACTION="/tcr/custom/createObject.jsp" >
<%
    HttpSession aSession = request.getSession();
    /* if the user is not logged-in, the login page will be displayed */
    if ( aSession.getAttribute( "loggedIn" ) == null ) {
        response.sendRedirect( "/tcr/custom/login.jsp?originalURL=" +
request.getRequestURI() );
    }
    else {
        String aUserName = (String) aSession.getAttribute( "userplusid" );
    }
%>
<% /*          This Enumeration is not required, To be removed in future */
    Enumeration enum = request.getParameterNames();
    // System.out.println( "enum: " + enum.hasMoreElements() );
    while (enum.hasMoreElements()) {

```

Figure 3-4. SelectFolder.jsp (Continued)

```

        String inputName = (String)enum.nextElement();%>
        <%String inputValue = request.getParameter(inputName);%>
    <% } %>
<%
// System.out.println("Project = " + request.getParameter("project"));
// System.out.println("UserName = " + aUserName);
    /*
     * Returns the list of objects the user has access to and a
specified set
     * of properties for each object on the list.
     * @param aUserName
     * @param request.getParameter("project") LOID representing the
object
     * @param DataBean.LIST.FOLDER name of the list. The list constants
     * are defined in DataBean.LIST class
     * @param new String[]{DataBean.PROPERTY.LOID} List of properties to
collect
     * for each object in the list.The DataBean
     * for each object will have the set of Properties defined in this
array.
     * The property constantsare defined in DataBean.PROPERTY
     * Vector of DataBean @Returns ResultBean The result data member of
the
     * ResultBean will have a objects.
    */
    ResultBean resPro = RequirementService.getList(aUserName,
        request.getParameter("project"),
        DataBean.LIST.FOLDER, new
String[]{DataBean.PROPERTY.LOID}); %>
        <!-- if no folder is found resPro.isSuccess() will be false -->
        <% if (resPro.isSuccess() == false) { %>
            <br>
                Did not find any projects.
                Successfully created the object= <%=
resPro.isSuccess() %>
        <% } %>

        <% Vector dbProject = (Vector)resPro.getResult();
            Iterator idbProj = dbProject.iterator();
        %>
        Select a folder from the list.
        <br>
        <Select Name="folder" SIZE = 1 >
        <option VALUE = ""> </option>
        <%
            /* Populate the drop down list box for the folders */
            while (idbProj.hasNext()) {
                DataBean dbProj = (DataBean) idbProj.next();
                %>
                <option VALUE = <%=
dbProj.getObjectId()%>>
        <%= dbProj.getName() %></option>
            <% } %>
        <% } %>

```

```
        </Select>
        <br>

<!-- Creates a text area for the input requirement -->

</TABLE><p>Enter Text</p>

<p> <textarea wrap=virtual name=bodyText cols=50 rows=10>
        </textarea>
<!-- Creates action buttons -->
<input type=submit value=Ok>
<input type=reset value=Reset>
<input type=button value=Cancel onClick=self.close() >
</FORM>
</BODY>
</HTML>
```

Figure 3-4. SelectFolder.jsp

Figure 3-5 creates a new requirement in the folder specified in figure 3-4 and sets the text to what was entered in above.

```

<!-- This is the 3rd and final page to be displayed when a requirement is created
-->
<!-- import statements -->
<%@ page import = "java.io.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.http.*" %>

<%@ page import = "com.edsplm.tc.req.enterprise.*" %>
<%@ page import = "com.edsplm.tc.req.databeans.*" %>

<HTML>
  <HEAD>
  </HEAD>
<BODY>
<FORM METHOD=POST >
<%
        HttpSession aSession = request.getSession();
/* If the user is not logged-in, the login page will be displayed */
if ( aSession.getAttribute( "loggedIn" ) == null ) {
        response.sendRedirect( "/tcr/custom/login.jsp?originalURL=" +
        request.getRequestURI() );
}
else {
        String aUserName = (String) aSession.getAttribute( "userplusid" );
%>
<%
%>
<% Enumeration enum = request.getParameterNames();
        while (enum.hasMoreElements()) {
                String inputName = (String)enum.nextElement();%>
                <%String inputValue = request.getParameter(inputName);%>
<% } %>

<%
        /**
         * Creates a new object in the database
         * @param aUserName The User Session ID of the user
         * @param request.getParameter("folder") the LOID of the owner
object
         * if the value of the position parameter
         * sibling (DataBean.POSITION) is FIRST_MEMBER or LAST_MEMBER;
         * or the LOID of the object if the value of the position parameter
is
         * NEXT_SIBLING or LAST_SIBLING.
         * @param "Requirement" The type of object to Createsin the
database.
         * This could be any of the constant type names defined in
DataBean.TYPE.
         * or the name of the UserTypeDefinition object
         * @param DataBean.POSITION.LAST_MEMBER Keyword specifying the

```

```
location
    * of the new object relative to ownerId. See DataBean.POSITION
    * for more details @Returns ResultBean. The result data member of
the
    * ResultBean will actually have
    * the DataBean of the object created.
```

Figure 3-5. CreateObject.jsp (Continued)

examples:

```

                                To Creates a Child Requirement on owner id (ex.
123.45.6789)
                                <CODE><BR>

createObject(userId,123.45.6789,DataBean.TYPE.REQUIREMENT,
                                DataBean.POSITION.LAST_MEMBER);
                                </CODE>
                                To Creates a next sibling Requirement on sibling
123.45.6789
                                <CODE><BR>

createObject(userId,123.45.6789,DataBean.TYPE.REQUIREMENT,
                                DataBean.POSITION.NEXT_SIBLING);
                                </CODE>
                                */
                                ResultBean res = RequirementService.createObject(aUserName,
                                request.getParameter("folder"), "Requirement", DataBean.POSITION.
                                LAST_MEMBER );
%>

<% /* if the object was not created res.isSuccess() will be false */
                                if (res.isSuccess() == false) { %>
<br>
                                The object was not created.
                                Successfully created the object= <%= res.isSuccess() %>
                                <br>
<% } else { %>
<%   DataBean db = (DataBean)res.getResult(); %>
<%
                                Set mySet = db.getPropertyKeySet();
                                Iterator itr = mySet.iterator();
%>

<% /* set the text on the newly created requirement */
                                ResultBean rb = RequirementService.setObject(aUserName,
                                db.getObjectId(),
                                new String[]{"Text"},
                                new String[]{request.getParameter("bodyText")}, 0, ""); %>
                                <!-- Display if the requirement text update was successful or not --
>
                                It was a success =<%= rb.isSuccess() %>
                                <br>
                                <!-- Creates an action button -->
                                <input type=button value=Close onClick=self.close() >

<% } %>

</FORM>
</BODY>
```

```
</HTML>  
<% } %>
```

Figure 3-5. CreateObject.jsp

Figures 3-6 and 3-7 provide an example of searching a project for objects that match specified search criteria.

Figure 3-6 displays a selectable list of projects and search options. You can select the object types, object name, text, and whether the search is case sensitive.

```

<%@ page import = "java.io.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.http.*" %>
<%@ page import = "com.edsplm.tc.req.enterprise.*" %>
<%@ page import = "com.edsplm.tc.req.databeans.*" %>

<!--
  This is the first page to be display for Text search.
  Next jsp pages to be activated is searchObject.jsp
-->
<FORM METHOD=POST ACTION="/tcr/custom/searchObject.jsp" >
<%
/*
If the user is not logged-in, the login form is presented.
*/
HttpSession aSession = request.getSession();
if ( aSession.getAttribute( "loggedIn" ) == null ) {
    response.sendRedirect( "/tcr/custom/login.jsp?originalURL=" +
    request.getRequestURI() );
}
else {
    String aUserName = (String) aSession.getAttribute( "userplusid" );
%>
<H1> Search for Object: </H1>
<%
    ResultBean resPro = RequirementService.getList(aUserName, "",
    DataBean.LIST.PROJECT, new String[]{DataBean.PROPERTY.LOID}); %>

<!-- ResultBean.isSuccess == true, at least a project was found
    ResultBean.isSuccess == false, no projects were found
-->
<% if (resPro.isSuccess() == false) { %>
    <br>        Did not find any projects.
    Successfully created the object= <%=
resPro.isSuccess() %>
    <% } %>
<!--
Iterate through getResult of the ResultBean
-->
<% Vector dbProject = (Vector)resPro.getResult();
    Iterator idbProj = dbProject.iterator();
%>
Select a project.
<br>
<!-- Provide a blank selection -->
<Select Name="StartObj" SIZE = 1 >
<option VALUE = ""> </option>
<!-- Populate the drop down list box with the list of projects -->
<%

```

```
while (idbProj.hasNext()) {  
    DataBean dbProj = (DataBean) idbProj.next();
```

Figure 3-6. Search.jsp (Continued)

```

                                %>
                                <option VALUE = <%=
dbProj.getObjectId()%>>

                                <%= dbProj.getName() %></option>
                                <% } }%>
                                </Select>
                                <br>

<%--
<P>Project ID or Folder where search should start<br><INPUT TYPE=text
NAME=StartObj
SIZE=30>
--%>
<br><br>
<P>Name<br><INPUT TYPE=text NAME=Name SIZE=30>
<br><br>
Content(Requirement & Note Only)<br><INPUT TYPE=text NAME=Content
SIZE=30></P>
<br>
<!-- Create the check boxes for Folder, Requirement and Note -->
<P><table>
    <tr>
        <td>
            <INPUT TYPE=checkbox NAME="Folder"
VALUE="Yes">
        </td>
        <td>
            Folder
        </td>
    </tr>
    <tr>
        <td>
            <INPUT TYPE=checkbox NAME="Requirement"
VALUE="Yes" checked>
        </td>
        <td>
            Requirement
        </td>
    </tr>
    <tr>
        <td>
            <INPUT TYPE=checkbox NAME="Note" VALUE="Yes">
        </td>
        <td>
            Note
        </td>
    </tr>
    <tr><td><br></td></tr>
<!-- Check box for the case sensitive search -->
<tr>
    <td>

```

```
        <INPUT TYPE=checkbox NAME="caseSensitive" VALUE="Yes">
    </td>
    <td>
        Case Sensitive
    </td>
</tr>
</table></P>
<!-- create the action buttons -->
<input type=submit value=Ok>
<input type=reset value=Reset>
<input type=button value=Cancel onClick=self.close() >
```

Figure 3-6. Search.jsp

Figure 3-7 performs the search and displays the results.

```

<!-- import statements -->
<%@ page import = "java.io.*" %>
<%@ page import = "java.text.*" %>
<%@ page import = "java.util.*" %>
<%@ page import = "javax.servlet.*" %>
<%@ page import = "javax.servlet.http.*" %>

<%@ page import = "com.edsplm.tc.req.enterprise.*" %>
<%@ page import = "com.edsplm.tc.req.databeans.*" %>

<%
    int x = 0;
    int k = 0;
    boolean caseSen = false;

    HttpSession aSession = request.getSession();
    /* if the user is not logged-in provide the login form */
    if ( aSession.getAttribute( "loggedIn" ) == null ) {
        response.sendRedirect( "/tcr/custom/login.jsp?originalURL=" +
request.getRequestURI() );
    }
    else {
        String aUserName = (String) aSession.getAttribute( "userplusid" );
        /* iterate through the request to Gets the list of the listed check boxes */
        Enumeration enum = request.getParameterNames();
        while (enum.hasMoreElements()) {
            String inputName= (String)enum.nextElement();%>
            <%String inputValue = request.getParameter(inputName);%>
            <%}
        %>

<!-- count the number of check boxes selected and assign this value to x -->
<% if (request.getParameter("Folder") != null && request.getParameter("Folder")
.equals("Yes")) {
    x = x+1; }
    if (request.getParameter("Requirement") != null && request.getParameter("Requirement")
.equals("Yes"))
        {x = x+1; }
    if (request.getParameter("Note") != null && request.getParameter("Note") .equals("Yes"))
    {
        x = x+1; }
    String[] SelectFrom = new String[x];
    /*select a an array of objects to select from. It could have Folder,
Requirement and Note.*/
    if (request.getParameter("Folder") != null && request.getParameter("Folder")
.equals("Yes")) {
        SelectFrom[k] = DataBean.TYPE.FOLDER;
        k = k+1;}%>

        <%if (request.getParameter("Requirement") != null && request.getParameter("Requirement")
.equals("Yes")
        ) {SelectFrom[k] = DataBean.TYPE.REQUIREMENT;
        k = k+1;}

```

```

    if (request.getParameter("Note") != null && request.getParameter("Note") .equals("Yes"))
    {
        SelectFrom[k] = DataBean.TYPE.NOTE;}
        /* If the search has to be case sensitive, assign true to caseSen. */
        if (request.getParameter("caseSensitive") != null &&
request.getParameter("caseSensitive")
            .equals("Yes"))
            caseSen = true;
    }%>
<% /*Search the selected object types for the text */
    for(int i= 0; i <SelectFrom.length; i++)
        System.out.println(SelectFrom[i]); %>

<% /*
    * Search the database for objects that match the specified criteria.
    * @param userName The sessionID + userName
    * @param request.getParameter("StartObj") The project or folder object ID
    * where search should start
    * @param request.getParameter("Name") The object name search criterion
    * @param request.getParameter("Content") The requirements content search
criterion
    * @param SelectFrom Objects types to search for (RequirmentDB, NoteDB,
FolderDB)
    * @param new String[]{"Text"} The desired properties to Returns for each
object
    * @param caseSen if true the search is case sensitive
    * @Returns Collection of DataBeans that match search criteria
    */
    ResultBean rb = RequirementService.search(aUserName,
request.getParameter("StartObj"),
    request.getParameter("Name"), request.getParameter("Content"), SelectFrom, new
String[]
        {"Text"} , caseSen);%>

        <% if (rb.isSuccess() == false) { %>
        <!-- Error in finding selected object -->
        <br>The text was entered successfully = <%= rb.isSuccess() %>
        <br>
        <input type=button value=Close onClick=self.close() >

    <% } else {%>
        <%
        /* No error, Match may or may not be found */
        if (rb.isSuccess() == true ) {
            Vector vdb = (Vector)rb.getResult();
            Iterator itr = vdb.iterator();
            %>

```

Figure 3-7. SearchObject.jsp (Continued)

```

<H2> Found : <%= vdb.size() %> </H2>
<TABLE BORDER=2>
<TR>
<!-- Column headings -->
<TD><b><i>Name </b> </i></TD>
<TD><b><i>Type</i></b></TD>
<TD><b><i>Text</i></b></TD>
<TD><b><i>Send email</i></b>
</TD>
</TR>
<TR>
<%
while (itr.hasNext()) {
    DataBean db = (DataBean) itr.next();
    %>
<TR>
<!-- Createslink to the text object -->
<TD><B><% if (db.getType().equals
(DataBean.TYPE.REQUIREMENT)) {%>
<a href="/tcr/custom/getText.jsp?loid=
    <%= db.getObjectId() %>
    " ><%= db.getName() %></a>
<% } else { %>
    <%= db.getObjectId()%>
<% } %>
</B></TD>
<TD>
<%= db.getType() %>
</TD>
<TD>
<%
/**
 * Gets the given object from the database with set
 * of properties
 * @param aUserName User Session id of the user
 * @param db.getObjectId() LOID of the object
 * @param new String[]{DataBean.PROPERTY.HTML}
 * List of properties to return.
 * Keyword can include
 * DataBean.ALL_PROPERTY/SYSTEM_PROPERTY/USER_PROPERTY
 * @param openForEdit True if the user is opening
 * the object with the
 * intent to modify the object. This will place
 * a reservation (lock) on the object
 * to prevent others from modifying the object.
 * The caller must use the
 * RequirementService.releaseObject method to
 * clear the reservation.
 * This flag is used only for Requirement and Note
 * objects and DataBean.PROPERTY.MHTML is in

```

```

value,
    * desiredProps.
        * For all other object types and property

    * this flag is ignored.
    * @Returns ResultBean The result data member of
    * the ResultBean
    * will have the DataBean for
    * the object and its desiredProps. The schemaList
    * data member
    * of the ResultBean will have
    * a data bean for Property Definitions
    * for each of the
    * property names in desiredProps. This
    * gives the caller all the necessary information
    * to edit the properties of this object.
    *
    */
    ResultBean res = RequirementService.getObject
    (aUserName, db.getObjectId(),new
    String[]{DataBean.PROPERTY.HTML},
    false);
    DataBean obj = (DataBean)res.getResult();
    %>
    <%= obj.get(DataBean.PROPERTY.HTML)    %>
    </TD>
    <TD>
    <a href="mailto:?subject=Approval is
    requested!&body=/tcr/custom/getText.jsp?
    loid=<%= db.getObjectId()%> Please click on the link
    to read the text.">
    Click here</a> to send an email for approval.

```

Figure 3-7. SearchObject.jsp (Continued)

```
        </TD>
</TR>
    <% } %>
<!-- close button -->
</TABLE><br><input type=button value=Close
onClick=self.close() >
<% } %>
<% } %>
                                <% } %>
```

Figure 3-7. SearchObject.jsp

Figure 3-8 provides an example of using the Teamcenter SSO for validation. The is **loginUsingTcSS.jsp** file is available in the **custom** directory of the **tcr.war** file.

```

<!-- This page is presented to the user when the user is not logged-in -->
<!-- Import Statements -->
<%@ page import="java.io.*,java.util.*" %>
<%@ page import="com.edsplm.tc.req.web.utils.ParamWebXml" %>
<%@ page import="java.text.*" %>
<%@ page import="javax.servlet.*" %>
<%@ page import="javax.servlet.http.*" %>
<%@ page import="com.edsplm.tc.req.database.API" %>
<%
String warFileName = API.getWarFileName();
%>

<html>
<head>
<title>Teamcenter 10 for systems engineering</title>
</head>
<body bgcolor="#ffffff" topmargin="10" leftmargin="10" MARGINWIDTH="10" MARGINHEIGHT="10"
>
<jsp:useBean id="validationBean" scope="page"
class="com.edsplm.tc.req.web.utils.CrossSiteValidationBean"/>
<script language= JavaScript>
</script>

<!-- Checking the authorization Error by getting the attribute value for
"TcR.AuthenticateError"
and checking for cross site validation -->
<%
String authError = (String) request.getSession().getAttribute("TcR.AuthenticateError");
if(!validationBean.isValidAgainstCrossSiteScripting(authError)) {
    response.sendRedirect("/") + warFileName + "/ugs/tc/req/CrossSiteScriptingThreat.jsp");
    return;
}

// Setting response header and ssoEnabled
response.setHeader("TcR.AuthenticateRequired", "TcR.AuthenticateRequired");
boolean ssoEnabled = false;

// Getting the ParamWeb value for SSO.Enabled
try {
    String temp = ParamWebXml.getConfigParam("SSO.Enabled",
getServletConfig().getServletContext());
    if (temp.equalsIgnoreCase("true")) {
        ssoEnabled = true;
    }
} catch (Exception e) {
    e.printStackTrace();
}

// If the SSO is enabled, authorization error is checked
if (ssoEnabled) {
    if (authError != null && authError.length() > 0) {
        <!-- Architect/Requirement is unable to authenticate user after SSO login -->

```



```

        <table cellpadding="0" cellspacing="8" border="0">
            <tr>
                <td><p><font color=red><b>Unable to authenticate user in
Teamcenter 10 for Systems Engineering.
                <p>Error: <%= authError %>.
                <p>Please contact your Teamcenter administrator</b></td>
            </tr>
        </table>
    </td>
</tr>
</table>
</body>
</html>

<%
    // ssoEnabled continued
        out.close();
        return;
    }

    // Get the SSO login URL
    String ssoLoginURL = ParamWebXml.getConfigParam("SSO.LoginURL",
getServletConfig().getServletContext());

    // If the SSOLogin URL is null, throw exception
    if (ssoLoginURL == null) throw new Exception("SSO.LoginURL param missing from
web.xml");

    // Else append the SSO login URL
    ssoLoginURL += "/weblogin/login_redirect";

    // Get the SSO AppID
    String ssoAppID = ParamWebXml.getConfigParam("SSO.AppID",
getServletConfig().getServletContext());
    if (ssoAppID == null) throw new Exception("SSO.AppID param missing from web.xml");

    // Get the URL TcR.LoginRequestURL
    String url = (String) request.getSession().getAttribute("TcR.LoginRequestURL");
    if(!validationBean.isValidAgainstCrossSiteScripting(url) ){
        response.sendRedirect("/") + warFileName +
"/ugs/tc/req/CrossSiteScriptingThreat.jsp");
        return;
    }

    int uriStart = -1;

    // Check for URL
    if( url != null){
        uriStart = url.indexOf("/tcr");
    }

```

```

// If URL is null, throw an exception
if (uriStart < 0) throw new Exception("Unable to determine SSO return URI. Request URL
is: " + url);
String ssoURI = url.substring(uriStart, url.length());

// Create a form and auto-post it to the SSO login URL
String launch_mode = request.getParameter("launch_mode");
if(launch_mode == null){
    launch_mode = "";
}
if(!validationBean.isValidAgainstCrossSiteScripting(launch_mode) ){
    response.sendRedirect("/") + warFileName +
"/ugs/tc/req/CrossSiteScriptingThreat.jsp");
    return;
}

String html = "<body onLoad=\"document['redirect'].submit()\" >"
+ "<form name=\"redirect\" action=\"\" + ssoLoginURL + "\" method=POST >"
+ "<input type=\"hidden\" name=\"TCSSOAPPID\" value=\"\" + ssoAppID + "\">"
+ "<input type=\"hidden\" name=\"TCSSORURI\" value=\"\" + ssoURI + "\">"
+ "<input type=\"hidden\" name=\"launch_mode\" value=\"\" + launch_mode + "\">";

// Pass the request parameters
java.util.Enumeration paramNames = request.getParameterNames();
while (paramNames.hasMoreElements()) {
    String paramName = (String) paramNames.nextElement();
    String paramValue = request.getParameter(paramName);
    if(!validationBean.isValidAgainstCrossSiteScripting(paramName) ||
!validationBean.isValidAgainstCrossSiteScripting(paramValue)) {
        response.sendRedirect("/") + warFileName +
"/ugs/tc/req/CrossSiteScriptingThreat.jsp");
        return;
    }
    html += "<input type=\"hidden\" name=\"\" + paramName + "\" value=\"\" +
paramValue + "\">";
}

html += "</form></body>";

out.println(html);
out.close();
return;
}

%>

```

Figure 3-8. loginUsingTcSS.jsp (Continued)

```

<script language= JavaScript>

function writeCookie() {
    var nextyear = new Date();
    var user = document.login.j_username.value;
    if (user == "") {
        alert("User Name field must not be blank");
        return false;
    }
    <!-- Encode the username and password -->
    var password = encodeURIComponent(document.login.j_password.value);
    document.login.j_password.value = password;
    document.login.j_username.value = encodeURIComponent(user);
    nextyear.setFullYear(nextyear.getFullYear()+1);
    document.cookie = "TcRLogin="+ escape(user) + ";
path=;/expires="+nextyear.toGMTString();
    return true;
}

function readCookie() {
    var allcookies = document.cookie;
    var pos = allcookies.indexOf("TcRLogin=");
    var value = "";
    if (pos != -1) {
        var start = pos + 9;
        var end = allcookies.indexOf(";", start);
        if (end == -1) end = allcookies.length;
        var value = allcookies.substring(start, end);
        value = unescape(value);
    }
    document.login.j_username.value = value
}

function setFocus() {
    if (document.forms[0].j_username.value.length > 0) {
    }
    else {
    }
}

function cancelLogin (queryParams)
{
    var url = '/<%=warFileName%>/ugs/tc/req/CloseApplication.jsp';
    if (queryParams != null && queryParams.length > 0)
    {
        url += ('?' + queryParams);
    }

    location.href = url;
}
</script>

```

Figure 3-8. loginUsingTcSS.jsp (Continued)


```

        </script>
    </tr>
    <tr>
        <td><nobr><strong><font face="Arial" size=2>User
Name:</font></strong></nobr></td>
        <td><input style="TCInput" type=text id="userID"
name="j_username" size="25"></td>
    </tr>
    <tr>
        <td><strong><font face="Arial"
size=2>Password:</font></strong></td>
        <td><input style="TCInput" type=password
name="j_password" value="" size="25"></td>
    </tr>
    <tr>
        <td valign=middle><b><font face="Arial" size=2>Language:
</font></b></td>
        <td valign=middle><SELECT name='LOCALE_PARAM'>
        <OPTION value="en_US" SELECTED>English (United States) - Default</OPTION>
</SELECT></td>
    </tr>
    <tr>
        <td colspan=2 align=right><input
align="middle" type="Submit" value=" Log In ">
        <%
        if (authError != null && authError.length() > 0) {

            StringBuffer url = request.getRequestURL();

            String queryStr = request.getQueryString();

            if(!validationBean.isInputValidAgainstCrossSiteScripting(url.toString()) ||
!validationBean.isInputValidAgainstCrossSiteScripting(queryStr)) {
                response.sendRedirect("/" + warFileName +
"/ugs/tc/req/CrossSiteScriptingThreat.jsp");
                return;
            }

            if (url != null &&
url.lastIndexOf("interface_login") != -1) {
                %>
                <input type="button" value=" Cancel Login "
                align="center"
                onclick="javascript:cancelLogin ('<%= queryStr %>')" />
                <%
            }
        }
        %>
    </td>
</tr>
</table>
</td>

```

```
</tr>
```

Figure 3-8. loginUsingTcSS.jsp (Continued)

```

    <%
    // Pass all the request parameters
    java.util.Enumeration paramNames = request.getParameterNames();
    while (paramNames.hasMoreElements()) {
        String paramName = (String) paramNames.nextElement();
        String paramValue = request.getParameter(paramName);
        if(!validationBean.isValidAgainstCrossSiteScripting(paramName) ||
            !validationBean.isValidAgainstCrossSiteScripting(paramValue)) {
            response.sendRedirect("/" + warFileName +
"/ugs/tc/req/CrossSiteScriptingThreat.jsp");
        }
    }
    %>
    <input type="hidden" name="<%= paramName %>" value="<%= paramValue %>" >
    <%
    } // end while
    %>

    </form>
</table>

</body>
</html>

```

Figure 3-8. loginUsingTcSS.jsp

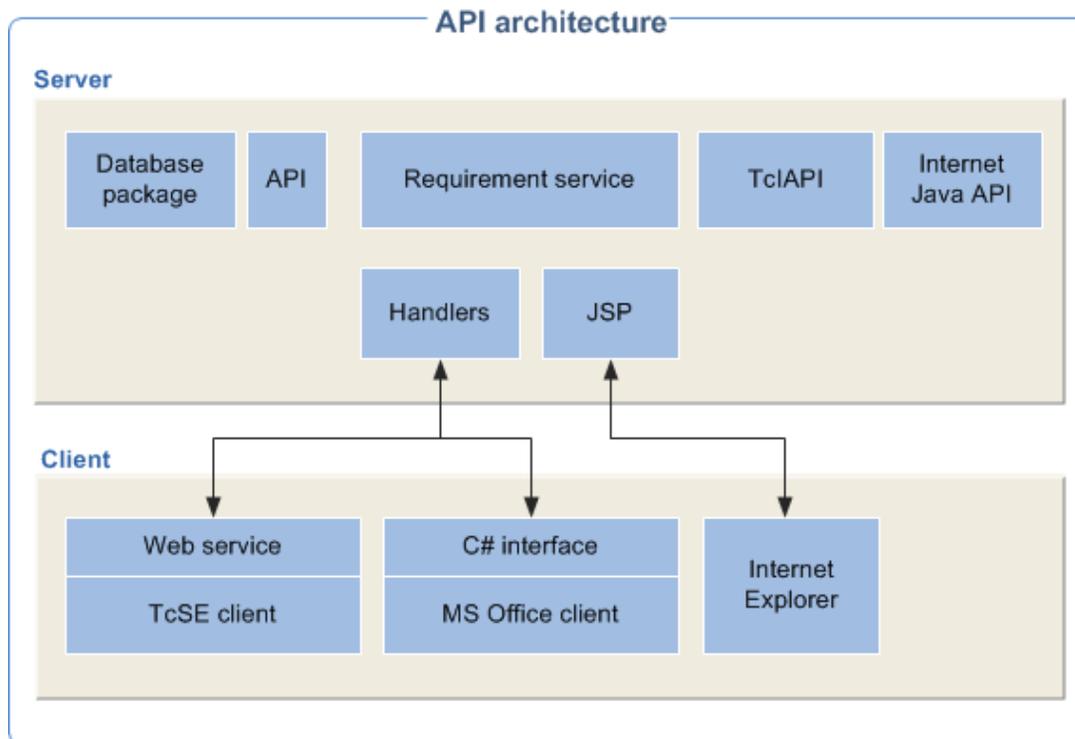
Chapter 4: Using C# API from COM (VBA)

This chapter contains information about using the C# API from COM (Component Object Model).

Introduction

The C# API methods provide functionality to make server calls from a COM client such as the VBA (Visual Basic for Applications) macro.

The following figure provides a graphical representation of the architecture of the Systems Architect/Requirements Management API.



The C# API contains methods to make Systems Architect/Requirements Management server calls. Interfaces are provided to make the calls using .NET, VBA or COM. The API methods are contained in

TcR.dll and **TcR.tlb** files. The **TcR.tlb** contains the information required to use procedures or classes in the **TcR.dll** file. The important components of API include:

ITcRConnection — **ITcRConnection** interface provides methods for making server calls from .NET code using VBA or COM.

ITcSEConnection — **ITcSEConnection** interface provides methods for making API calls using .NET and exposes the API methods to COM. **ITcSEConnection** is a wrapper around **ITcRConnection** to support COM-specific parameter format.

ResultBean — The API methods return a **ResultBean**. In addition to the requested data, the **ResultBean** contains status information about the call. This includes a flag indicating if the call succeeded and any error, warning or informational messages generated during the call. The success flag must be checked after each API call and error messages can be extracted and displayed. If no error occurs, the **ResultBean** contains the desired return value for the API call, typically a **DataBean** or list of **DataBean** that represent Systems Architect/Requirements Management objects.



Systems Architect/Requirements Management uses beans to return data from the Systems Architect/Requirements Management server to the client applications. In Java, a bean is a type of object that meets certain criteria. This includes the ability to be passed from a web server to a client application.

DataBean — A **DataBean** represents a single Systems Architect/Requirements Management object. A **DataBean** contains the name, type and unique database identifier (LOID) of an object. **DataBean** also contain a list of property names and values.

DataBean constants — There are a number of **DataBean** sub classes that contain keywords for use in API calls. For example, the **DataBean.LIST** class contains the legal values for the listType argument of the **getList** API method. When referring to these values, you must use these constants instead of the actual values.

For more information on on Systems Architect/Requirements Management's public API, see the online API manual.

Configuring COM Client (VBA)

This section specifies the prerequisites and describes the procedures to get started with programming in VBA, a COM client.

Prerequisites

- Systems Architect/Requirements Management Client must be installed.

Adding Reference to TcR.tlb

Adding reference to **TcR.tlb** enables a VBA macro to make calls to Systems Architect/Requirements Management C# API.

1. Open the Visual Basic Editor and choose **Tools**→**References**.
2. Browse to the location and select **TcR.tlb**.

Connecting to Systems Architect/Requirements Management

A connection to the Systems Architect/Requirements Management server is made using the **TcRConnectionService** class. You can connect to a Systems Architect/Requirements Management server through the Systems Architect/Requirements Management client or directly to the Systems Architect/Requirements Management server. **TcRConnectionService** first attempts to connect through the Systems Architect/Requirements Management client. If a Systems Architect/Requirements Management client is not running then a direct connection to the Systems Architect/Requirements Management server is used.

A connection through the Systems Architect/Requirements Management client uses the **TcRChannel** class. If the Systems Architect/Requirements Management client is not running or not configured (connection port is not specified), it connects to the Systems Architect/Requirements Management server directly using **TcRWebService**. You can use the **TcRChannel** class instead of **TcRConnectionService** class if you need to control the type of connection.



Siemens PLM Software recommends using **TcRConnectionService** to establish the connection. **TcRConnectionService** automatically selects the appropriate connection type.

To make a connection, the following information is required.

- Machine name of the Systems Architect/Requirements Management server
- Port used by the application server
- Valid Systems Architect/Requirements Management user ID and password

When connecting through the Systems Architect/Requirements Management client, you also need to specify a port number for communicating with the Systems Architect/Requirements Management client.

- Connect through **TcRConnectionService**

This is the preferred way to connect to Systems Architect/Requirements Management. It is used to obtain an instance of either **TcRChannel** or **TcRWebService**.

```
Dim connectionService As TcR.TcRConnectionService
Dim tcrConnection As TcR.ITcSEConnection
Set connectionService = New TcR.TcRConnectionService
connectionService.tcr_client_socket_port = "4000"
connectionService.tcr_client_socket_url = "tcseclient"
connectionService.tcr_server_controller_url = _

    "http://tcseserver:8080/tcr/controller/"
connectionService.user_name = "tcseuser"
connectionService.user_password = "password"

Set tcrConnection = connectionService.CONNECTIONCOM
If (tcrConnection.connectCOM()) Then
    If (tcrConnection.isConnectedCOM = CONNECTION_CLIENT) Then
        MsgBox "Connection to TcRChannel Successful"
    ElseIf (tcrConnection.isConnectedCOM = CONNECTION_SERVER) Then
        MsgBox "Connection to TcRWebService Successful"
    End If
Else
    MsgBox "Connection to TcR Failed"
End If
```

The following is the explanation of the parameters used in the above example:

Parameter	Definition
tcr_client_socket_port	<p>The port on which the Systems Architect/Requirements Management client listens and the data is communicated. For example, "4000"</p> <p>Do not set if the Systems Architect/Requirements Management client is not installed. The connection happens automatically.</p> <p>Ports available for use are specified in the PortRangeStart and PortRangeEnd web application parameters.</p> <p> A Microsoft Office application launched from the Systems Architect/Requirements Management client uses one of the ports making it unavailable.</p>
tcr_client_socket_url	<p>The machine name on which the Systems Architect/Requirements Management client is installed. For example, "tcsclient"</p> <p>Do not set if the Systems Architect/Requirements Management client is not installed. The connection happens automatically.</p>
tcr_server_controller_url	<p>The controller URL of the server. You must provide the URL of the server as a value to this parameter. For example, "http://tcseserver:8080/tcr/controller/".</p>
user_name	<p>The Systems Architect/Requirements Management user name. For example, "tcruser"</p>
user_password	<p>The password for the login user name provided as value to the user_name parameter.</p> <p>For example, "password"</p>

The above example indicates that the Systems Architect/Requirements Management client is installed from the **http://tcseserver:8080/tcr** server on the machine with name **tcsclient** and the client is running on the port **4000**.

You can determine the name of the machine dynamically (at the run time) using the following method:

```
Private Declare Function GetComputerName Lib "kernel32" _
    Alias "GetComputerNameA" _
    (ByVal lpBuffer As String, nSize As Long) As Long

Function ReturnComputerName() As String
Dim rString As String * 255, sLen As Long, tString As String
tString = ""
On Error Resume Next
sLen = GetComputerName(rString, 255)
sLen = InStr(1, rString, Chr(0))
If sLen > 0 Then
tString = Left(rString, sLen - 1)
Else
tString = rString
```

```

    End If
    On Error GoTo 0
    MsgBox UCase(Trim(tString))
    ReturnComputerName = UCase(Trim(tString))
End Function

```

- **Connect through TcRWebService**

This method is used to directly communicate with the Web server using the **TcRWebService** instance. This is an HTTP communication.

```

Dim webService As TcR.TcRWebService
Dim tcrConnection As TcR.ITcSEConnection

Set webService = New TcRWebService
webService.TcRControllerCOM = _
    "http://pni3p179:8080/tcr/controller/"
webService.user_idCOM = "test"
webService.passwordCOM = "password"
If (webService.connectCOM()) Then
    MsgBox "Connection to TcRWebService Successful"
Else
    MsgBox "Connection to TcRWebService Failed"
End If

```

- **Connect through TcRChannel**

This method is used to communicate through a socket with the Systems Architect/Requirements Management client. The Systems Architect/Requirements Management client then forwards the call to the Systems Architect/Requirements Management Web server. This is a socket service communication. As **TcRConnectionService** attempts to connect using **TcRChannel** first, avoid using **TcRChannel** directly to make a connection.

When connecting through the Systems Architect/Requirements Management client any modifications to database objects is refreshed in the client.



You must always use the **TcRConnectionService** instance to get the right connection type (**TcRWebService** or **TcRChannel**).

VBA Examples for Calling C# API Methods

This section provides a few examples for using C# API methods from VBA. For a list of all the methods, visit the Systems Architect/Requirements Management client launch page and click **API C# Doc**.

A working example of using the API methods from VBA and C# is provided with the Systems Architect/Requirements Management release package. You must have Microsoft Visual Studio 2010 installed to view the example from C#.

For more information on the examples, see [Appendix - Examples for using C# API's](#).

createObjectCOM

DESCRIPTION

Creates an object in TcR and sets the given properties. It is a wrapper around the **createObject** method.

SYNTAX

```
Public Function createObjectCOM ( _  
    owner As String, _  
    type As String, _  
    position As String, _  
    ByRef propertyNames As String(), _  
    ByRef propertyValues As String() _  
) As ResultBean
```

ARGUMENTS

- **owner (String)**: Object ID of the owner of the new object.
- **type (String)**: Object type defined in **DataBean.TYPE**.
- **position (String)**: Object position defined in **DataBean.POSITION**.
- **propertyNames (String ())**: String array for the property names.
- **propertyValues (String ())**: String array for the property values.

RETURNS

A **ResultBean** object from Systems Architect/Requirements Management.

EXAMPLE

```
`Define variables  
Dim propName(1) As String  
Dim propVal(1) As String  
Dim rBean As TcR.ResultBean  
  
`Set Variables  
propName(0) = "Name"  
propVal(0) = "MyReq1"  
  
Set rBean = tcrConnection.createObjectCOM("5.0.3131327",  
"RequirementDB", "LAST_MEMBER", propName, propVal)  
  
If (rBean.isSuccess) Then  
    MsgBox "Requirement created succesfully"  
End If
```

getObjectCOM

DESCRIPTION

Gets the properties of a given object. It is a wrapper around the **getObject** method.

SYNTAX

```
Public Function getObjectCOM ( _  
    objectId As String, _  
    ByRef propertyNames As String(), _  
    ByRef propsInFile As Boolean(), _  
    reqChange As Boolean _  
) As ResultBean
```

ARGUMENTS

- **objectId** (String): Object ID of the object.
- **propertyNames** (String ()): String array for the property names.
- **propsInFile** (Boolean ()): Boolean array indicating property names are to be set in a file.
- **reqChange** (Boolean): Boolean to indicate locking the object.

RETURNS

A **ResultBean** object from Systems Architect/Requirements Management.

EXAMPLE

```
`Define variables  
Dim propName(1) As String  
Dim propInFile(2) As Boolean  
Dim obj As Object  
Dim rBean As TcR.ResultBean  
  
`Set variables  
propName(0) = "Name"  
propInFile(0) = False  
propInFile(1) = False  
  
Set rBean = tcrConnection.getObjectCOM("1.0.183339",  
propName, propInFile, False)  
  
Set obj = rBean.getResult()  
MsgBox obj.getValue("Name")
```

setObjectsCOM

DESCRIPTION

Sets the object properties in Systems Architect/Requirements Management. It is a wrapper around the **setObjects** method.

SYNTAX

```
Public Function setObjectsCOM ( _  
    ByRef databeans As DataBean() _  
) As ResultBean
```

ARGUMENTS

- databeans (DataBean ()): Array of **DataBean** objects.

RETURNS

A **ResultBean** object from Systems Architect/Requirements Management.

EXAMPLE

```
`Define variables  
Dim dBArray(1) As DataBean  
Dim dBean As New TcR.DataBean  
Dim rBean As TcR.ResultBean  
  
`Set variables  
dBean.objectId = "1.0.151747"  
dBean.setValue "Name", "MyReq123"  
  
Set dBArray(0) = dBean  
  
Set rBean = tcrConnection.setObjectsCOM(dBArray)
```

createLinksCOM

DESCRIPTION

Creates trace links with the given type in Systems Architect/Requirements Management. It is a wrapper around the **createLinks** method.

SYNTAX

```
Public Function createLinksCOM ( _  
    from As String, _  
    ByRef to As String(), _  
    linkType As String, _  
    subType As String _  
) As ResultBean
```

ARGUMENTS

- `from (String)`: Object ID to indicate the beginning of links.
- `to (String ())`: String array for object IDs to indicate the end of links.
- `linkType (String)`: String for the trace link type.
- `subType (String)`: Subtype of link to create. If null, the base type is used.

RETURNS

A **ResultBean** object from Systems Architect/Requirements Management.

EXAMPLE

```
`Define variables  
Dim toLink(0) As String  
Dim rBean As TcR.ResultBean  
  
`Set Variables  
toLink(0) = "1.0.151747"  
  
Set rBean = tcrConnection.createLinksCOM("1.0.151761", toLink, "Defining", "")
```

getResultCOM

DESCRIPTION

Gets the output from the result bean. It is a wrapper around the **getResult** method.

SYNTAX

```
Public Function getResultCOM ( _  
    output As Object, _  
    databeans to As DataBean() _  
)
```

ARGUMENTS

- `output (Object)`: Result if the return type is String or DataBean, else null.
- `databeans (DataBean())`: Array of DataBean collection if the return type is DataBean collection, else null.

RETURNS

None.

EXAMPLE

```
`Define variables  
Dim toLink(1) As String  
Dim rBean As TcR.ResultBean  
Dim output As Object  
Dim beans() As DataBean  
  
`Set Variables  
toLink(0) = "1.0.151747"  
Set rBean = tcrConnection.createLinksCOM("1.0.151761", toLink, "Defining", "")  
rBean.getResultCOM output, beans
```

In this example, the **createLinksCOM** method returns a result bean that contains a collection of DataBean, which is now available in the beans array.

getPropertiesCOM

DESCRIPTION

Gets the property names and values from the DataBean.

SYNTAX

```
Public Function getPropertiesCOM ( _  
    name As String(), _  
    value to As String() _  
)
```

ARGUMENTS

- name (String()): String array for the property names present in the DataBean.
- value (String()): String array for the property values present in the DataBean.

RETURNS

None.

EXAMPLE

```
`Define variables  
Dim toLink(1) As String  
Dim rBean As TcR.ResultBean  
Dim name As String()  
Dim value() As String()  
Dim dBean As DataBean  
  
`Set Variables  
toLink(0) = "1.0.151747"  
Set rBean = tcrConnection.createLinksCOM("1.0.151761", toLink, "Defining", "")  
rBean.getResultCOM output, beans  
Set dBean = beans(0)  
dBean.getPropertiesCOM name, value
```

In this example, the **getPropertiesCOM** method returns two arrays, one array containing property names and the other array containing the values for the property names.

NOTES

Similarly, you can use the **getChangeFlagsCOM** method on DataBean to get the change flags.

getDataBeansCOM

DESCRIPTION

Gets the data beans from the DataBean dictionary.

SYNTAX

```
Public Function getDataBeansCOM ( _  
    beans As DataBean() _  
)
```

ARGUMENTS

- `beans (DataBean())`: Data beans from the DataBean dictionary.

RETURNS

None.

EXAMPLE

```
`Define variables  
Dim toLink(1) As String  
Dim rBean As Tcr.ResultBean  
Dim rd As DataBeanDictionary  
Dim beans() As DataBean  
  
`Set Variables  
toLink(0) = "1.0.151747"  
Set rBean = tcrConnection.createLinksCOM("1.0.151761", toLink, "Defining", "")  
Set rd = rBean.getResultDictionary()  
rd. getDataBeansCOM beans
```

In this example, the **getDataBeansCOM** method returns an array (beans) with all the data beans from the DataBean dictionary.

runActivatorCOM

DESCRIPTION

Run an activator's Tcl script.

SYNTAX

```
Public Function runActivatorCOM ( _  
    activatorID As String, _  
    ByRef selectedIds As String() _  
    ) As ResultBean
```

ARGUMENTS

- `activatorID (String)`: Object ID of the activator.
- `selectedIds (String ())`: String array of Systems Architect/Requirements Management object IDs or other values. This array is available in Tcl as a global list named "selected".

RETURNS

A ResultBean object from Systems Architect/Requirements Management. The result value is the string specified in the Tcl return statement.

EXAMPLE

```
`Define variables  
Dim selectedIds(1) As String  
Dim rBean As TcR.ResultBean  
Dim tclReturnValue As Object  
  
`Set Variables  
selectedIds (0) = "Hello"  
`Run the activator named Hello World in MyProject  
Set rBean = tcrConnection.runActivatorCOM("\\MyProject\Activators\HelloWorld",  
    selectedIds)  
Set tclReturnValue = rBean.getResult()
```


Chapter 5: Using the Tcl Scripting API to Access the API

This chapter contains information on using the Tcl API client library to access the API. If you are developing your client application in Java, you can use the Java client library included in the toolkit.



The list of Java API functions is provided in the API Javadoc. The API Javadoc describes each function along with the response expected from the server. Additionally, the list of Property Names, Lists, Keywords and other Constants are defined in the `DataBean` class in API Javadoc.

You can access the Javadoc from Systems Architect/Requirements Management home page.

1. Click **API Javadoc**.
2. Click the **com.edsplm.tc.req.databeans** package link.
3. From the Class Summary list, click the **DataBean** link.

The constants are defined in a HTML table.



The list of Java API functions is provided in the API Javadoc. The API Javadoc describes each function along with the response expected from the server. Additionally, the list of Property Names, Lists, Keywords and other Constants are defined in the `DataBean` class in API Javadoc.

You can access the Javadoc from Systems Architect/Requirements Management home page.

1. Click **API Javadoc**.
2. Click the **com.edsplm.tc.req.databeans** package link.
3. From the Class Summary list, click the **DataBean** link.

The constants are defined in a HTML table.

Introduction

Tcl (*Tool Command Language*) is an interpreted language, originally developed by Professor John Ousterhout at the University of California, Berkeley. The Tcl interpreter is freely available on the Internet. Tcl has been ported to most popular operating systems (Microsoft Windows, many UNIX variants, and Macintosh) and hardware platforms.

Tcl is similar to other scripting languages like the Bourne shell, C shell, and Perl. These scripting environments let you execute other programs, providing enough programmability to integrate existing tools into a new tailored application that fits your needs. As an internal macro-like language, Tcl lets you create small applications that automate routine sets of commands.

Systems Architect/Requirements Management includes JACL, a Tcl interpreter written in Java. JACL supports the basic Tcl language (Tcl version 8.0), but does not support Tk, the graphical user interface companion to Tcl.

There are many Internet resources for additional Tcl/Tk information. This is a good starting place:

<http://www.tcl.tk>

There are number of Tcl books available in bookstores that carry a good selection of computer related technical books. A partial list follows:

- *Practical Programming in Tcl and Tk*, Brent Welch; Prentice Hall
- *Tcl and the Tk Toolkit*, John K. Ousterhout,; Addison-Wesley
- *Tcl/Tk in a Nutshell*, Paul Raines & Jeff Tranter, O'Reilly & Associates
- *Tcl/Tk Pocket Reference*, Paul Raines, O'Reilly & Associates

A wide variety of Tcl extensions are freely available in the public domain. However, these have not been tested with Systems Architect/Requirements Management and are not supported in any way by Siemens PLM Software.

Executing Tcl Scripts

Tcl scripts can be executed in two ways:

- Tcl scripts stored in activator objects can be triggered when a specified event occurs. (See chapter [Using Activators](#).)
- Java API clients can execute Tcl scripts using the **RequirementService** methods **runActivator** and **runScript**.

Transaction Management

Tcl scripts run in a single database transaction. If an error occurs, all changes made by the script are rolled back. When activators are triggered by a **RequirementService** call, the activators run in the same transaction as the API method. So if an error occurs in any activator, the changes made by all the activators and the API call are rolled back.

Parameter Types

In addition to the standard Tcl parameter types, Systems Architect/Requirements Management Tcl has these addition types:

- **OBJECT**: TcR database object identifier.
- **OBJECT_LIST**: A Tcl list of database object identifiers.

A Systems Architect/Requirements Management database object can be identified by its unique database ID (LOID). Objects in the administration module may also be identified by name. The following naming convention is used:

\\projectName\administrationFolderName\objectName

For example, the Requirement type definition in Project1 can be identified with:

```
\\Project1\Type Definitions\Requirement
```

The *administrationFolderName* in this syntax refers to the special predefined folders that appear just beneath a project in the administration module. Subfolders below that level are not recognized in this syntax. For example, if within the **Activators** folder there is a **Macros** folder, and under that a **Get Input** macro, its name would be *\\projectName\Activators\Get Input*, without the Macros level.

Listing of Tcl Methods

The sections that follow detail Tcl methods available to the Tcl Scripting API.

calculateProperties

DESCRIPTION

Calculates the numeric properties in objects. The values are calculated from all member objects. If it is a tree, it is calculated bottom up, using a formula at every parent object, recursively.

SYNTAX

```
calculateProperties objects properties
```

ARGUMENTS

- `object`: OBJECT_LIST – objects selected for calculation.
- `properties`: OBJECT_LIST – numeric properties, which has formula.

RETURNS

Void, the calculated values are directly set on the objects.

EXAMPLES

```
calculateProperties $objectORobjectList $properties
```

changeApproval

DESCRIPTION

This method is used to update the status of a change approval object. This method is used to submit a requirement or building block for approval or update the status of an existing change approval object. It can be used to force a change of change approval status of an object already being routed. This method is independent of the four out-of-the-box activators shipped with Systems Architect/Requirements Management, Change Approved, Change Rejection, Change Response, and Change Submit.

SYNTAX

```
changeApproval objectId action comment subject
```

ARGUMENTS

- **objectId**: OBJECT_LIST – The object ids (LOID) of the change approval objects. The object ids (LOID) of the change approval objects, or (when the action is Change Submitted) the LOIDs of the objects being submitted for approval.
- **action**: STRING_LIST – List of string identifying actions for the objectId (**Change Rejected**, **Change Approved**, and **Change Submitted** are the only valid actions. Use only one of these actions). This list has a one to one correspondence with objectId.
- **comment**: STRING_LIST – List of string identifying user comments for the objectId. The comment applies to this change approval object by this user. This list has a one to one correspondence with objectId.
- **subject**: STRING – The subject line of the email.

RETURNS

The updated objects.

NOTES

The lists (objectId, action, and comment) must have the same number of entries. The **changeApproval** call fails if multiple objectId and one action and comment are passed.

EXAMPLES

```
#get loid of the Change Approval Object.  
set objectId [list $currentObject]  
  
set action [list "Change Rejected"]  
  
set comment [list "Comment for $objectId"]  
  
changeApproval $objectId $action $comment $subject
```

copyObjects

DESCRIPTION

Copies the source objects to the specified destination. In addition to the source objects, all their descendants, owned objects and links are also copied.

SYNTAX

```
copyObjects sources destination deep
```

ARGUMENTS

- `sources`: OBJECT_LIST – objects to copy.
- `destination`: OBJECT – owner of copies.
- `deep`: BOOLEAN – If true, copies all descendants of the sources. If false, copies the source objects only.

RETURNS

OBJECT_LIST—The copied objects.

NOTES

Some operations are performed using copy and paste in the Architect/Requirements client but are not possible using the copyObjects API.

For example, you can copy and paste an object into a group in the Architect/Requirements client, but you must use the createLinks method when using the API.

EXAMPLES

```
set copies [copyObjects $requirements $folder]
```

createAction

DESCRIPTION

Allows macros or activators executed on Systems Architect/Requirements Management servers to initiate Systems Architect/Requirements Management client actions, for example, launch workstation applications.

`createAction FileDownload` allows macros or activators executed on Systems Architect/Requirements Management servers to download a file to the Systems Architect/Requirements Management client workstation, and optionally open it in an application. See [createAction FileDownload](#).

SYNTAX

```
createAction actionClass [parameter or list_of_parameters]

createAction LaunchWebBrowser urlString
createAction LaunchWebBrowser [list urlString sessionVarName sessionID]
createAction RunMacro [list projId macroName selected]
createAction RunReport [list testReport startObjectID]
createAction RunReport [list testReport]
createAction GoToAndOpen objectId
createAction GoToAndOpen [list objectId OpenCommand]
createAction GoToAndOpen [list objectId OpenCommand readOnly]
createAction GoToAndOpen [list objectId OpenCommand readOnly moduleName]
createAction Refresh
```

ARGUMENTS

- `actionClass`: **STRING** – Identifies the client action to be taken.
 - `LaunchWebBrowser`: Open the given URL in a browser.
 - `RunMacro`: Run the named macro from the given.
 - `RunReport`: Run saved reports from Macros or Activators.
 - `GoToAndOpen`: Navigate to an object and optionally open it.
 - `Refresh`: Refreshes the client as though the user pressed the **Refresh** button on the client toolbar.
- `urlString`: **STRING** – URL to open in the browser.
- `sessionVarName`: **STRING** – optional mechanism to allow the Systems Architect/Requirements Managements client to pass a `sessionID` to a JSP, avoiding the need for the JSP to perform its own Teamcenter Requirements login. This is the name of an argument to pass to the JSP. The Systems Architect/Requirements Managements client appends an argument by that name to the URL before sending it to the web browser.
- `sessionID`: **STRING** – optional mechanism for the JSP to obtain the argument to be used for the `sessionID` `sessionVarName` above. The value obtained (from the Systems Architect/Requirements Management server) for `sessionID` is assigned to `sessionVarName`.
- `projId`: **STRING** – `objectId` (LOID) for the project in which the macro is found.
- `macroName`: **STRING** – The name of a Macro to run in that project.
- `selected`: **STRING** – This value is set as the `selected` global Tcl variable when the macro runs. This argument replaces the normal value of `selected`, which is a list of the selected objects in the

user interface. This argument is optional. If it is not provided, the normal value for `selected`, the selected objects, is used.

- `testReport`: **STRING** – The name of the report.
- `startObjectID`: **STRING** – start object for search.

If there is no `startObjectID` specified, the action class uses the currently selected object or saved location as start object. The action class displays all reports in the project and lets the user pick one if the specified report does not exist in the project.

- `objectId`: **STRING** – The ID for the object to become selected.
- `OpenCommand`: **STRING** – Keyword that selects whether to simply go to the object, or to go to the object and open it.

To go to the object and open it, use `GoToAndOpen`. The object is opened in the application that is used when the `Open` command is used on the object.

To go to the object without opening it, use `GoTo`.

If `OpenCommand` is not specified, the default is `GoTo`.

- `readOnly`: **STRING** – `true` if the object to be opened should be opened in read-only mode. Otherwise, `false`.

If `readOnly` is not specified, the default is `false`.

- `moduleName`: **STRING** – Identifies whether to go to the Administration Module or the User Module.

Use `tcAdmin` for the Administration module.

Use `tcR` for the user module.

If `moduleName` is not specified, the default is `tcR`.

RETURNS

None

NOTES

Calling **createAction** adds information to the transaction result that requests certain actions to occur at the Systems Architect/Requirements Management client. These actions occur only when the client initiates the request. If the action came from Systems Architect/Requirements Management's Excel Live or Visio Live, or from a JSP calling the Requirements Service API, the requested action does not occur. For example, a **Create** activator might call **createAction RunMacro** to prompt the user for additional information on the newly created object. If the new object is created through Excel or Visio, the macro will not be run.

EXAMPLES

There are two ways to call the **LaunchWebBrowser** action class.

- Launch Internet Explorer and navigate to a specified URL:

```
set url http://www.ugs.com
createAction LaunchWebBrowser $url
```
- Launch Internet Explorer and pass in the user name and session to a Systems Architect/Requirements Management custom JSP in order to avoid login when accessing Systems Architect/Requirements Management objects:

```
set url "http://tcr_server_url:8080/tcr/.../xxx.jsp"
set jspParmName tcrSessionId
createAction LaunchWebBrowser [list $url $jspParmName]
```

The `userplusid` is the parameter name for user id and session. The jsp should use the parameter value. To avoid this from happening login as follows:

```
String tcrID = request.getParameter("tcrSessionId");
ResultBean result = RequirementService.getList(tcrID, null,
DataBean.LIST.PROJECT, new String[] {DataBean.ALL_PROPERTY}, 1, 0);
```

- The following is an example for the **RunMacro** action class:

```
createAction RunMacro [list $currentProject macroName]
```

where `macroName` is an existing macro name in the current project.

- The following is an example for the **GoToAndOpen** action class:

```
createAction GoToAndOpen [list $myObject GoToAndOpen]
```

This example navigates to the object `myObject`, and opens the object for edit.

- The following is an example for the **GoToAndOpen** action class:

```
createAction GoToAndOpen [list $myActivator GoToAndOpen tcrAdmin]
```

This example navigates to the object `myActivator` in the Administration module and opens the object for edit.

The following is an example of how to obtain the `sessionID` from the server in your macro or activator:

```
set sessionID [setEnvironment SessionID]
createAction LaunchWebBrowser [list urlString sessionVarName $sessionID]
```

createAction FileDownload

DESCRIPTION

Allows macros or activators executed on Architect/Requirements servers to download a file to the Architect/Requirements client, and optionally open it in an application. Also allows optionally opening a local client file without downloading from the server.

SYNTAX

```
createAction FileDownload [list serverFile app localFilePath isLive]
```

ARGUMENTS

- `serverFile`: **STRING** — (required) The following options are available:
 - The path of the file on server that the user wants to download.
 - The keyword `OpenLocalFileOnly`. When that keyword is used, the file is not downloaded from the server, but already exists on the client machine.
- `app`: **STRING** — The following options are available:
 - If this parameter is not present, the file is downloaded and a message is shown providing the location. No application is launched.
 - If the value of this parameter is keyword, **Default**, the downloaded file is opened in Internet Explorer.
 - If the value of this parameter is keyword, **MS_EXCEL** or **EXCEL**, the file is opened in Microsoft Excel.
 - If the value of this parameter is keyword, **MS_WORD**, the file is opened in Microsoft Word.
 - If the value of this parameter is keyword, **VISIO**, the file is opened in Microsoft Visio.
 - In all other cases, the file is opened using the user-specified application. The complete path name to the executable file must be specified.
 - If the value of this parameter is keyword, **NO**, then the file is not opened, and the user is not notified that file download has successfully completed.
- `localFilePath`: **STRING** — The local file path. If this parameter is missing or contains a "" string, a temporary file is created.

If this parameter is a simple file name, then that file name is used when writing to a temporary path. The temporary path is the same path that is retrieved when using the **ClientJavaAPI.getTempDir** function, which can be called from external Java code (run via **createAction RunJava**).
- `isLive`: **STRING** — This is `true` if the file is to be opened live. Otherwise the value is `false`. If this parameter is missing or contains a "" string, the value of `false` is used.

RETURNS

None

NOTES

If the **localFilePath** argument is omitted and the default **TcSE-nnnn** temp file name is used, the client file gets deleted when the user logs out.

EXAMPLES

- When the activator runs, the file **D:/XYZ.ppt** is downloaded from the server and is opened using Internet Explorer:

```
set serverFile "D:/XYZ.ppt"
set application Default
createAction FileDownload [list $serverFile $application]
```

- When the activator runs, the file **D:/XYZ.ppt** is downloaded from the server and is opened using the user-selected application, PowerPoint:

```
set serverFile " D:/XYZ.ppt "
set application {C:\win32app\Microsoft
  Office\Office10\POWERPNT.EXE}
createAction FileDownload [list $serverFile $application]
```

- When the activator runs, the file **D:/XYZ.ppt** is downloaded from the server and is opened using the user-selected application, Excel:

```
set serverFile " D:/XYZ.ppt "
set application "EXCEL"
createAction FileDownload [list $serverFile $application]
```

- When the activator runs, the file **D:/XYZ.vsd** is downloaded from server, copied to **D:/ABC.vsd**, and it is opened using the user-selected application, Visio:

```
set serverFile "D:/XYZ.vsd"
set clientFile "D:/ABC.vsd"
set application {C:\win32app\Microsoft Office\Visio11\VISIO.EXE}
createAction FileDownload [list $serverFile $application $clientFile]
```

- Or the application keyword can be used: When the activator runs, the file **D:/XYZ.vsd** is downloaded from server, copied to **D:/ABC.vsd**, and it is opened using the user-selected application, Visio:

```
set serverFile "D:/XYZ.vsd"
set clientFile "D:/ABC.vsd"
set application "VISIO"
createAction FileDownload [list $serverFile $application $clientFile]
```

- When the activator runs, no file is downloaded from server, local file **D:/ABC.mht** is opened Live using the user-selected application, Excel:

```
set serverFile " OpenLocalFileOnly"
set clientFile "D:/ABC.mht"
set application "EXCEL"
set isLive "true"
createAction FileDownload [list $serverFile $application $clientFile
  $isLive]
```

createAction RunJava

DESCRIPTION

Allows identification of Java classes and execution of methods from those classes in activators, macros, or custom menu items.



- **createAction RunJava** executes your Java code in the Architect/Requirements client process. Faulty code can cause the client to freeze, abort, consume excessive memory, or become unstable. The burden is on the Java developer to avoid such consequences.
- Although **createAction RunJava** runs Java code in the Architect/Requirements client, this is not intended as a customization point to add or modify client behavior, and there is no client Java API. This is intended as an interface point to aid in integrating Architect/Requirements with other client applications, and is supported only in that context.

SYNTAX

```
createAction RunJava [list className methodName]
createAction RunJava [list className methodName requiresOfficeLive]
createAction RunJava [list className methodName requiresOfficeLive parameter1
parameter2 ...]
```

ARGUMENTS

- `className`: **STRING** — The name of the class containing the method to run.
- `methodName`: **STRING** — The name of the method in the class to be run.
- `requiresOfficeLive`: **STRING** — `true` if Office Live installation is required to run the method. Otherwise, `false`.

The Java file may be a wrapper around C# code that is communicating with the client through the socket, and thus may require that the Office Live interface be installed to function properly. If this parameter is not used, the value is `false`.

- `parameter1`, `parameter2`, *etc.*: **STRING** — Values to be passed to the Java method being called.

RETURNS

None

NOTES

- To run methods from the **.jar** file, the class must always include a method with this signature:

```
public String connectTcSE(String controllerPath, String serverIP,
String socketServicePort, String sessionID)
```

This method provides the necessary parameters needed for the external application to connect back with Architect/Requirements. To connect back to the client, the first three parameters can be used. To connect back to the server, the last can be used.

- Only public methods can be called.
- Valid return types for the Java methods are `void`, `String`, and `String[]`. Any values returned from the method are printed to the log file.

- The method parameters must either be empty (no parameters) or must be a single parameter of type `String[]`.
- The location of the `.jar` file must be identified to Architect/Requirements via the **Package.Location** parameter set by the administrator.

EXAMPLES



For source code related to these examples, see [Using createAction RunJava](#) in chapter ***Unsatisfied xref reference**, **Unsatisfied xref title***.

- When the activator runs, it gets a list of all the public methods in the class:

```
createAction RunJava [list TestRunJavaClass ShowMethods]
```

- When the activator runs, it runs the method named **doNothing**, which prints a message to the log and displays a dialog for the user to click:

```
createAction RunJava [list TestRunJavaClass doNothing]
```

- When the activator runs, it verifies that the live Office interface is installed before running the **doNothing** method:

```
createAction RunJava [list TestRunJavaClass doNothing true]
```

- When the activator runs, it displays a **Hello World!** dialog:

```
createAction RunJava [list TestRunJavaClass printHelloWorld false arg1]
```

- When the activator runs with the class name misspelled, it returns a message indicating that the class could not be found:

```
createAction RunJava [list TestRunJavaClass doNothing]
```

- When the activator runs with the method name misspelled, it logs and returns a message indicating that the method could not be found:

```
createAction RunJava [list TestRunJavaClass doNothin]
```

- When each of these activators runs, there are no errors:

```
createAction RunJava [list TestRunJavaClass printHelloWorld false arg1]
createAction RunJava [list TestRunJavaClass doNothing false arg1 arg2 arg3]
createAction RunJava [list TestRunJavaClass doNothingString]
createAction RunJava [list TestRunJavaClass doReturnStringArray]
```

- When each of these activators runs, it results in errors and a return message indicating that the method could not be found:

```
createAction RunJava [list TestRunJavaClass returnInt]
createAction RunJava [list TestRunJavaClass doSomething false 21]
createAction RunJava [list TestRunJavaClass doSomething false arg1 arg2 arg3]
createAction RunJava [list TestRunJavaClass printIt whatever]
```

- When Java runs on the client that calls a server macro:

```
createAction RunJava [list TestRunJavaClass runTestMacro false
[getValue $selected Name]]
```

You must create a macro named **HelloWorldMacro** for the command to work. Add the following code to the macro:

```
displayMessage "Hello World, From Macro, selected: $selected"
return "Hello World (Result)"
```

The example demonstrates passing information to and from the client. The name of the selected object is passed to a Java method running on the client and back to a macro running on the server where the name is displayed. The return value from the macro is passed to the client. The client then displays the return value.

createBaseline

DESCRIPTION

Creates a baseline containing the specified objects.

SYNTAX

```
createBaseline objects name
```

ARGUMENTS

- **objects:** OBJECT_LIST – List of versionable objects to baseline.
name: STRING – The new baselines name.

RETURNS

Void

NOTES

Versionable object types are Requirements and Building Blocks.

EXAMPLES

```
createBaseline [getList $folder MEMBER_LIST] "Version 1"
```

createExternalLink

DESCRIPTION

Creates a Teamcenter Interface (WOLF) link from Systems Architect/Requirements Management to an object in an external application such as Teamcenter Engineering or Teamcenter Enterprise. This API creates a Systems Architect/Requirements Management database entry to record the link, but its operation is entirely internal to Systems Architect/Requirements Management.



createExternalLink does not create such a link in the external application. It does not make any call to notify the external application of the existence of the link. To create such a link in the external application, you must invoke an API of that application, either before or after calling **createExternalLink**.



createExternalLink can be used only for creating original WOLF links; it cannot be used to create new proxy links.

SYNTAX

```
createExternalLink GUID OID name icon biDirectional tcrObject description
```

ARGUMENTS

- **GUID**: **STRING** – Teamcenter Interface external application identifier.
- **OID**: **STRING** – Teamcenter Interface external object identifier.
- **name**: **STRING** – Name of the external object.
- **icon**: **OBJECT** – This argument is reserved for future use.
- **biDirectional**: **BOOLEAN** – This argument is reserved for future use.
- **tcrObject**: **OBJECT** – Systems Architect/Requirements Management end of the WOLF link.
- **description**: **STRING** – Description of the link.

RETURNS

OBJECT–The new WOLF handle object.

createLinks

DESCRIPTION

Creates trace links, generic links, or connections from one object to a list of objects.

SYNTAX

```
createLinks from to [linkType] subtype
```

ARGUMENTS

- **from**: **STRING** – start of links or connections.
- **to**: **OBJECT_LIST** – end of links or connections.
- **linkType**: **STRING** – type of trace link, generic link, or connection to create, default **Complying**, see examples and **DataBean.LINK**.
- **subtype**: **STRING** – subtype of trace link, generic link, or connection to create, if null the base type is used.

RETURNS

OBJECT_LIST–The new trace links, generic links, or connections.

EXAMPLES

- To create trace link where *from* is defining object and *to* is complying:

```
createLinks $from $to  
set newLinks [createLinks $defining $complyingObjs Complying $subType]
```

- To create trace link where *from* is complying object and *to* is defining:

```
createLinks $from $to Defining
```

- To create connection beginning at *from* and ending at *to*:

```
createLinks $from $to Connection
```

- To create connection in the reverse direction:

```
createLinks $from $to {Defining Connection}
```

- To create generic link beginning at *from* and ending at *to*:

```
createLinks $from $to {Outgoing Generic Link}
```

- To create generic link in the reverse direction:

```
createLinks $from $to {Incoming Generic Link}
```

SEE ALSO

`createObject`

createObject

DESCRIPTION

Creates a design object in the database. This command does not create a trace link or project, those are separate commands.

SYNTAX

```
createObject name owner type [position]
```

ARGUMENTS

- **name**: STRING – Name of the new object, if blank, a unique name is generated.
- **owner**: OBJECT – Owner, or sibling, of the new object.
- **type**: STRING – Type, or subtype, of the new object
- **position**: STRING – position of the new object relative to owner. See `DataBean.POSITION`.

RETURNS

OBJECT– The new Object. A Tcl error if objects of the give type can not be owned by subordinate to objects of the given owners type.

NOTES

Valid positions are `LAST_MEMBER`, `FIRST_MEMBER`, `LAST_SIBLING` and `NEXT_SIBLING`. Systems Architect/Requirements Management schema objects may also be created. These objects are normally only created by a Systems Architect/Requirements Management project administrator.

EXAMPLES

```
# Create a requirement as a sibling of a given requirement
createObject "" $Req Requirement NEXT_SIBLING
# Create a Note named "mynote" with the "Rationale" subtype
set newNote [createObject "mynote" $req Rationale]
```

createProject

DESCRIPTION

Creates and initialize a new Systems Architect/Requirements Management project.

SYNTAX

```
createProject [name]
```

ARGUMENTS

- `name`: **STRING** – the new projects name, if not provided an unique default name is generated.

RETURNS

OBJECT–The new project object.

NOTES

Project administration privilege level is required to use this command.

EXAMPLES

```
set proj [createProject {My Project}]
```

createShortcuts

DESCRIPTION

Creates a new Shortcut to the existing object.

SYNTAX

```
createShortcuts orig owner position
```

ARGUMENTS

- `orig`: OBJECT_LIST – The original objects for which the shortcuts are to be created.
- `owner`: OBJECT – The owner (or sibling) of the shortcut.
- `position`: STRING – Position of new shortcut relative to owner, see `DataBean.POSTITION`. This argument is optional. It defaults to `LAST_MEMBER`.

RETURNS

- `scObjects`: The new shortcut objects.

EXAMPLES

```
createShortcuts $requirement $folder LAST_MEMBER
```

createUser

DESCRIPTION

Creates a new Systems Architect/Requirements Management user object.

SYNTAX

```
createUser owner name password maxPrivilege
```

ARGUMENTS

- **owner**: OBJECT – Project that the user belongs to.
- **name**: STRING – The new projects name, if not provided a unique default name is generated.
- **Password**: STRING – The user's initial password.
- **maxPrivilege**: STRING – The maximum privilege the user can have to any project. See `DataBean.USER_PRIVILEGE`.

RETURNS

OBJECT–The new user object.

NOTES

If the owner is null the Administration Project is used. All user objects are created in the Administration project. If the owner is another project, the user is added to that project. If a user with the given name already exists then a new user object is not created, but the user is added to the given project.

EXAMPLES

```
set user [createUser $project "fred" "" "Read and Write"]
```

createVariant

DESCRIPTION

Creates a variant of a versionable object.

SYNTAX

```
createVariant object
```

ARGUMENTS

- `object`: **OBJECT** – The object.

RETURNS

OBJECT–The new variant.

EXAMPLES

```
set variant [createVariant $Req]
```

createVersion

DESCRIPTION

Creates a version of a versionable object.

SYNTAX

```
createVersion object
```

ARGUMENTS

- `object`: **OBJECT** – The object to version.

RETURNS

OBJECT—The new version.

EXAMPLES

```
set version [createVersion $Req]
```

deleteLinks

DESCRIPTION

Remove links between one object and a list of objects.

SYNTAX

```
deleteLinks from to [linkType] subtype
```

ARGUMENTS

- `from`: OBJECT – Start of links.
- `to`: OBJECT_LIST – End of links.
- `linkType`: STRING – Type of link to delete, default Complying, see `DataBean.LINK`.
- `subtype`: OBJECT - Type definition ID for the subtype of link to delete. If this is null, any subtype may be deleted. This argument is optional.

RETURNS

OBJECT_LIST–The deleted links.

NOTES

`linkType` may be Defining, Complying, Connection, Uses, Group, Incoming Generic Link, or Outgoing Generic Link.

A link can also be deleted using a `deleteObjects` call on the link itself. In some cases, there can be multiple links of the same type between two objects. This causes `deleteLinks` to fail due to ambiguity. `deleteObjects` should be used in these cases.

EXAMPLES

```
deleteLinks $from $to  
set newLinks [deleteLinks $defining $complyingObjs Complying $subType]
```

SEE ALSO

`restoreFromTrashcan`

deleteObjects

DESCRIPTION

Deletes Systems Architect/Requirements Management objects and moves them to trashcan. Objects that are already deleted are destroyed.

SYNTAX

```
deleteObjects objects
```

ARGUMENTS

- `objects`: OBJECT_LIST – The objects to delete or destroy.

RETURNS

OBJECTS_LIST–Successfully deleted objects or VOID if destroying objects.

NOTES

Be very careful using **deleteObjects**. For most objects, deletions occur without any further confirmation. The extent of the **deleteObjects** action is the same as if it were done from the user interface. The action for **deleteObjects** is subject to the Access Control settings in the database. When you delete an object (either directly, or because it was a descendent of a deleted object or owned by a deleted object), subsequent attempts to reference it may generate a Tcl error.

When an object is deleted using the Systems Architect/Requirements Management client, a confirmation message window is displayed. For some objects, a second confirmation message window is displayed. The second message warns about special conditions, such as not being able to undo the delete. Tcl developers should confirm in these cases, using **setEnvironment SetResponse**. Projects and other schema objects require **OPT_PARAM_DELETE_PROJECT** set to **Yes** in order to work.

For example:

```
# delete the project
setEnvironment SetResponse Yes OPT_PARAM_DELETE_PROJECT
deleteObjects $project
```

See the **setEnvironment** command for additional details.

Using **deleteObjects** on an object that is already deleted causes it to be removed from the recycle bin. This means, calling **deleteObjects** twice on the same object causes it to be destroyed rather than left in the recycle bin.

EXAMPLES

```
deleteObjects $objList
```

displayMessage

DESCRIPTION

Displays a message in the Systems Architect/Requirements Management client.

SYNTAX

```
displayMessage message [type] [object]
```

ARGUMENTS

- `message`: `STRING` – The message to display.
- `type`: `INTEGER` – See `MessageBean`.
 - 4 – `INFO` (default) display message in popup information box.
 - 5 – `WARNING` Display message in popup warning box.
 - 6 – `ERROR` Display message in popup error box.
- `object`: `OBJECT` – Database object to attach to the message. The object name is added to the message.

RETURNS

`VOID`.

NOTES

Messages are displayed after the current transaction has completed. Popup message windows remain until you click **OK**.

EXAMPLES

```
displayMessage "Hello World"  
displayMessage "Message" 4 $req
```

emptyTrashcan

DESCRIPTION

Destroys the objects in the current user's trashcan.

SYNTAX

```
emptyTrashcan
```

ARGUMENTS

- None

RETURNS

VOID

EXAMPLES

```
emptyTrashcan
```

export2Excel

DESCRIPTION

Exports a Systems Architect/Requirements Management object to a Microsoft Excel file.

SYNTAX

```
export2Excel objects properties isAlive templateID level  
relationship [sessionID]
```

ARGUMENTS

- **objects**: OBJECT_LIST – List Systems Architect/Requirements Management objects to export.
- **properties**: STRING_LIST – The properties to export for each object. This value is unused if a `templateID` argument is provided.
- **isAlive**: BOOLEAN – If true the Excel file can be used in an Excel live session.
- **templateID**: OBJECT – The Excel template or saved view used to format the output. This value must be { } or "" if a properties list is provided to format the output.
- **level**: INTEGER_LIST – Indentation level of corresponding object. This value is used only when the `templateID` argument identifies an Excel template object. See Notes for additional details.
- **relationship**: STRING_LIST – Relationship between corresponding object and its superior. This value is used only when the `templateID` argument identifies an Excel template object. See Notes for additional details.
- **sessionID**: STRING – Optional session ID.

RETURNS

STRING – Excel file name.

NOTES

Either a `properties` list or a `templateID` must be provided. If both are provided, then the `templateID` argument is used and `properties` is ignored.

If an Excel template is identified in the `templateID` argument and the `level` and/or `relationship` lists are provided, they must each be the same length as the `objects` list. Entries in `level` and `relationship` are matched to the corresponding Systems Architect/Requirements Management object in the `objects` list. If the Excel template does not have `level` or `relationship` rules, then the `level` and/or `relationship` arguments can be passed as { } or "".

A unique file name is generated automatically for the excel file. The pathname of the file is returned.

EXAMPLES

Locate a search object and extract the query. The search script is as follows:

```
SELECT Requirement  
  FOREACH  
    ADD Complying Objects  
  
set search {\\Test\Reports and Formatting\ExcelExampleSearch}  
set script [getValue $search Script]
```

Run the search.

```
set result [search $script $selected]
```

Iterate over the first-level search result entries. Entries are lists with 3 elements, the Architect/Requirements object, relationship to the object it is indented under, and a list of sub-entries.

```
foreach entry $result {
    # extract the information from the entry
    set req [lindex $entry 0]
    set relation [lindex $entry 1]
    set subEntries [lindex $entry 2]

    # build lists for the objects, level and relationship arguments for
    export2Excel
    lappend objects $req
    lappend relations $relation
    lappend levels 1

    # iterate over the second level entries (complying objects) ,
    # extract the information and add it to the argument lists
    foreach subEntry $subEntries {
        set obj [lindex $subEntry 0]
        set relation [lindex $subEntry 1]
        lappend objects $obj
        lappend relations $relation
        lappend levels 2
    }
}
```

Set the Excel template to use for the export. Its rule table appears as follows:

Level	Relationship
{%L-1}	
{%L-2}	{%R-Complying Objects}

```
set template {Excel Example}
```

Initialize the remaining export2Excel arguments.

```
set live true
set properties {}
```

Export the excel file.

```
set excelFile [export2Excel $objects $properties $live $template $levels
$relations]
displayMessage $excelFile
```

exportDocument

DESCRIPTION

Writes objects from the database to a Word, Excel, AP233 STP (Part 21 Standard), AP233 XML (Part 28 Standard), or Systems Architect/Requirements Management XML file.

SYNTAX

```
exportDocument objectList type outputTemplate deep includeOLE live
```

ARGUMENTS

- `objectList`: OBJECT_LIST – The objects to export.
- `type`: STRING – The type of file to export. MS_WORD, MS_EXCEL, AP233, AP233_28, XML, SCHEMA, or PROJECT. The default is MS_WORD.
- `outputTemplate`: OBJECT or STRING – The Excel or document template used to format the output. The template can be identified by its object LOID or by its name. The LOID is preferred because it avoids the overhead of searching for a name match.
- `deep`: BOOLEAN – For Word exports, `deep` specifies whether to include the descendents of the selected object(s) in the export. The default is true.
- `includeOLE`: BOOLEAN – For Word exports, `includeOle` specifies whether embedded OLE objects are included in the export. The default is true.
- `live`: BOOLEAN – Applies to Excel exports only. If true, creates an Excel live file. If false, creates a static Excel file. The default is false.

RETURNS

STRING—Path name of the exported file.

NOTES

If a folder is specified in `objectList`, the document template property of the folder is used as the style sheet of the document if the `outputTemplate` argument is omitted.

The exported file types are:

- MS_WORD – Word file in MHTML format.
- MS_EXCEL – Excel file in MHTML format.
- AP233 – AP233 STP file in STEP format, conforming to the Part 21 standard.
- AP233_28 – AP233 XML file, conforming to the Part 28 standard.
- XML – Systems Architect/Requirements Management XML file of the specified objects.
- SCHEMA – XML file of schema objects for the specified project.
- PROJECT – XML file for all of the specified project.
- TC_XML – Export data for migration to Teamcenter.



Exports done through this API are always carried out within the web server process, even if the Systems Architect/Requirements Management's **ExternalImportExport** setting has been configured to use an external process. This could result in performance or memory consumption issues within the web server process.



The file created by `exportDocument` is temporary. It may not have the correct extension and it is deleted automatically after some time. To retain the file, it must be copied or renamed after the export. This can be done using the Tcl `file copy` or `file rename` command.

EXAMPLES

```
set wordFile [exportDocument $folder]
file rename $wordFile {myDocument.mht}
set excelFile [exportDocument $folder MS_EXCEL $excelTemplate]

create an Excel live file
set excelFile [exportDocument $selected MS_EXCEL "Default Excel Template" false
false true]
```

exportXML

DESCRIPTION

Exports Systems Architect/Requirements Management schema objects or projects to an XML file.

SYNTAX

```
exportXML objects [filename]
```

ARGUMENTS

- **objects:** OBJECT_LIST – List of objects to export.
- **filename:** STRING – Filename for the new XML file. The default is to generate a unique filename.

RETURNS

STRING – the XML file name.

NOTES

If objects contains a single project object then that entire project is exported, otherwise the individual schema objects in the list are exported.



This method is deprecated. Instead, the [exportDocument](#) API method is recommended.

EXAMPLES

```
set xmlFile [exportXML $myProject]
```

getEnvironment

DESCRIPTION

Retrieves one of Systems Architect/Requirements Management configuration parameters.

SYNTAX

```
getEnvironment paramName
```

ARGUMENTS

- `paramName`: **STRING** – Name of the parameter to retrieve.

RETURNS

STRING – the parameter value.

NOTES

The valid parameter names can be seen in the parameter name column of Systems Architect/Requirements Management' Web Application Configuration web page.

EXAMPLES

```
set path [getEnvironment ImportExportDir]
```

getList

DESCRIPTION

Retrieves a list of Systems Architect/Requirements Management objects that are related to the given object by the specified relationship.

SYNTAX

```
getList object listType [depth]
```

ARGUMENTS

- `object`: OBJECT – Object containing or owning the list.
- `listType`: LIST_ENUM – The type of object list, see `DataBean.LIST`.
- `depth`: INTEGER – Recursive depth to follow relationship. Default is 1.

RETURNS

OBJECT_LIST.

NOTES

The depth argument is only supported when `listType` is `MEMBER_LIST`, and the given object is a requirement.



The list of Java API functions is provided in the API Javadoc. The API Javadoc describes each function along with the response expected from the server. Additionally, the list of property names, lists, keywords and other constants are defined in the **DataBean** class in API Javadoc.

You can access the Javadoc from the Systems Architect/Requirements Management home page.

1. Click **API Javadoc**.
2. Click the **com.edsplm.tc.req.databeans** package link.
3. From the Class Summary list, click the **DataBean** link.

The constants are defined in a HTML table.

EXAMPLES

```
set members [getList $folder MEMBER_LIST]

getList $obj {PROPERTY_LIST}
```

Returns the list of actual property objects for the given object. This list includes all user-defined properties. It includes some standard properties, but does not include system-defined properties that are stored as Java data members on the object itself, like Create or Change User and Time.

For each Property instance, these calls will return its name and value:

```
getValue $prop {Name}

getValue $prop {Current Value}
```

getObject

DESCRIPTION

Returns a list of property values for the given object.

SYNTAX

```
getObject object propertyList
```

ARGUMENTS

- `object`: **OBJECT** – The Systems Architect/Requirements Management object.
- `propertyList`: **STRING_LIST** – List of desired properties, see `DataBean.PROPERTY`.

RETURNS

OBJECT_LIST – Property values for regular calls and for `PickList` property objects that are stored in the dynamic choice property.

NOTES

In addition to the system properties defined in `DataBean.PROPERTY`, any applicable user-defined properties may be used.

The LOID of a pick list object can be obtained using the `getObject` TcL function. The only needed change is that the property name in these function has a prefix of **LOID**.

So when you need the values for a property, you simply use the `getObject`. But if your need the LOID(s) of the objects that are stored as values in the `PickList` choice property, use `getObject` and prefix **LOID**: in front property name. For additional information, see [Using A Pick List Activator](#) in chapter *Unsatisfied xref number*, *Unsatisfied xref title*.

For information on user-defined, dynamic choice property (pick list property, see the section *Setting a Dynamic Choice List for a Choice Property Definition* in the *Systems Architect/Requirements Management Project Administrator's Manual*.

EXAMPLES

```
set values [getObject $Requirement {ROIN Name Text}]
```

If **Assign to** is a dynamic choice property that exists on the object:

- To obtain the value in the **Assign to** property, make a call as follows:

```
getObject $object "Assign To"
```
- To obtain the LOID of the object in the **Assign To** property, make a call as follows:

```
getObject $object {"LOID:Assign To"}
```

getProjects

DESCRIPTION

Returns a list of Systems Architect/Requirements Management project objects.

SYNTAX

```
getProjects PROJECT_LIST
getProjects ADMIN_VIEW_LIST
getProjects USER_VIEW_LIST
```

ARGUMENTS

- **PROJECT_LIST: OBJECT_LIST** – Get all projects except the admin project.
- **ADMIN_VIEW_LIST: OBJECT_LIST** – Get all projects including admin.
- **USER_VIEW_LIST: OBJECT_LIST** – Get all projects except the admin and include the trash can. This argument is the default if no argument is given.

RETURNS

OBJECT_LIST – Projects to which this user has access.

EXAMPLES

```
set projects [getProjects PROJECT_LIST]
set projects [getProjects ADMIN_VIEW_LIST]
set projects [getProjects USER_VIEW_LIST]
```

getPropertiesWithFormula

DESCRIPTION

Returns a property definition that applies to the given object.

SYNTAX

```
getPropertiesWithFormula objects
```

ARGUMENTS

- `object`: OBJECT – object for which numeric properties with formula is obtained.

RETURNS

OBJECT_LIST–All the Numeric Properties that have formula for the passed objects.

EXAMPLES

```
getPropertiesWithFormula $objects
```

getPropertyDefinition

DESCRIPTION

Returns a property definition that applies to the given object.

SYNTAX

```
getPropertyDefinition object property
```

ARGUMENTS

- `object`: OBJECT – object containing or owning the list.
- `property`: STRING – The property name.

RETURNS

OBJECT– The property definition.

EXAMPLES

```
set propertyDef [getPropertyDefinition $folder Name]
```

getPropertyDefinitions

DESCRIPTION

Returns a list of property definition names that apply to the give object types.

SYNTAX

```
getPropertyDefinitions object types
```

ARGUMENTS

- `object`: OBJECT – type or subtype definition.
- `types`: STRING_LIST – type of property definitions to retrieve. The default value is **All Property**.

RETURNS

STRING_LIST–Property definition names.

NOTES

Hidden properties are those that do not appear in the content pane of the Systems Architect/Requirements Management client. These properties contain information that is not readable.

getRemoteObjectTraceReport

DESCRIPTION

Retrieves the trace report for the remote object. This is used in conjunction with **Proxy linking** between Systems Architect/Requirements Management and Teamcenter Engineering or Teamcenter Enterprise. It is not useful with the original WOLF linking. This action posts a request to the remote system to retrieve the trace report. Hence, there can be a delay at that point.



For troubleshooting, first try using the **Trace Report UI** command. If the **Trace Report** fails then it is likely a configuration problem and not mis-use of the API. Check the Architect/Requirements and Teamcenter Engineering/Teamcenter Enterprise logs for error messages.

SYNTAX

```
getRemoteObjectTraceReport tcrObject
```

tcrObject is the proxy object created when a Teamcenter Engineering/Teamcenter Enterprise object is dragged (or copy/pasted) into Architect/Requirements.

ARGUMENTS

- `tcrObject`: **STRING** – Proxy object.

RETURNS

- `String traceReport`: Trace report in HTML format.

getValue

DESCRIPTION

Retrieves a value from an object.

SYNTAX

```
getValue object property
```

ARGUMENTS

- `object`: OBJECT – Object containing desired value.
- `property`: STRING – System property name (see `DataBean.PROPERTY`) or user-defined property name.

RETURNS

The property's value, or an empty string if the property is not found on the object.

NOTES

The LOID of a pick list object can be obtained using the `getValue` Tcl function. The only needed change is that the property name in these function has a prefix of LOID.

When you need the values for a property, use the `getValue`. But if you need the LOID(s) of the objects that are stored as values in the Pick list choice property, use `getValue` and prefix LOID: in front of the property name.

For more information, see [Using A Pick List Activator](#) in chapter ***Unsatisfied xref number***, ***Unsatisfied xref title***. For information about setting a dynamic choice list for a choice property, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



The MHTML keyword is used to retrieve the text content of a note or requirement including graphics. For performance reasons the OLE content is not included when the MHTML keyword is used. If the text content is opened in Microsoft Word, or used to set the content of another requirement, the OLE objects will be represented with a graphic, but the OLE application cannot be launched. To include OLE content use the MHTML_FULL keyword instead. The MHTML_FULL keyword is only supported in `getValue`, always use MHTML in `setValue`.

For example: # copy the content of a note to a new note including OLE objects

```
setValue $newNote MHTML [getValue $oldNote MHTML_FULL]
```



The list of Java API functions is provided in the API Javadoc. The API Javadoc describes each function along with the response expected from the server. Additionally, the list of Property Names, Lists, Keywords and other Constants are defined in the `DataBean` class in API Javadoc.

You can access the Javadoc from Systems Architect/Requirements Management home page.

1. Click **API Javadoc**.
2. Click the **com.edsplm.tc.req.databeans** package link.
3. From the Class Summary list, click the **DataBean** link.

The constants are defined in a HTML table.

The **Member Count** property returns an approximate value. Some members may not be visible because they are deleted or not effective. For performance reasons, these are not filtered out of the member count. To get a precise member count, use **Actual Member Count**.

EXAMPLES

```
set name [getValue $obj Name]
set txt [getValue $object Text]
```

If **Assign to** is a dynamic choice property that exists on the object.

- To obtain the value in the **Assign to** property, make a call as follows:

```
getValue $object "Assign To"
```

- To obtain the LOID of the object which is stored as the value in the **Assign to** property, make a call as follows:

```
getValue $object "LOID:Assign To"
```

Get the number of items in a folder:

- Get the approximate number of items in a folder

```
set memberCount [getValue $folder "Member Count"]
```

- Get the precise number of items in a folder

```
set memberCount [getValue $folder "Actual Member Count"]
```

importDocument

DESCRIPTION

Imports a Word file or other document type into Systems Architect/Requirements Management.

SYNTAX

```
importDocument owner filename docLocation [subtype] [filetype]
```

ARGUMENTS

- `owner`: OBJECT – The document owner.
- `filename`: STRING – Full path name and the MHTML or XML file name.
- `docLocation`: STRING – Path name of the original document.
- `subtype`: STRING – For MS_WORD and MS_EXCEL import file types, `subtype` identifies the type or subtype name to use for all imported objects. Default is the base requirement type.
For AP233 import file type, the `subtype` argument should specify the property name to use as the unique identifier for objects to be updated. If no updates are desired, specify "".
For all other import types, the `subtype` argument should be present; however, it is unused (specify "").
- `filetype`: STRING – The type of file to import. MS_WORD, STYLESHEET, MS_EXCEL, EXCEL_TEMPLATE, AP233, XML, XML_UPDATE, SCHEMA or PROJECT. The default is MS_WORD.

NOTES

The imported file types are:

- MS_WORD – Word file in MHTML format.
- STYLESHEET – Word file in MHTML format. Only the stylesheet section is imported. Owner must be a Stylesheet.
- MS_EXCEL – Excel file in MHTML or XML format.
- EXCEL_TEMPLATE – Excel file in MHTML format. Owner must be an Excel Template.
- AP233 – AP233 file in STEP format.
- XML – XML file with information about new objects to be created.
- XML_UPDATE – XML file with information to update existing objects (when the ID matches an existing object's LOID) or create new objects.
- SCHEMA – XML file of schema objects. Owner must be a project.
- PROJECT – XML file for an entire project. Specify a real project for the owner. A new project is created.

For more information about the Architect/Requirements XML format, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



Imports done through this API are always carried out within the web server process, even if the Systems Architect/Requirements Management's **ExternalImportExport** setting has been configured to use an external process. This could result in performance or memory consumption issues within the web server process.

RETURNS

STRING—Folder LOID number.

moveObjects

DESCRIPTION

Moves the source objects to the specified destination.

SYNTAX

```
moveObjects sources destination
```

ARGUMENTS

- `sources`: OBJECT_LIST – Objects to move.
- `destination`: OBJECT – New owner of moved objects.

RETURNS

OBJECT_LIST–The moved objects.

EXAMPLES

```
moveObjects $requirements $folder
```

restoreFromTrashcan

DESCRIPTION

Restores deleted objects. Restored objects are no longer marked as deleted and are moved to the specified new owner or back to their original owner.

SYNTAX

```
restoreFromTrashcan objects [newOwner]
```

ARGUMENTS

- `objects`: OBJECT_LIST – Object containing desired value.
- `newOwner`: OBJECT – New owner for the restored objects, default is to restore to the original owner.

RETURNS

OBJECT_LIST–The restored objects.

EXAMPLES

```
set object [restoreFromTrashcan $deleteObject]
```

runActivator

DESCRIPTION

Runs the specified activator specified by Activator ID.

SYNTAX

```
runActivator name current selected
```

ARGUMENTS

- `name`: **STRING** – The name of the activator.
- `current`: **OBJECT** – Object to set as the current object in the activator
- `selected`: **LIST** – List of additional information to pass into the activator

RETURNS

STRING - the activator result or an error message if the activator failed.

runReport

DESCRIPTION

Searches the database for objects that match the specified criteria and output them in a report file.

SYNTAX

```
runReport report template startingObject live
```

ARGUMENTS

- `report`: OBJECT – Saved Search object that contains the search query.
- `template`: OBJECT - Excel Template, Document Template or saved View object used to format the report output.
- `startingObject`: OBJECT - Starting object for the search. If omitted or empty then the starting object for the saved search is used.
- `live`: BOOLEAN - Pass **true** if exporting to Excel via a Template or View to get a "Live" spreadsheet, or **false** for static spreadsheet.

The `live` argument is not used when exporting to MS Word. Export to MS Word is implied when the `template` argument identifies a Document Template object.

RETURNS

Full pathname of the server file where the Word or Excel report output is written. The **createAction FileDownload** API can be used to move this file to the client workstation.

EXAMPLES

```
set report "\\[getValue $currentProject Name]\\Reports and Formatting\\myReport"
set report [getValue $report LOID]

set template "\\[getValue $currentProject Name]\\Reports and
Formatting\\myTemplate"
set template [getValue $template LOID]

# Run the report, download the server file to the client and open in Excel
set filePath [runReport $report $template $startingObject "false"]
createAction FileDownload [list $filePath MS_EXCEL]
```

search

DESCRIPTION

Return a list of objects matching the search parameters.

SYNTAX

```
search searchSpec startingObject
```

ARGUMENTS

- `searchSpec`: **STRING** – XML specification for the search.
- `startingObject`: **OBJECT** – Starting point for search

RETURNS

OBJECT_LIST– Nested list of objects matching the search criteria.

NOTES

Constructing the `searchSpec` XML can be difficult. Instead of constructing it from within your Tcl, use the **Search** module user interface to construct and debug a search, save the search and let TcSE give you the XML. In the **Administration** module, select the **Search** object and scroll the **Properties** tab to find its **Script** property. Triple-click in the field and use Ctrl+C to copy the script's XML text to the clipboard. Now, paste the text in a text editor, or directly into your Tcl code. You can see where to substitute different property names or values in **WHERE** clauses, or conditionally exclude or add sections of the script as required.

EXAMPLES

```
set results [search $searchXml $obj]
```

search

DESCRIPTION

Searches the database for objects that match the specified criteria. The search is not case sensitive.

SYNTAX

```
search base nameSpec contentSpec types [caseSensitive]
```

ARGUMENTS

- **base**: OBJECT – Folder or project to search in.
- **nameSpec**: STRING – The object name search criterion.
- **contentSpec**: STRING – The body text search criterion.
- **types**: STRING – Object types to include in the result, see `DataBean.TYPE`.
- **caseSensitive**: BOOLEAN – True to perform case sensitive search, false for noncase sensitive search. The default is false.

RETURNS

OBJECT_LIST— Objects matching search criteria.

NOTES

Types may only include `NoteDB`, `FolderDB`, and `RequirementDB`. The wildcards `?` and `*` may be used in the spec strings. **contentSpec** applies for notes and requirements; it is ignored for folders.

EXAMPLES

```
set reqs [search $project "" "shall" {RequirementDB}]
```

sendEmail

DESCRIPTION

Tcl Command class to send E-Mail to List of EmailIds.

SYNTAX

```
sendEmail recipients subject objectContentFormat objects sendMailAs  
mailBodyMessage docTemplate
```

ARGUMENTS

- `recipients`: **LIST** – List holding recipients IDs.
- `subject`: **STRING** – Variable to hold the subject of the mail.
- `objectContentFormat`: **STRING** – Variable to hold object's content format options: Text,HTML,MHTML or URL.
- `objects`: **LIST** – List of Systems Architect/Requirements Management objects.
- `sendMailAs`: **STRING** – String to hold send mail option: send as body/attachments.
- `mailBodyMessage`: **STRING** – String to hold mailBodyMessage.
- `docTemplate`: **STRING** – String document template used to get object's content as expected document.

RETURNS

- `mailSendStatus`: **Boolean** indicating success or failure of sendMail function.

NOTES

- The *recipients* argument is a Tcl list including one or a combination of the following:
 - Actual email addresses in the form name@host.
 - UserGroup object Name or LOID.
 - User object Name or LOID.
- When a User is identified by Name, LOID or User Group membership, the email is sent using the **Email** property value for that user.
- The *objects* argument must be a list of LOIDs.
- TcSE's MailServerIP configuration setting must point to a valid email server.

setEnvironment

DESCRIPTION

Sets one of Systems Architect/Requirements Management' configuration parameters.

SYNTAX

```
setEnvironment param value
```

ARGUMENTS

- `param`: STRING – Name of the parameter to set.
- `value`: STRING – The new value for `param`.

RETURNS

VOID.

NOTES

Parameters that may be set are:

- `ChangeList`: Value may be set to true or false. Setting `ChangeList` to false suppresses change handling in Systems Architect/Requirements Management. This can be useful to improve the performance of large Tcl transactions. Turning change handling off suppresses After activator execution, but Before activators run as usual. Also, turning change handling off results in the Systems Architect/Requirements Management client not refreshing completely.
- `checkRedundantLinks`: Value may be set to true or false. The default value is true. Setting `checkRedundantLinks` to false suppresses redundant relationship checking when creating trace links and group links. This can be useful to improve the performance of operations that create a large number of links. The setting lasts until it is explicitly changed again or the end of the transaction. Exercise caution when setting `checkRedundantLinks` to false as it creates redundant links.
- `checkCircularLinks`: Value may be set to true or false. The default value is true. Setting `checkCircularLinks` to false suppresses circular relationship checking when creating trace links and group links. This can be useful to improve the performance of operations that create a large number of links. The setting lasts until it is explicitly changed again or the end of the transaction. Exercise caution when setting `checkCircularLinks` to false as it creates circular links.
- `ContextProject`: Value passed must be the LOID of a project object. Systems Architect/Requirements Management always has the notion of a "current project". This value is reflected in the **currentProject** global Tcl variable. But, there are times when a different project context is needed. For example, the `getValue` call for the "User Access" property of a User object returns that person's access to the current project. This example would allow getting the access level for a different project:

```
setEnvironment ContextProject $proj2
set access [getValue $user "User Access"]
```

The Project context change only applies during the activator execution where the `setEnvironment` call is made. When the activator ends, the `ContextProject` is reset to the current project. But, within the same activator, it may be necessary to reset the context project back to the original value as it could influence later API calls within that activator. If your activator continues on to perform other actions it is better to reset the context project to the current project:

```
setEnvironment ContextProject $currentProject
```

- `FastMode`: Boolean option to enable or disable implicit re-number refresh. The default is true, which enables implicit re-number refresh. Requirements and building blocks have a hierarchy property. When a numbered object is created, deleted or moved it may cause some of its siblings and descendents to be re-numbered. Refreshing the client can be expensive when a large number of siblings are present. Suppressing re-number refresh may improve performance significantly, at the cost of displaying outdated number information until the next refresh.

The `FastMode` setting is effective for the entire client session. The mode applies until the client logs out or it is reset with another `setEnvironment` call.

- `FlushUndo`: No value required. `FlushUndo` empties the undo queue so no prior transactions can be undone.
- `initProgress`: Macros and menu commands run from the client displays the client's progress meter thereby preventing timeouts. `initProgress` allows Tcl developer to inform users of a command's progress and enables cancelling the command. Tcl `setEnvironment` actions controls the content displayed in the progress dialog.

```
setEnvironment initProgress <message> <totalSteps>
```

The given message is displayed in the progress dialog's message area. The `totalSteps` given in the `initProgress` informs the progress meter of the "steps" it required to reach 100%. The `currentStep` values passed in subsequent `updateProgress` calls must be from 0 to the `totalSteps` value. If `totalSteps` is omitted, the progress meter oscillates and does not show a progression from 0% to 100%.

- `MessageQueue`: Value may be omitted or set to **clear**.
 - If the value is omitted, the current contents of the message queue (message tags, not the full text) is returned.
 - If the value is **clear**, the message queue is also cleared.



API callers need to use this call with caution. Users may be left unaware of important messages when this call is used.

- `queueResult` – This command causes a copy of the results of the current server call to be stored off so they can be retrieved and processed by the TcSE client at a later time. This is useful in situations where a server call is made from something other than the TcSE client, such as a JSP, but you want the results to be handled in the client. Login activators run during a server call from the login JSP so this command is needed if the login activator generates any messages or actions that need to be processed on the client. Information on the result includes:
 - Change list – Information about objects modified in the transaction that is used to refresh the client.
 - Message list – Message generated by the TcSE server or the Tcl `displayMessage` command.
 - Action list – actions generated by the Tcl `createAction` command.

If this command is included in Tcl code that was run during a call from the client it will be ignored. This prevents the same result information from being processed twice. Once a queued result is saved on the server it will be returned to the client immediately following the next server call from the client. This means that some user action, such as selecting an object, is required in order to see the result.

- **SaveData:** Allows Tcl to save values that can be retrieved during a different transaction later in a user's session. This data will be lost if the session times out, or when the user logs out. The form of the call to save a value is:

```
setEnvironment SaveData uniqueID value
```

The `value` argument is any Tcl string. The `uniqueID` argument is a string identifier, which will be used later to retrieve the value. If a value has already been saved with that ID, the old value is replaced by the new one. But, that old value is returned as the result of the `setEnvironment` call. So, you can retrieve the old value and set a new one all as one call:

```
set oldValue [setEnvironment SaveData uniqueID newValue]
```

The form of the call to just retrieve a previously saved value is:

```
set myValue [setEnvironment SaveData uniqueID]
```

If there is no previously saved value with that ID, the result is an empty string, with no error indication. A retrieved value remains stored and can be retrieved again. To release the memory from a saved value, set it to the empty string:

```
setEnvironment SaveData uniqueID ""
```

This call retrieves the existing value and clears it as one operation:

```
set oldValue [setEnvironment SaveData uniqueID ""]
```



Clearing values that are retrieved and no longer needed is a recommended practice. Values saved with this feature occupy Web server memory associated with the user's HTTP session. Use this feature for storing simple flags, or short values. Avoid saving large amounts of text this way.

- **SetResponse:** Value is **Yes** (the default value) or **No**. Use the `SetResponse` parameter for operations that require a user response before executing. For example, some database modifications cause the system to prompt you with a warning; you must click **Yes** to continue. When this occurs during Tcl execution, the programmer must provide the response using the `SetResponse` parameter.

The `setEnvironment` method supports the following named responses:

<code>OPT_PARAM_DELETE_PROJECT</code>	Used by the <code>deleteObjects</code> method, for objects in the Administration module.
<code>OPT_PARAM_CREATE_BASELINE</code>	Used by the <code>createBaseline</code> method.
<code>OPT_PARAM_NO_PENDING_CHANGE_CREATE_BASELINE</code>	Used by the <code>createBaseline</code> method.

For example, a selected activator can be deleted by using a macro that contains:

```
setEnvironment SetResponse Yes OPT_PARAM_DELETE_PROJECT
deleteObjects $selected
```

The unnamed response `""` is used by all other methods.

- **SessionID:** No value required. `SessionID` contains the value the server maintains for the current Systems Architect/Requirements Management `SessionID`. This value can be used for

automating Systems Architect/Requirements Management logins from JSP code launched from an activator or macro.

- `assignRoins`: In order to avoid locking conflicts with the ROIN counter object a ROIN is not assigned to a requirement when it is created. ROINs are instead assigned when the transaction is committed. If you need to examine the ROIN of a newly created requirement you will need to force the assignment of a ROIN. The `assignRoins` command will assign ROINs to all the requirements created in the current transaction that do not yet have a ROIN.

```
setEnvironment assignRoins
```

- `updateProgress`: Macros and menu commands run from the client displays the client's progress meter thereby preventing timeouts. `updateProgress` allows Tcl developer to inform users of a command's progress and enables cancelling the command. Tcl `setEnvironment` actions controls the content displayed in the progress dialog.

```
setEnvironment updateProgress <message> <currentStep>
```

The given message is displayed in the progress dialog's message area. The `totalSteps` given in the `initProgress` informs the progress meter of the "steps" it required to reach 100%. The `currentStep` values passed in subsequent `updateProgress` calls must be from 0 to the `totalSteps` value. If `totalSteps` is omitted, the progress meter oscillates and does not show a progression from 0% to 100%.

- `deltaMode`: The web application parameter `DB.DeltaCapture` sets the capture settings for the LOIDs of modified database objects. The LOIDs of modified database objects are captured if `DB.DeltaCapture` is set to **true**. `DB.DeltaCapture` can be overridden within the current transaction, using the boolean `deltaMode` parameter.

Following is the command to disable LOID capture:

```
setEnvironment deltaMode false
```

- `includeImages`: For use only in activators used in object templates. An activator in an object template can return rich text content from a different Requirement or Note by referencing its `HTML` property. In that case images in the referenced object are not included. The `includeImages` keyword instructs the Word export process to include images for the specified list of objects. For example:

```
setEnvironment includeImages [list $otherRequirement]
return [getValue $otherRequirment HTML]
```

EXAMPLES

Example of common usage:

```
setEnvironment ChangeList false
```

Example of common usage of `SessionID` parameter:

```
set sessionID [setEnvironment SessionID]
createAction launchWebBrowser [list urlString sessionVarName $sessionID]
```

setObject

DESCRIPTION

Sets a list of property values on the given object.

SYNTAX

```
setObject object propertyList valueList
```

ARGUMENTS

- `object`: **OBJECT** – The Systems Architect/Requirements Management object.
- `propertyList`: **STRING_LIST** – List of desired properties, see `DataBean.PROPERTY`.
- `valueList`: **STRING_LIST** – List of the values to set.

RETURNS

VOID.

NOTES

In addition to the system properties defined in **`Databean.PROPERTY`**, any applicable user defined properties may be used.

If you are setting the value of the property by passing some String, simply use the regular `setObject`. But if you want to pass the LOID of the objects for the Pick list property as its values, you need to make the same `setObject` call with `LOID:` as a prefix for the property. If you are setting the value on a pick list, hat value should be a name of the object or its LOID.

For additional information, see [Using A Pick List Activator](#) in chapter ***Unsatisfied xref number***, ***Unsatisfied xref title***.

For information on user-defined, dynamic choice property (pick list property, see the section *Setting a Dynamic Choice List for a Choice Property Definition* in the *Systems Architect/Requirements Management Project Administrator's Manual*.

Setting a dynamic choice property using the `LOID:` form is significantly faster than setting it by value.

EXAMPLES

```
setObject $Requirement {Name {Paragraph Number}} {Speed 2}}
```

If **Assign to** is a dynamic choice property that exists on the object, and **c User Group** is an object that exist in the project:

- To set the object of the **Assign To** property, make the following call:

```
setObject $currentObject "Assign To" "c User Group"
```
- To pass the LOID of the object of the **Assign To** property as its values, make the following call:

```
setObject $currentObject "LOID:Assign To" "497.0.7800"
```



497.0.7800 is an example LOID.

setPassword

DESCRIPTION

Changes a user's password.

SYNTAX

```
setPassword name newPassword oldPassword
```

ARGUMENTS

- `name`: **STRING** – The name of the user.
- `newPassword`: **STRING** – The new password for user.
- `oldPassword`: **STRING** – User's old password.

RETURNS

OBJECT–The modified user object.

NOTES

May be used only by the project administrator or to change your own password. Project administrators do not need to specify `oldPassword` when changing another user's password.

EXAMPLES

```
setPassword "fred" "xyzy" "plugh"]
```

setUserPreferences

DESCRIPTION

Allows users to specify their location so that information can be formatted appropriately.

SYNTAX

```
setUserPreferences names values
```

ARGUMENTS

- `name`: `STRING_LIST` – List of preferences names.
- `values`: `STRING_LIST` – List of preference values.

RETURNS

None.

NOTES

Preferences that can be set are:

`timezone`: Specify the time zone of the client so dates and times are displayed in local time (see **DataBean.PROPERTY.TIMEZONE** and **java.util.TimeZone**). Timezone determines the time shift from GMT.

`locale`: Specify the locale of the client (see **DataBean.PROPERTY.LOCALE** and **java.util.Locale**). Locale controls how date, time and numbers are formatted, and in what language months are displayed

Java developers can use `setObject` to set preferences.

EXAMPLES

```
SetUserPreferences timezone "America/Los_Angeles"
```

```
setUserPreferences [list timezone locale] [list "GMT-8:00" "en_US" ]
```

setValue

DESCRIPTION

Sets the value of a property on the given object.

SYNTAX

```
setValue object property value [lightweight]
```

ARGUMENTS

- **object**: OBJECT – The Systems Architect/Requirements Management object.
- **property**: STRING – The property to set, see `DataBean.PROPERTY`.
- **value**: STRING – New value for property.
- **lightweight**: BOOLEAN – True if setting a lightweight property, false otherwise. The default is false.

RETURNS

BOOLEAN – Value indicating whether or not the operation was successful.

NOTES

In addition to the system properties defined in `DataBean.PROPERTY`, any applicable user defined property may be used.

Lightweight properties allow an API caller to save a named value on an object without requiring a property definition. Setting a lightweight property creates, or updates, a text property instance with the given name and value. Setting a lightweight property to an empty string (“”) causes the lightweight property instance to be deleted.

Lightweight properties are intended for cases where only a minority of objects need that particular value saved. If most or all instances of a type require the value, an ordinary property definition should be defined and assigned to that object type. Lightweight properties have several limitations:

- Cannot be displayed as a column in the Systems Architect/Requirements Management client views
- Appear in the Properties tab, but are not editable
- Do not appear in the selection list of properties in the Search module
- Cannot have a lightweight property of the same name as a System or User Defined property on the same object.

When certain properties are set, such as the Properties property of a Type Definition, you are prompted with a warning message and must click **Yes** to continue. When setting these properties using `setValue`, the answer to the question must be supplied. The following line of code must be included before the `setValue` or the property will not be set:

```
setEnvironment SetResponse Yes
```

To add, remove, or reorder the allowed choices for a Choice List property definition, set its **Choice List** property. When calling `setValue`, you must pass a string that includes alternating new and old values, in the order you want them to be in. The values are separated by the internal delimiter (middle dot (.)) as follows: `new.old.new.old`. Added choices or deleted choices are indicated by a null string (empty string) between the delimiters. That is, when adding a choice, there is no old value, so there are two adjacent middle dot characters. Likewise, when deleting a choice, the new position is empty.

If you are setting the value of the property by passing some String, use the regular `setValue`. But if you want to pass the LOID of the objects for the Pick list property as its values, you need to make the same `setValue` call with `LOID:` as a prefix for the property. If you are setting the value on a pick list, that value should be a name of the object or its LOID. For more information, see [Using A Pick List Activator](#) in chapter ***Unsatisfied xref number**, **Unsatisfied xref title***.

Setting a dynamic choice property using the LOID: form is significantly faster than setting it by value. For information about setting a dynamic choice list for a choice property, see the *Systems Architect/Requirements Management Project Administrator's Manual*.



Use the **Static** property for freezing and unfreezing an object.

- Setting **Static** to **true** freezes the object.
- Setting **Static** to **false** unfreezes the object.

For example:

```
# freeze the requirement
setValue $req Static true
```



Due to delayed ROIN assignment it may be necessary to explicitly assign ROINs before setting the text content (MHTML) of a requirement (see `setEnvironment assignRoins`).



The MHTML keyword is used to retrieve the text content of a note or requirement including graphics. For performance reasons the OLE content is not included when the MHTML keyword is used. If the text content is opened in Microsoft Word, or used to set the content of another requirement, the OLE objects will be represented with a graphic, but the OLE application cannot be launched. To include OLE content use the MHTML_FULL keyword instead. The MHTML_FULL keyword is only supported in `getValue`, always use MHTML in `setValue`.

For example:

```
# copy the content of a note to a new note including OLE objects
setValue $newNote MHTML [getValue $oldNote MHTML_FULL]
```



The list of Java API functions is provided in the API Javadoc. The API Javadoc describes each function along with the response expected from the server. Additionally, the list of property names, lists, keywords and other constants are defined in the **DataBean** class in API Javadoc.

You can access the Javadoc from the Systems Architect/Requirements Management home page.

1. Click **API Javadoc**.
2. Click the **com.edsplm.tc.req.databeans** package link.
3. From the Class Summary list, click the **DataBean** link.

The constants are defined in a HTML table.

EXAMPLES

```
setValue $Requirement Name "Maximum speed"
```

To make the changes in the following table:

Table 5-1. Sample Change List

New	Old	Description
Susan	Susie	Change name from Susie to Susan.
Mark	Mark	No change.
Sally	null	Add Sally in the third position.
null	Peter	Delete Peter.
Henry	null	Add Henry in the fourth position.
y	x	Rename from x to y; troublesome rename, server catches.
x	y	Rename from y to x; troublesome rename, server catches.

The Tcl code to set the choices in the example would be:

```
global choiceDelimiter
set choices [list Susan Susie Mark Mark Sally "" "" Peter Henry "" y x x y]
setValue $object "ChoicePropertyName" [join $choices $choiceDelimiter]
```

- To set the object of the **Assign To** property, make the following call:

```
setValue $currentObject "Assign To" "c User Group"
```

- To pass the LOID of the object of the **Assign To** property as its values, make the following call:

```
setValue $currentObject "LOID:Assign To" "383.0.19579"
```



383.0.19579 is an example LOID.

uncoupleShortcuts

DESCRIPTION

Uncouples a shortcut from the master object, creating a copy of the master object. If the shortcut includes the children of the original object, then it is a deep copy. Otherwise only the master object is copied.

SYNTAX

```
uncoupleShortcuts shortcut
```

ARGUMENTS

- `shortcuts: OBJECT_LIST` – The shortcut objects to be uncoupled.

RETURNS

`newObjects`: The new object copies.

EXAMPLE

```
uncoupleShortcuts $shortcut
```

writeLog

DESCRIPTION

Writes a message to the web server's log file.

SYNTAX

```
writeLog message
```

ARGUMENTS

- `message`: **STRING** – Message to place in log file.

RETURNS

VOID.

NOTES

The name and default location of the web server log file varies with each of the web servers, and the server may be configured to use a different location.

Use caution in writing to the log file. Excessive logging can impact performance, and lead to very large log files. Recording error conditions or other significant events is appropriate, but do consider how frequently these events may happen in production. Debug printing is appropriate when developing activators or other Tcl, but must be removed for production operations.

Chapter 6: Using Activators

This chapter describes activators, including macros, and explains how to use them.

Introduction

In Systems Architect/Requirements Management customized behavior can be associated with object types. This behavior is defined in objects called activators. For example, an activator can be associated with requirements so that it is triggered when any requirement is modified.

Activators can access the Systems Architect/Requirements Management database, operating system commands, and other applications. Systems Architect/Requirements Management uses Tcl (Tool Command Language) as the language for programming activators.

Access Privileges for an Activator

Users need the script authoring privilege to create, edit and run activators in each project. To give this privilege to a user for a particular project, change the user's user object in the project's users folder by setting the Additional Privilege property to include Script Authoring. Because Enterprise users have access to all projects in the system, their user objects appear only in the users folder of the Systems Architect/Requirements Management Administration project. However, a user does not need script authoring privilege if an activator is triggered as a result of another user's actions.



You must have the following access privileges to create, edit, or view a Tcl script for an activator:

- To create or edit an activator, the maximum privilege level assigned to you in the project must be **Project Administrator** or **Enterprise Administrator**.
- After an activator is created, it can have a security profile that grants editing rights to other users with **Read and Write** access privilege.
- In all cases, you must have the special privilege of **Script Authoring** in the project to create, edit, or view the text of an activator.

For more information about project access and special privileges, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

Describing Activator Objects

Activators are executable behavior that trigger and execute when certain events occur as a result of user action in the Systems Architect/Requirements Management client. Activators contain executable code written in Tcl. When an activator triggers, the Tcl code it contains is executed. For example, an activator that triggers on requirement modification could:

- Create a note to record the rationale for the requirement change.
- Set an attribute showing that the requirement had changed and needs review.
- Notify members of the Configuration Control Board of the change through E-mail.

Activators are created in the Activators folder of a project in the Administration module. Activators have an event property that indicates which events cause the activator to trigger.

To enhance performance and for technical reasons, there are cases where Activator events are not triggered by design.

- Activators triggered by a Before event do not cause other Before activators to be run. But, Before activators cause After activators to be run.
- Activators triggered by an After event do not cause any other Before or After activators to be run.
- After events are not triggered when the `setEnvironment changeList FALSE` option is in effect. The internal mechanism that collects the change list also drives the After events, so that information is not available when the change list is suppressed. Before activators are still run as usual.
- When you import Word, XML, AP-233, or Excel file in create or update mode, the **Undo** command does not work.
- When you update properties while importing an AP233 file, Change Logs are captured if you enable **Change Logging** for those modifications. For all imports (Word, XML, AP-233, or Excel) in create mode, changes should not be logged even if **Change Logging** is enabled

Creating Activators

This section presents the steps required to set up an activator.



See [Access Privileges for an Activator](#) for information about access privileges required to create an activator.

To set up an activator:

1. In the administration module, open a project, select the Activators folder and run the **New:Activator** command.
2. Open the activator and enter the Tcl script to execute.
3. Edit the Activators Events property and set the events that will trigger the activator.
4. Open the Type Definitions folder, select an object type, edit the activators property and select the activator so that it applies to objects of this type.



- After the Tcl script of an activator is saved, there may be a small delay of a few seconds until the changes are saved on the server. If the user reopens the **.tcl** file immediately, the changes may not be reflected.

This delay is caused by the mechanism used to determine whether any changes are made to the file. The Systems Architect/Requirements Management has no direct access to the editing application as it allows the user to use any editor to edit an activator. Therefore, the Systems Architect/Requirements Management must poll the edited file to check whether there are any changes. Polling too often may affect the performance, so there is an interval of a few seconds between the polls.

- When you open an activator for editing in multi-pane editors, the Systems Architect/Requirements Management places a reservation on the activator. You must use the **Tools→Release Reservation** command if other users want to edit the same activator. For more information, see the chapter *Working With Object Properties* in the *Systems Architect/Requirements Management User's Manual*.

Using Activators in Excel and Object Templates

Excel templates and object templates allow activators to be called as follows:

- {**%Activator**%Name }

The activator is called when exporting an object using the Excel template row or the object template. If an activator by the given name is found in the current project, it is used. If none is found, the Architect/Requirements Administration project is checked. If none is found there, an error is generated.

When the activator is run, the object being exported is identified in the **currentObject** global Tcl variable. Arguments can be passed to the activator using this syntax:

For more information about Excel templates and document templates, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

- {**%Activator**%Name%Arg1%Arg2%Arg-n }

From within the activator code, the arguments are found as a Tcl list in the **selected** global Tcl variable.

Properties of the current object can be passed as argument values by embedding the `{%propName}` syntax, as it is normally used in Excel and object templates:

- `{%Activator%Name%{%Create User}%{%Change User}%}`.

Within the activator, those arguments are retrieved from the Tcl list like this:

```
set createUser [lindex $selected 0]
set changeUser [lindex $selected 1]
```

When used within an object template, an activator is expected to return valid HTML text. This could be a plain text string with no HTML markup or the string could include simple HTML markup, such as `bold` and `<I>italic</I>`. It can also include complex tables, using the `<table>`, `<tr>`, `<td>` notation, or any other valid HTML content that Microsoft Word allows. If the string is to include any HTML reserved characters as visible text, they must be encoded using HTML conventions. For example, ampersand (&) is encoded as `"&"`, and less than (<) as `"<"`. Non-ASCII characters must be encoded, else they do not display correctly.

Architect/Requirements property values are allowed to contain non-ASCII characters, so strings returned by **getValue** must be encoded. Use the following command for encoding:

```
set encodedContent [java::call org.htmlparser.util.Translate encode $content]
```

When used in an Excel template, an activator returns one of the following:

- o A static string to be displayed in that cell of the exported Excel spreadsheet. The returned string can be plain text, or can include HTML markup, as described above, and subject to what Excel allows in cell content. Even if the spreadsheet is exported to live Excel, this text is not associated with any object or property.



The `getValue $obj` HTML return value for a requirement (or paragraph or note) may or may not be valid HTML to be inserted in an Excel cell. Excel does not support all the content and formatting that Word does, or in the same way. For example, paragraph breaks and certain style ranges in the HTML text cause extra rows to appear in Excel. These are traits of how Excel handles HTML content, and is not Architect/Requirements behavior. Also, if the text includes embedded graphics or OLE objects, they are not included in the result. If you want to include an object's text, use the **Text** property, although it does not retain any markup.

- o An object and property reference to create a live cell, as a colon-delimited string in the form `:nn.n.nnnn:propName:`, where `nn.n.nnnn` is the LOID of an object (in valid format), and `propName` is the name of a property of that object. In the Excel export file, the cell displays the current value of that property on that object. If the spreadsheet is exported to live Excel, cells exported in this form are associated with the object and property and can be edited in the same way as other live cells.



An object template activator can return rich text content from a different requirement or note by referencing its HTML property. When this is done the images from the referenced object are not included in the Word export. To include the images use the `setEnvironment includeImages` command. For example:

```
setEnvironment includeImages [list $otherRequirement]
return [getValue $otherRequirement HTML]
```

Defining Events

Systems Architect/Requirements Management recognizes many internal states, each of which could be considered an event. When such an event occurs, any activators defined for that event, on that type of object, are executed. Events generally fall into two categories:

- **Object Modify Events** occur when objects get modified. This includes relationships, moves, and editing properties.
- **Session Activity Events** are events that are triggered by logging into or out of Systems Architect/Requirements Management.



Objects modified while running an activator do not trigger an event.

Defining Object Modify Events

An Object Modify event type is generated when objects are modified, created, or deleted. There are two types of Object Modify events: Before Change events and After Change events.

- A *Before Change* event is generated just before an object is modified, but only after the necessary objects are locked and access control is checked.
- An *After Change* event is generated after the modification is complete, but just before the transaction completes and the response is returned to the user.

Before Change events are triggered only on objects that are selected for an operation, while After Change events are triggered for all modified objects. For example, if a folder is selected for deletion, a Before Delete event is triggered on the folder, but not on requirements in the folder. But After Delete events are triggered for all objects that are deleted. When more than one After Modify activator is triggered by an operation, they may be run in any order.

Before Change events are triggered when a modify operation is requested. After Change events are not triggered until the end of the API call, after all changes have been completed. Even if an object is modified multiple times, there is only one activator run for a particular event. For example, if two property values are modified, an After Modify Activator runs once for the modified object. The change list identifies which two properties were modified.

Before Change events are useful if you want to execute, check, or test a few conditions before the actual modification takes place. For example, to do additional consistency checking, enforce a design

methodology, or impose additional access control, you could define a Before Modify event on an activator.

After Modify events are useful if the user wants to perform additional operations on the objects involved in the event or on some other objects; for example, if you want to set a few attributes after an object is created.

The typical trigger for Object Modify events is a user modifying objects in the Systems Architect/Requirements Management client. But, Tcl API calls can also be the trigger. Ordinary activators running in response to object events do not cause other activators to be triggered. This behavior protects against the infinite recursion that's possible if one activator caused others to run. But, other Tcl code, running outside the context of an event, can cause Object Modify events to be queued and applicable activators to run. This includes Tcl running as macros and menu commands.

The following events are supported in Systems Architect/Requirements Management:

- **Before Create:** This event is set on the Type Definition for the type of object that is about to be created. But, the new instance of that type does not exist at the time of the Before event, so the target object of the activator is the intended owner of the new object. (You need two target objects when creating a link.) This activator could cause the Create action to be cancelled before any change takes place.
- **Create:** This event is generated after an object is created.
- **Before Modify:** This event is generated when a property of an object changes. The event is triggered before the actual database modification occurs.
- **After Modify:** This event is generated when a property of an object changes.
- **Before Delete:** This event is generated when an object is deleted. This operation occurs before the object is actually deleted. This event is triggered only on objects that are selected for deletion. Objects that are deleted as a side effect of deleting the selected object (such as notes, trace links, and children of the selected object) do not trigger this event.

When a User Group object is deleted, a Before Delete event gets executed twice.

- It is triggered first just after you reply **Yes** to the **Are you sure you want to delete the selected object?** message, just before the **This operation cannot be undone. Continue?** message appears. Even though the Before Delete event has triggered at this point, the User Group is not yet deleted, and does not get deleted if you reply **No**.
- If you now reply **Yes**, then the Before Delete event is triggered the second time and the User Group is deleted.

There is no way to tell the difference between these two Before Delete events. Unlike ordinary Groups, there is no After Delete event when a User Group is deleted.

Performing actions in before activators generally does not work, and it is not recommended. For example, if you want to put deleted objects in a special folder instead of the trash you cannot use a before activator. You can move the objects, but they are still deleted when the command continues.

- **After Delete:** This event is generated when an object is deleted. Because this event occurs after an object is deleted many operations, such as **setValue**, will fail. The object would first have to be restored from the trash can before it could be modified. After Delete events trigger for all objects that are deleted, not just the objects that are selected for deletion. For example, deleting a folder triggers After Delete activators on requirements in the folder.
- **Before Open:** This event occurs when text is opened for modification in Microsoft Word. **Before Open** activators can be used to prevent Word from opening or to open the text read-only. Use this Tcl code to open the text read-only:

```
global actionContinue
set actionContinue stop
```

Use the Tcl error command to stop the transaction and prevent Word from opening. If you wish to suppress the normal Tcl error message use this syntax:

```
error STOP
```

- **After Relation:** This event is generated when a relationship to an object changes. Relationship changes include moving an object and adding or removed a note, trace link, or child.

Defining Session Modify Events

Session Activity events are events that are triggered by logging into or out of Systems Architect/Requirements Management. The following Session Modify events are supported by Systems Architect/Requirements Management:

- **After Login:** This event is generated when you first login Systems Architect/Requirements Management. This event is also generated when a re-login occurs after session timeout.
- **Before Logout:** This event is generated before you log out of Systems Architect/Requirements Management Logout can occur either by exiting the application or from a timeout due to inactivity.

Using A Login/Logout Activator

The After Login and Before Logout events are a unique and you must consider that in writing activators that are enabled for these events.

The After Login event occurs in the server shortly after the user successfully enters their UserID/Password in the login Web page. This is before the client is initialized, and it occurs without the context of any target object, or even any project.

Similarly, the Before Logout event occurs in the server at a time when Systems Architect/Requirements Management client is already in the process of shutting down. This event also has no context of a target object or project, regardless of what object might have been selected in the Systems Architect/Requirements Management client user interface at the time.

As a result, these conditions apply to After Login and Before Logout activators:

- They must be defined in the Systems Architect/Requirements Management Administration project.
- They must be set as activators for the User type definition object in that project.
- Since the Systems Architect/Requirements Management client is either not yet initialized or is shutting down, calls to API functions that depend on the Systems Architect/Requirements Management client are ineffective, including `displayMessage` and `createAction`. For the **After Login** activators, this problem can be overcome by using the `queueResult` action of the `setEnvironment` API. However, this is not effective for **Before Logout** activators.
- **After Login** and **Before Logout** activators can examine, create, and modify objects. However, the errors occurring during these times are not reported to users because the client is not in a normal running state.
- Unlike other activators, errors occurring in **After Login** activators do not cause the current transaction to be aborted because that would abort the entire session login action. This abort means no one can log in. However, such errors may cause other side effects and hence, the **After Login** activators should be tested well before enabling that event.

Change Approval Routing Events

The following events are change approval routing events:

- **Change Submitted:** This event is generated when a change is submitted. It performs as the following tasks:
 - Prepares to send an email with present text and proposed text (new text). Addresses the email to everyone on Change Approvers list and on Change Notifiers list of the Change Approval object
 - Prepares the text messages for both sets of emails. Sends email to everyone on the Change Approvers list of the Change Approval object. If sending email is successful, it freezes the object (Requirements/Building Block).
 - Prepares and sends emails to everyone on the Change Notifiers list.
- **Change Approved:** This activator is generated by Change Response activator. This activator performs the following tasks:
 - Makes a line entry to the table in the Change Approval object with information such as time, user name, comment, and action.
 - Attaches the MHTML content of the Change Approval object and the content of the parent object (Requirement or the Building Block) to the email.
 - Freezes the parent object if it is not frozen.
 - Sends an email to each approver and to the originator of the change approval request.
- **Change Rejected:** This activator is generated by Change Response activator. This activator performs the following tasks:
 - Makes a line entry to the table in the Change Approval object with information such as time, user name, comment, and action.
 - Attaches the MHTML content of the Change Approval object and the content of the parent object (Requirement or the Building Block) to the email.
 - Freezes the parent object if it is not frozen.
 - Sends an email to the originator of the change approval request.
- **Change Response:** This activator obtains the user name (tcrUser), the user input text, and the user action, either approved or rejected, from the database and the jsp form input.

This program enters the information such as user name, user comments, user's response, and the time in the table in the Change Approval object.

Import Events

An activator trigger on a folder can be used to prevent a document import. An after import event allows post processing of imported data.

- **Before import:** A before import allows you to cancel before a document import occurs. The type of import is passed in as a global. This code selectively prevents Word imports.

```
# Check if this is a Word import, could also be XML, AP233, PROJECT, SCHEMA,  
# STYLESHEET, MS_EXCEL or EXCEL_TEMPLATE  
if {$selected == "MS_WORD"} {  
  
    # throw error to cancel transaction and provide message back to user  
    error " Word import not allowed"  
  
}
```

- **After import:** An after import event allows post processing of imported data. Only the selected object appears in the change list because change processing is turned off for performance reasons. A change flag indicates the type of import.

Also because change list processing is disabled during import, Create activators are not triggered for most of the objects created by the import. Create events are only triggered for the topmost objects created, that is, those objects created directly in the folder where the import took place. Tcl can iterate from those objects down to visit all the objects created.

Storing Event Context

When an activator fires, information about the event that caused the activator to fire is passed into the Tcl environment. The event information is stored as Tcl global variables. The following global variables are set in the Tcl interpreter when activator execution begins.

- **currentObject:** The object that triggered the activator.
- **currentProject:** The project in which the **currentObject** resides.
- **event:** The type of event that triggered the activator.
- **tcrUser:** The user object of the current user.
- **tcrSubtype:** The subtype name for the object that is about to be created when a **Before Create** activator runs. The activator can reset this variable to a different subtype name, but it must be a subtype of the same base type.
- **selected:** A list of additional information that may be passed into activator when an activator is run using the **RequirementService runActivator** method.
- **changeList:** An array detailing exactly what changes were made to the **currentObject**. This list is not available in *Before* activators.
- **choiceDelimiter:** Separator used between the choices of a multi choice property.

Defining the Change List

Events, such as *After Modify*, do not indicate what was modified for an object. However, specific information about a change is kept in the change list. The change list array contains flags that indicate exactly what changes occurred. The change flags are the indexes of the Tcl global array named **changeList**. The presence of an index in the array indicates that type of change occurred. Some of the flags also have a value in the array. These values contain additional information about the change. For example, if a note is added to a requirement, an after relation activator fires. The `changeList` array has an *Add Note* entry whose value is the new note.

The following code detects that a note has been added and gets the note object:

```
set changeFlag "Add Note"
# get the list of change flags
set flags [array names changeList]
# check if notes have been added
if {[lsearch $flags $changeFlag] != -1} {
    # note has been added, get the note objects
    set notes $changeList($changeFlag)
}
```



The change flag constants are documented in the Javadoc for the **AccessEnum** class.

Defining Flags

Three types of change flags can be set in activators.

Defining Relation Flags

Relation flags are the change flags that may be set in an *After Relation* activator. All the add and remove flags have a list of the added or removed objects as the **changeList** array value.

- **Add Member:** A child or member object has been added (either created or moved in).
- **Remove Member:** A child or member object has been removed (either deleted or moved out).
- **Add Note:** A note has been added.
- **Remove Note:** A note has been removed.
- **Add Defining:** A trace link has been created that gives the current object a new defining object.
- **Remove Defining:** A trace link has been deleted that removes defining object.
- **Add Complying:** A trace link has been created that gives the current object a new complying object.
- **Remove Complying:** A trace link has been deleted that removes a complying object.
- **Move Object:** The current object has been moved; the old owner is stored as the **changeList** value.
- **Modify Owner:** The current object has a new owner.
- **Restore Object:** The current object has been restored from the trash can.

Defining Modify Flags

Modify flags are the change flags that may be set in an *After Modify* activator. They indicate the properties that have changed and, in some cases, the original property value.

- **Modify Name**
- **Modify ROIN**
- **Modify Text:** The original text is stored as the array value.
- **Modify Owner**
- **Modify Value:** Some system property has been modified.
- **Modify Security:** The security profile property has been modified.
- **Modify Property:** A user-defined property has been modified; a list of the modified user property names is stored as the array value.

Defining Delete and Create Flags

Delete and create flags are set when an object is deleted or created.

- **Delete Object**
- **New Object**

Using A Pick List Activator

Pick list activators list the database objects that can be used in the choice list of a dynamic choice property. The pick list activator returns any list of objects, based on user-supplied logic. You can create any number of pick list activators in a project. For information about setting a dynamic choice list for a choice property definition, see the *Systems Architect/Requirements Management Project Administrator's Manual*.

Three sample scripts for pick list activators are in the **Activators** folder of the **TcSE Administration** project as example pick list activators. For more information, see the methods [getObject](#), [getValue](#), [setObject](#), and [setValue](#) in chapter *Unsatisfied xref number*, *Unsatisfied xref title*.



See [Access Privileges for an Activator](#) for information about access privileges required to create, edit, or view an activator.

- **Choice List:** This example returns a list of folders in a particular Project. This list can be used as a choice list in a dynamic choice property.

```
set project $currentProject
set members [getList $project FOLDER_LIST]
```

- **Users:** This example returns a list of users in the current project. This list can be used as a choice list in a dynamic choice property.

```
set project $currentProject
set members [getList $project USER_LIST]
```

- **User Groups:** This example returns a list of users and user groups in the current project.

```
set project $currentProject
set members [getList $project USER_AND_GROUP_LIST]
```

Pass Owner to Tcl Context

Depending on how they are used, Dynamic Choice Properties may have a choice list that is constant across all objects, or choices specific to an object and its state. For example, the choices for an Assigned To property might be the list of all Users in a Project. Or, the list of Assigned To choices might depend on a Status property. As objects move through their life cycle, the list of people the work might be assigned to would move from designers to engineers to QA to production. Architect/Requirements supports both cases.

The Choice Property's *Pass Owner to Tcl Context* indicates which behavior applies; **No** when the property always has the same list of choices, or **Yes** when the choices may vary depending on the object. Setting this value to **No** allows Architect/Requirements to operate more efficiently by caching choice lists. Setting it to **Yes** requires the activator to be run every time a choice list is needed.

When *Pass Owner to Tcl Context* is set to **Yes**, the `currentObject` Tcl global variable is set to the object of interest whenever the choice activator is called.

When A Pick List Activator Gets Called

This section presents the different actions that cause the activator to be called, and the `currentObject` value for each case; either an object that has the property, the Property Definition, or the Project. In the latter two cases, the activator author must return a choice list that's suitable for all objects.

- On a Dynamic Choice Property (e.g. **Assign To**) when you select a Pick List activator in the **Update Choice list Dynamically** field, Pick List activator is called, and the result gets populated in the **Choice List** field.

`currentObject` = Choice Definition

- In the User module, when a **Dynamic Choice** property is opened for edit (the choice selection dialog is launched) the Pick list activator for the **Dynamic Choice** property is executed.

`currentObject` = Selected Object in the User Module

- In the User module when you try to set the value on the Object for this Dynamic Choice property:

`currentObject` = Selected object in the User Module

- In the Search module, selecting the Dynamic Choice Property itself:

`currentObject` = Current Project

- If a Dynamic choice property is used in a macro's Form Values, and that macro got executed, when user double clicks the **Value**, the Pick List Activator is executed

`currentObject` = Current Project

- In a Visio diagram when you edit Systems Architect/Requirements Management properties, when the Dynamic Choice property is edited for setting values, the Pick List Activator is executed.

`currentObject` = Object in Systems Architect/Requirements Management User module for which property is edited in the Visio diagram

- On exporting the Systems Architect/Requirements Management object(s) which has the Dynamic Choice Property to Excel Live:

`currentObject` = Current Project



Choice lists are not dynamic in live Excel. The Choice List activator is run once at the time of Excel export to generate one list of choices that is used for all objects with a value for that property.

- When value is set for Dynamic Choice Property on Systems Architect/Requirements Management object in Excel Live:
 - currentObject = Current Object for which value is set
- In XML Import and Excel Import, the Pick List activator is called once for each imported object that has a value to be set for a Dynamic Choice Property.

It is clear that Pick List activators can be called in many situations. It is the activator author's responsibility to insure that the activator will return a choice list that's valid for the currentObject in each case.

Pick List Activators and Performance

Activator authors must be aware that Pick List activators can have a significant negative impact on system performance. Unless the pick list activator author takes responsibility for testing the performance impact, it becomes a hidden burden on the system as a whole. The pick list overhead can impact many operations, but there is no way for a user to know the cause of the slow performance.

The degree of pick list impact on performance is influenced by:

- How often the activator is invoked
 - o The number of object types that use the pick list property
 - o How often instances of those types are encountered during user sessions
 - o Whether the property definition has Pass Owner to Tcl Context set or not
- The work that the activator does to generate its list
 - o Whether it just follows existing relationships, or uses search
 - o How many objects it handles while generating its result list
 - o Whether it uses transaction or session caching strategies
 - o The overall quality and efficiency of the Tcl logic

The easiest way for activator developers to find out how often a pick list activator is being called is to add a **displayMessage** call in the activator. It must be commented out for production use, but this can be a very useful diagnostic tool during pick list activator development. In addition to testing normal user browsing and object actions, also test multi-object cases such as copying a structure, and document, Excel and XML import.

The number of types and objects using pick list activators depends on the business needs. In planning the schema for a project, avoid using pick list properties unless there is a legitimate need for them. Also, avoid using them on a base type, if the property is really only needed on a limited set of subtypes.

If **Pass Owner to Tcl Context** can be set to **No**, the system can avoid running the pick list activator in some cases. This helps performance, but is not a cure-all. The other techniques listed here must also be considered.

If at all possible, search must be avoided in collecting the list of objects in pick list activators. Caching of the list is especially beneficial when it's necessary to use search to generate the list. If search is truly necessary, the search must be developed using the search module, and timed with actual production data. Different approaches for achieving the same search results can have dramatically different execution times.

Whether search is used or not, several different caching strategies can be beneficial. The appropriate caching approach depends on how dynamic the pick list choices are.

If the set of objects in a pick list seldom changes, then the best performance comes from calculating the list the first time it is needed in a session, and then caching it for the duration of the user session. The list can be saved using the **SaveData** option of the **setEnvironment** API. Each time the pick list activator is called, attempt to retrieve the list by making a **setEnvironment SaveData** call. If a list is returned, use it. Otherwise, calculate the list, save it for later reuse with a **setEnvironment SaveData** call, and return it for this use.

Often a pick list is based on some relatively static set of objects, but the exact subset returned by the activator may need to vary, depending on the state of the object the property applies to. In that case, use the **setEnvironment SaveData** approach to save the basic list. Then, each call to the activator would just filter that list down to the proper subset for that case.

If a pick list must be more dynamic, and always be sensitive to recent database changes, there is still some benefit to caching the choice list even within a single transaction. That's especially true if there is more than one choice property that uses the same dynamic list, or a closely related dynamic list based on the same underlying set of objects. For these cases, cache the list in a Tcl global variable, rather than using **setEnvironment SaveData**. Given that difference, the other comments in the two paragraphs above still apply.

Implementing Transaction Control

Both Before and After activators run within the same database transaction as the operation that triggered the event. If an uncaught error occurs in the Tcl, the transaction is rolled back and an error message is generated. To prevent the operation from occurring, you can throw a Tcl using the Tcl **error** command or set a global variable indicating the transaction should be aborted: set **actionContinue stop**.

The difference between these two methods is that the error command generates an error message while setting the global variable does not.



Setting **actionContinue** to **stop** does not immediately abort the Tcl execution. The flow of that activator continues. When the Tcl ends or returns, the Architect/Requirements server notices that the **actionContinue** flag is set. At that point, the current action is stopped and the transaction is aborted.

Creating a Tcl Where Clause

You can create a Tcl where clause to reference an activator to determine if an object meets the specified criteria or not. A Tcl Where clause can be used anywhere a where clause can be used.

Creating a Where Clause

To create the where clause:

1. Add a row to search criteria and indicate the command is a Tcl where clause.
Systems Architect/Requirements Management presents a list of activators. The activator must return a Boolean value. Only activators of subtype **Where Clause** appear in the list. Activators are gathered from both the current project and the administration project. Where clause activators in the administration project are available for use in all projects.
2. Select the activator to use.
Systems Architect/Requirements Management displays usage description. The description is a text property on activator.
3. Optionally, enter parameters to pass into activator.
4. Systems Architect/Requirements Management adds Tcl where clause to query.

Searching With a Tcl Where Clause

To run search with a Tcl where clause:

1. Click the **Search** button.
2. Systems Architect/Requirements Management runs search query and displays result.
Tcl where clauses are processed like other where clauses. The activator is run with the current object set as the object being evaluated. User-entered parameters are passed in as a global variable name parameter.

Tcl Where Clause Example

To verify that all functional requirements are met, you must ensure that all functional requirements are linked to a design element in the functional decomposition. This is done by creating a search called **Unallocated Requirements** that selects functional requirements that do not have complying objects. The result is a list of functional requirements that still need to be linked to the design. This search is inadequate to verify that the requirements are linked to elements in the functional decomposition as the **Complying Objects Count** property displays only complying objects. A Tcl where clause is required to examine the complying objects and verify that at least one functional design element (a building block with a **Functional Design** subtype) is included.

Unallocated requirement report script does not verify if the allocation is to the functional design. For example,

```
SELECT Functional Requirement
      WHERE Complying Objects Count = 0
```

Unallocated requirement report script with Tcl where clause verifies if the allocation is to a specific type. For example,

```
SELECT Functional Requirement
      WHERE hasComplying Functional Design
```

Tcl script for the **hasComplying** where clause activator. For example,

```
proc hasComplying {type} {
    foreach obj [getList $currentObject COMPLYING_OBJECT_LIST] {
        if {[getValue $obj Subtype] == $type} {
            return true
        }
    }
    return false
}
hasComplying $parameter
```

This activator can be used in a where clause to prevent an object from being added to the results more than once in a **FOREACH ADD DEEP** command.

```

# initialize list to keep track of objects already in result
global visitedList
if { ! [array exists visitedList] } {
    set visitedList(0) 0
}
# check if the the current object is already in the list
set loid $currentObject
set newObject 1
catch {
    if { $visitedList($loid) == 1 } {
        # Found object in the list
        set newObject 0
    }
}
# add current object to visited list
set visitedList($loid) 1
# return result, 0 if this object was already in the visited list, 1 if the object
was
#not in the visited list
set newObject

```

Tcl Global Variables in Where Clause Activators

When a where clause activator is run during a search, the following Tcl global variables are set:

- **currentObject** is the object that is examined by the where clause for possible inclusion in the search result.
- **superior** is the object that the current object is indented under in the search result.
- **relation** names the relation that caused the superior object to be in the search results. For example, if the superior was added to the search results by an ADD Members statement, the relation value is **MEMBER_LIST**.
- **level** is the indentation level of the superior object, where the left-most or top-level objects are at level 0. If the current object is added to the report, it is added one level above the level specified by **level**.
- **parameter** is a string holding the text that is entered in the **Parameters** field when adding the activator reference to the search script.

If a script's initial SELECT statement has a where clause with an activator, there is no superior object in this case; hence the values of **superior**, **relation**, and **level** are as listed here:

- **superior** has the same value as **currentObject**
- **relation** is empty
- **level** is 0

As is customary with Tcl global variables, if referenced from within a Tcl procedure, these variables must be declared as global:

```
global currentObject superior relation level parameter
```

Writing Activators When Objects are Deleted, Restored, or Discarded

Writing activators that behave correctly can be difficult when objects are deleted to the Recycle Bin, restored from the Recycle Bin, or discarded from the Recycle Bin.

When an object is initially deleted, the possible activator events are:

- Before Delete on the specific object(s) being deleted
- After Delete on that object, plus all descendents and attachments deep
- After Relation (defining/complying) on any of the objects that had trace links to objects that weren't deleted

If an object is restored from the Recycle Bin, possible activator events are:

- After Relation (modify owner) on the one object that was restored
- After Relation (defining/complying) on any of the objects that had trace links to be restored

When an object is discarded from the Recycle Bin, either individually or by emptying the Recycle Bin, there are no possible events.

You are not allowed to modify objects in the Recycle Bin. So, other than the restore action, there should not be modification events on objects once they are in the Recycle Bin.

It is the responsibility of the activator author to monitor all cases where the activator may fire. Or, at least catch errors when they occur. Pay close attention to your After Relation events, since they can fire as objects go into the Recycle Bin and when they are restored from it.

The easiest way to tell if an object is in the Recycle Bin is to test the **Date Deleted** property:

```
if {[getValue $obj "Date Deleted"] == "" } {  
    ...not deleted...} else {...is deleted...}
```

The test works for objects visible in the Recycle Bin, and all their descendents and attachments as well.

TC_XML Export Activator

When you export data to TC_XML, an activator is triggered. The activator configures the data being exported. The activator is triggered regardless of how you initiate the export. You can initiate the export in one of the following methods:

- Using the Architect/Requirements client
- Using the command prompt
- Using the Tcl **exportDocument** command

You can locate the activator by its name. Typically, the name of activator is **TcXmlConfig**. Architect/Requirements first searches for the activator in the current project. If a **TcXmlConfig** activator is not found then Architect/Requirements searches the administration project. This activator is for configuring TC_XML export. You can configure the activator to:

- Exclude unwanted data from the export
- Adjust type and property names
- Assign Teamcenter item IDs
- Specify the versions to export

The following sections provides information on the variables used in configuring the activator.

Assigning Teamcenter Item IDs

Item ID is the unique identifier assigned to Teamcenter objects. It is similar to a **ROIN** in Architect/Requirements. Teamcenter requires item IDs to be specified for TC_XML imports. You can specify the item ID in different methods. The method used is specified in the **itemIdRule** variable. You can use one of the following values for the **itemIdRule** variable to specify the method to be used:

- File
- Counter
- **ROIN**

Item IDs can be generated by Teamcenter and written to a file. Siemens PLM Software recommends using this method as it guarantees that the item IDs are unique and comply with Teamcenter naming conventions. To use the item IDs from a file, set the **itemIdRule** variable to **File** and specify the name of the file to use for which object type is specified in the **ItemIdFiles** map. Following is an example of assigning Teamcenter Item IDs using the file method:

```
set itemIdRule File
set ItemIdFiles {
  Requirement {C:/temp/reqIds.txt}
  Paragraph {C:/temp/paraIds.txt}
  Note {C:/temp/noteIds.txt}
  Folder {C:/temp/folderIds.txt}
}
```

The concept of inheritance applies when assigning Teamcenter Item IDs. If a requirement subtype name is not found in the map, Architect/Requirements searches the parent type.

Architect/Requirements can also generate item IDs. To do so, specify the item ID prefixes in the **itemIdPrefixes** list. Additionally, specify the counter format and starting number using the **itemIdStart** variable. Following is an example of assigning Teamcenter Item IDs using the Counter method:

```
set itemIdRule Counter
set itemIdPrefixes {
  Requirement "REQ-"
  Paragraph "REQ-"
  Note "CSTMNOTE-"
  Folder "REQSPEC-"
}
set itemIdStart 000100
```

You can also use the Architect/Requirements **ROIN** as the item ID in Teamcenter. To use the Architect/Requirements **ROIN** as the item ID, set **itemIdRule** to **ROIN**. If the **ROIN** is not available, the **LOID** of the type is used. You must ensure that the naming rules for your types in Teamcenter allow the Architect/Requirements **ROIN** format. Following is an example of assigning Teamcenter Item IDs using the **ROIN** method:

```
set itemIdRule ROIN
set itemIdPrefixes {
  Note "CSTMNOTE-"
  Folder "REQSPEC-"
}
```

Excluding data from export

Removing unwanted information from the export can simplify mapping and make the migration process faster. You can specify the object types and properties that are not required, using the following variables:

- **excludeTypes**
Specifies the object types to be excluded. Unsupported types are automatically excluded.
- **excludeSysProps**
By default, system properties that are unlikely to be useful in Teamcenter are excluded. Setting this variable replaces the default list. Hence, you have complete control over the system properties that are exported.
- **excludeProps**
Additional list of user defined or system properties to be excluded.

Following is an example of removing unwanted information from the export:

```
set excludeTypes [list {Change Log} ImageNote DataDictionary]
set excludeProps [list {Change Time} {Change User} {My Property}]
```

Adjusting type and property names

It is likely that the type and property names in Architect/Requirements are not an exact match of the corresponding names in Teamcenter. Teamcenter requires prefixes for custom type and property names while Architect/Requirements does not have such requirements. A mapping step is provided in the migration process to handle such mismatch. However, you can perform some of the mapping during the export process. You can add prefixes to type and property names which simplifies the mapping process.

You can also set up maps to specify the names to use in the export. You can use the following variables to set up the mapping:

- **typePrefix**
Specifies the prefix for all object types.
- **sysPropPrefix** and **userPropPrefix**
Specifies the prefix for system and user defined property names.
- **typeMap**
Establishes a map for the name to use for a given object type.
By default, **typeMap** maps the out of the box Architect/Requirements types to the corresponding out of the box Teamcenter type.
- **propMap**
Establishes a map for the name to use for a given property. If a type or property name is included in a map, no prefix is added to it.

Following is an example of mapping the type and property names:

```
set typePrefix "Tcr4"  
set sysPropPrefix "Sys4"  
set userPropPrefix "Tcr4"  
  
set typeMap {  
  Requirement "Tcr5MyReqType"  
  DerivedRequirement "Tcr4DerivedReq"  
}  
  
set propMap(Project) "Tcr4TcSEProject"
```

Specifying the versions to export

You can control the requirement versions that are exported using the **versionRule** variable. Following are the valid values for **versionRule**:

- **Effective**
Exports the currently effective version.
- **Baseline**
Exports the current version and all baselined versions.
- **All**
Exports all versions.

By default, only the currently effective version is exported. If multiple versions are exported, only the current version is configured into a requirement specification during the import. For baseline exports, snapshot objects named after the baseline are created to assist in configuration of the baselines.

Following is an example of specifying the version to export:

```
set versionRule Baseline
```

LOID Property

LOID is the unique database identifier for Architect/Requirements objects. Mapping **LOID** to a property in Teamcenter is useful for connecting Teamcenter objects to the corresponding Architect/Requirements object. The mapping is required for setting rich text content. The **loidProperty** variable contains the name of the Teamcenter property that stores the **LOID** value.

Following is an example of setting the **LOID** property:

```
set loidProperty Tcr4LOID
```

Activator Examples

This section provides examples of activators. You can copy and paste the examples in this section and use them directly. This is possible because the examples here are dependent on the `currentObject`.

Determining Requirements With The Same Name

The following example (figure 6-1) shows how many requirements have the same name in the present project- Activator Name **Same Name Requirements**. You set the Events property of the activator to **After Modify**. Then, enter this program in an activator and set the activator property of the Requirement Type Definition to **Same Name Requirements**.

```
proc Iterator { x name} {
  global item
  set members [getList $x MEMBER_LIST]
  foreach z $members {
    set KK [getValue $z "Type Name"]
    if { $KK == "Requirement" } {
      if { [getValue $z Name] == $name } {
        set item [expr $item+1]
      }
    }
  }
  Iterator $z $name
}

return $item
}

set item 0
set strName [getValue $currentObject Name]
set subMembers [Iterator $currentProject $strName]
displayMessage " $item Requirements have the same name."
```

Figure 6-1. Activator Name, Same Name Requirements

Creating a Note

The following example (figure 6-2) creates a note before a requirement is opened for editing and copies its text property to the note.

```
set strTypeName [getValue $currentObject "Type Name"]
if {$strTypeName == "Requirement" } {
    #Creates a note
    set noteLOID [createObject oldText $currentObject "Note"]
    #Gets Text property of current object
    set strText [getValue $currentObject "Text"]
    #set text property of the note.
    setValue $noteLOID Text $strText
}
```

Figure 6-2. Creating a Note

Creating An After Delete Activator Event

The following activator example (figure 6-3), when set to activate After Delete for a requirement, moves the deleted requirement to a folder named **DeletedRequirements** if this folder exists or creates this folder if it does not exist.

```
# locate or create the deleted requirements folder
proc getDeletedFolder {} {
    global currentProject
    set folders [getList $currentProject MEMBER_LIST]
    foreach folder $folders {
        if {[getValue $folder Name] == "Deleted Requirements"} {
            return $folder
        }
    }
    set folder [createObject "Deleted Requirements" $currentProject Folder]
    return $folder
}

# do not move children of the deleted requirement
if {[getValue [getValue $currentObject Owner] "Type Name"] == "TrashCan"} {

    # locate or create a folder for deleted requirements
    set folder [getDeletedFolder]

    # Undelete the requirement and place in folder
    restoreFromTrashcan $currentObject $folder

    # Change to deleted subtype so it can be filtered from search
    setValue $currentObject Subtype Deleted

    displayMessage "'[getValue $currentObject Name]' has been
        move to deleted objects folder"
}
```

Figure 6-3. After Delete Activator Event

Using createAction FileDownload

This section provides several examples using createAction FileDownload to download a file from the server to the Systems Architect/Requirements Management client workstation, and optionally open it in an application.

Producing an HTML File

The following example (figure 6-4) shows how to produce an HTML file from a report and launch a Web browser to view the report results.

```
set serverFile [runReport $report $template $startingObject $live]
  #run the report using parameters
set application Default #set IE as output application
createAction FileDownload [list $serverFile $application]
  #download the exported file from server and open using IE
```

Figure 6-4. Producing an HTML File

This example cannot be copied and pasted directly because it depends on the parameters, reportID, templateID, startingObjectID and Boolean live flag.

E-mailing Requirements or Producing PowerPoint Project from Requirements

The following example (figure 6-5) can be used to produce a PowerPoint project, one slide per Requirement from a selected folder. It can also be used to select a set of Requirements and E-mail them using Microsoft Outlook.

The output is determined by the application you select in the chooser. If you select PowerPoint, the downloaded file is opened in PowerPoint. If you select Outlook, a new E-mail window opens, and the downloaded file is attached.

```
set objectList $currentObject #take current object
set serverFile [exportDocument $objectList MS_WORD] #export this object to word
set application Chooser #choose the application
createAction FileDownload [list $serverFile $application]
  #download the exported file from server and open using selected application
```

Figure 6-5. E-mailing Requirements or Producing PowerPoint Project from Requirements

Using createAction RunJava

You can create custom Tcl code and custom Java code that run on the client JVM. You can use the custom code to interface with other applications or create custom interfaces such as a Java form for user input. **createAction RunJava** allows external programs to interface with Architect/Requirements using the C# interface. It allows the external Java code to be run from within Architect/Requirements through the Tcl code. It also allows identification of Java classes and execution of methods in the classes from the Tcl code of activators, macros, or custom menu items.

Setting up to run Java code in Architect/Requirements client

The location of the **.jar** file must be set by the administrator using the **Package.Location Web Application Configuration** parameter. The **jar** file must contain a method named **connectTcSE** with the prescribed method signature.

To prepare the Java code to run in the Architect/Requirements client:

1. Create a file named **TestRunJavaClass.java** containing the source code.
2. Compile the Java file. The class path must contain the three jar files from the Architect/Requirements client installation directory:
 - **tcrPackages**
 - **tcrShared**
 - **rcf**

The following code is an example of the steps required to compile the Java file:

```
set TCSE _DIR=C:\Program
Files\SiemensPLM\Teamcenter\SystemEngineering\Release_9.1
set CP=%TCSE _DIR%\tcrPackages.jar;%TCSE _DIR%\tcrShared.jar;%TCSE
_DIR%\rcf.jar
javac -cp %CP% TestRunJavaClass.java
```

3. Add the compiled class to a jar file.

```
jar -cf test.jar TestRunJavaClass.class
```
4. Update the **Package.Location Web Application Parameter** to reference the jar file path.

Example:

C:\test\test.jar

The ClientJavaAPI class

The **ClientJavaAPI** class is the beginning of a client interface.

To call an Architect/Requirements macro from the external Java code, the class file must import the **ClientJavaAPI** class found in the **tcrPackages.jar** file. The **tcrPackages.jar** file can be added to the **classpath** variable. You can also unzip it and use the **ClientJavaAPI** class file in the development area.

runMacro() Method

The **runMacro()** method of the **ClientJavaAPI** class is used to call a macro from the external Java code.

A basic method calls the macro by name by passing a space delimited string of LOIDs for the macro. For example:

```
public String runMacro(String macroName, String objectIDs)
```

Another method uses the macro name and the macro LOID along with the object IDs. In this case, if the macroID is passed and is not null, the macroName is not used. For example:

```
public String runMacro(String macroName, String macroID, String objectIDs)
```

Another extensively used method uses values passed into the macro from the form mechanism, passing in the form properties and the associated values. If the macroID is passed and is not null, then the macroName is not used. For example:

```
public String runMacro(String macroName, String macroID, String objectIDs, String[] formProps, String[] formValues)
```

getTempDir() Method

If required, you can get a temporary directory from Architect/Requirements. The **getTempDir()** method returns a string containing the system temporary directory used by Architect/Requirements.

The **ClientJavaAPI** class must be available to the custom package. The **ClientJavaAPI** class must be imported with the following statement:

```
import com.edsplm.tc.req.client.shared.api.ClientJavaAPI;
```

It must be instantiated with the following statement:

```
protected ClientJavaAPI javaAPI = new ClientJavaAPI();
```

After instantiating the **ClientJavaAPI** class, you can use it through the following code:

```
result = javaAPI.runMacro(macroName, objectIDs);
```

OR

```
result = javaAPI.runMacro(macroName, null, macroParameters, formProps, formValues);
```

Java Development Environment

It is important to set up everything correctly, extract the files to the proper location, get the compiler setup to compile correctly and have it compile in the right location so that Architect/Requirements can find it.

Siemens PLM Software recommends creating subclasses of the delivered classes so that any potential changes made are not overwritten when a new Architect/Requirements deployment is made.

Complete the following tasks before you begin the customization:

- Set up the development environment, preferably with an IDE. You can use the Eclipse IDE from www.eclipse.org.
- Setup **Classpath** correctly.
- Set up packages correctly.
- Prepare to package the code in a **jar** file for distribution.
- Understand how to extend the delivered class to create new customized behavior.

The goal is to create a **jar** file for distribution or to replace the **tcrPackages.jar** file. You can start with unzipping the **tcrPackages.zip** file that contains the two external Java files in the working directory.

Here are additional recommendations to create the **jar** file.

- The new custom Java code must be in its own package.
- The Java package does not require a **main()** method; it is a collection of methods.

- The methods must be called from Architect/Requirements to be an extension of the Architect/Requirements client.

Unzip the **tcrPackages.zip** file and become familiar with the Java source code for the **TcSECMInterface.java**. It is the best example of the customization toolkit.

Code example

This section shows the source code for a complete Java program related to the examples in [createActionRunJava](#) in chapter *Unsatisfied xref reference*, *Unsatisfied xref title*.

```

/* -----
 * This software and related documentation are proprietary to
 * UGS Corp.
 * (c) 2007 UGS Corp. All rights reserved.
 * -----
 */
import javax.swing.*;
import com.edsplm.tc.req.client.shared.api.ClientJavaAPI;

/**
 * This is a test class, to be used to test and understand the RunJava createAction.
 */
public class TestRunJavaClass
{
    ClientJavaAPI javaAPI = new ClientJavaAPI();

    // Can be called via RunJava - main method is not required, but may be used
    public static void main (String args[]) {
        printHelloWorld(args);
        printArgs(args);
    }

    // Can be called via RunJava - Run a macro on the server
    public void runTestMacro(String[] theArgs) {

        // specify macro name
        String macroName = "HelloWorldMacro";

        // run the macro and capture its return value, pass it the first
        // argument from the CreateAction RunJava call
        String result = javaAPI.runMacro(macroName, theArgs[0]);

        // display the return value
        showDialog(result);
    }

    // Can be called via RunJava - static method can be called
    public static void printHelloWorld(String someArgs[]) {
        // nothing is done with someArgs[]
        printIt("printHelloWorld");
        // popup dialog for user to click
        showDialog("Hello World!");
    }

    public void printAllArgs(String[] theArgs) {
        // this is the public method to show all arguments
        // it calls the private method that does the work
        printArgs(theArgs);
    }

    // Can be called via RunJava - no return, no parameters
    public void doNothing() {
        printIt("doNothing");
        // popup dialog for user to click

```

```
    showDialog("doNothing");  
}
```

```

// Can be called via RunJava - prints first parameter sent, returns nothing
public void doNothing(String[] someArgs) {
    printIt("doNothing(String[])");
    System.out.println(" > someArgs[0] is " + someArgs[0]);
}

// Can be called via RunJava -
public String doNothingString() {
    printIt("doNothingString");
    // prints this line to the log
    return "prints this line to the log";
}

// Can be called via RunJava -
public String[] doReturnStringArray() {
    printIt("doReturnStringArray");
    // prints these values to the log
    String[] returnValues = new String[]{"returnMsg1", "returnMsg2",
"returnMsg3"};
    return returnValues;
}

// Cannot be called directly via RunJava - must have this method!
// This method is called "automatically" every time another method is called.
// The intent and purpose of this method is to provide the necessary parameters
needed
// in order for the external application to connect back with TcSE. To connect
back to
// the TcSE client, the first three parameters may be used; to connect back to the
// TcSE server, the last may be used.
public String connectTcSE(String controllerPath, String serverIP, String
socketServicePort,
                        String sessionID) {
    System.out.println("here in connectTcSE, values are: " + controllerPath + ", "
        + serverIP + ", " + socketServicePort + ", " + sessionID);
    return "Finished connectTcSE";
}

// Cannot directly call this method, itâ€™s private. Method to call must be
public.
private void thisIsPrivateMethod(String[] someArgs) {
    System.out.println("TestRunJavaClass.thisIsPrivateMethod: cannot call this");
}

// Cannot directly call this method, itâ€™s private. Method to call must be
public.
private static void printArgs(String[] someArgs) {
    printIt("printArgs");
    if (someArgs == null) {
        System.out.println(" > args is null");
    } else if (someArgs.length < 1) {
        System.out.println(" > args length is zero");
    } else {

```

```

        int idx = someArgs.length;
        for (int i=0; i<idx; i++) {
            System.out.println(" > arg[" + i + "] = " + someArgs[i]);
        }
    }
}

// Cannot call this method, wrong return type. Must return void, String or
String[].
public int returnInt() {
    System.out.println("TestRunJavaClass.returnInt: cannot get here, wrong return
type");
    return 41;
}

// Cannot call this method, wrong number of parameters. Must have no parameters
// or one String[] parameter.
public void doSomething(String[] someArgs, String twoParams) {
    System.out.println("TestRunJavaClass.doSomething: cannot get here,
wrong number of parameters");
}

// Cannot call this method, wrong type of parameter. Must have no parameters or
// one String[] parameter
public void doSomething(int someArgs) {
    System.out.println("TestRunJavaClass.doSomething: cannot get here,
wrong type of parameter");
}

// Cannot call this method, it's private
private static void printIt(String methodName) {
    String msg = "inside method TestRunJavaClass." + methodName;
    System.out.println(msg);
}

// Cannot call this method, it's private
private static void showDialog(String text) {
    JOptionPane.showMessageDialog(null, text);
}
}

```

Running an Activator From the Command Line With the tcradmin Script

You can enter the **tcradmin** script on the command line to run an Architect/Requirements activator.

The following example shows the **tcradmin** syntax:

```
tcradmin -action runActivator -script <ActivatorID> -selected <listOfParameters> -  
user <username> -password <password> [-key <installationKey>]
```

Table 6-1 describes the **tcradmin** arguments.

Table 6-1. tcradmin Arguments for Running an Activator From the Command Line

Argument	Description
-script	LOID of the activator to run.
-selected	Comma-separated list of database object LOIDs or other information.  This information is passed to the activator as the global variable selected .
-user	Architect/Requirements user name under which you want to log in to the server.
-password	Architect/Requirements password for the user name under which you want to log in to the server.
-key	Web server installation identifier used to look up Web application parameters such as ImportExportDir .  This argument defaults to <i>machinename</i> + "1", which is correct in most circumstances. This argument may be needed if there are no or more than one Architect/Requirements Web servers installed on this machine.

Using Macros

A macro is a type of activator that is run from a menu command rather than being triggered by an event. A macro specifies what input values, if any, it requires and the user interface will prompt for those values when the macro runs. The objects selected in the user interface are also passed into the macro.

Creating a Macro



See [Access Privileges for an Activator](#) for information about access privileges required to create, edit, or view an activator, which are also applicable to a macro.

1. Select the activator folder in the administration module and then choose **New** then **Activator** from the **File** menu.
2. Edit the activator's **Subtype** property by setting it to **Macro**.
3. If the macro requires user input, edit the activator's **Form Values** property, and select the values for which to prompt the user.
4. Open the activator and enter the Tcl script.

Form Values

The **Form Values** property on a macro allows you to specify the values to prompt the user for when the macro runs. Form Values is a multichoice property whose choices are the names of all the properties in the project. If the **Form Value** property is not empty, the user interface displays a dialog window to accept user input when the macro runs. Each property specified in **Form Values** has an entry field in the dialog window. The entry fields accept values based on the property type, text, choice, numeric, or date. The values entered by the user are passed into the Tcl environment using a global array name **tcrProperties**. If none of the existing properties is appropriate for the macro, you can create new properties.

Tcl Environment

Information from the user is passed into the Tcl interpreter as global variables. In addition to the normal variables passed into activators, there are two extra variables for macros:

- **selected:** List of objects selected in the user interface.
- **tcrProperties:** Array of values entered by the user. The array index is the property name. The value is what the user entered for the property. There is an entry for each property specified in the **Form Values**.

Event related variables that are passed into activators are not available in macros; these are the **currentObject**, **event** and **changeList**.

Running a Macro

Macros are run from the requirements module as follows:



For additional information, see the *Systems Architect/Requirements Management User's Manual*.

1. Choose **Run Macro** from the **Tools** menu.
The command displays a list of all the macros in the project.
2. Select the desired macro and click **OK**.
3. If the macro requires user input a dialog box appears. Enter values for each property in the macros **Form Values**.

Macro Examples

This section provides examples of macros.

Setting a Property on Several Objects at the Same Time

Figure 6-6 demonstrates setting the **AssignedTo** property on several objects at once. The example assumes an **AssignedTo** property exists in the project and that the macro's **Form Values** is set to **AssignedTo**.

```
# get the value the user entered for AssignedTo
set assigned $tcrProperties(AssignedTo)

# iterate over the objects selected in the user interface
foreach object $selected {
    # set the AssignedTo property to the user entered value
    setValue $object AssignedTo $assigned
}
```

Figure 6-6. Setting a Property on Several Objects at the Same Time

Setting Several Properties on Multiple Objects

Figure 6-7 demonstrates setting any number of user entered properties on the selected objects. The properties are specified in the macro's **Form Values**.

```
# iterate over the objects selected in the user interface
foreach object $selected {
    foreach property [array names tcrProperties] {
        # set the assignedTo property to the user entered value
        setValue $object $property $tcrProperties($property)
    }
}
```

Figure 6-7. Setting Several Properties on Multiple Objects

Creating a Derived Requirement

Figure 6-8 demonstrates how to create a derived requirement for the selected source requirement. The derived requirement has a trace link to the source and some property values are set automatically. The example assumes a **Derived Requirement** subtype exists in the project.

```
set sourceReq $selected ; # assumes 1 requirement is selected

# create a derived requirement owned by the selected requirement
set derived [createObject "" $sourceReq "Derived Requirement"]

# copy property values from the source to the derived requirement
setValue $derived Name [getValue $sourceReq Name]
setValue $derived MHTML [getValue $sourceReq MHTML_FULLL] ; # sets the
body text

# set the assignedTo property to the current user
setValue $derived assignedTo [getValue $usrUser Name]

# create a trace link to the derived requirement
createLinks $sourceReq $derived "Complying"
```

Figure 6-8. Creating a Derived Requirement

Collecting Metrics on Requirements

Figure 6-9 iterates over requirements, beneath the selected object, and gather statistics. The example assumes a status property is assigned to requirements in this project.

```
# get the requirements in the selected project or folder
set reqs [search $selected "*" "" {RequirementDB}]

# iterate over the requirements and check their status
set complete 0
set total [llength $reqs]
foreach req $reqs {
    if {[getValue $req Status] == "Completed"} {
        incr complete
    }
}

displayMessage "Status of requirements in [getValue $selected Name]"
displayMessage "Total requirements: $total"
displayMessage "Completed requirements: $complete"
displayMessage "Percent complete: [expr $complete * 100.0 / $total]"
```

Figure 6-9. Collecting Metrics on Requirements

Working with Shortcut Objects

Shortcuts act as a separate occurrence of an object in a different location. When processing objects using the Systems Architect/Requirements Management API methods, you may encounter shortcuts. For example, the `getList MEMBER_LIST` call on a folder or other parent object returns shortcut members along with the rest of the true objects.

Shortcuts act in many ways like the actual object. The API forwards most **getValue**, **setValue**, and **getList** methods to the actual object. In some cases, it may be necessary to determine that you are working with shortcuts and handle them specially.

For this special handling, shortcuts support several properties, such as:

- **Shortcut** indicates whether this object is a shortcut ("true") or is not ("").
- **Original Object** is the LOID of the actual object.
- **Shortcut Options** is a property with flags to choose the behavior of shortcuts, such as whether to allow the shortcut to be expanded to show the actual objects children.

The following code recognizes that an object is a shortcut and gets the actual object:

```
if {[getValue $obj Shortcut] == "true"} {  
    set actualObj [getValue $obj "Original Object"] }
```

There are some properties that are retrieved directly from the shortcut, instead of being forwarded to the actual object in the **getValue** and **setValue** methods. A shortcut's local properties are:

- **Owner** is the LOID of the shortcut's owner.
- **Owner Name** is the name of the shortcut's owner.
- **Security Profile** is the profile on the shortcut itself, if any.
- **WhereUsed Object Count** is the number of objects that reference the object. For shortcuts, this count is typically zero.
- **Used Object Count** is the number of objects that the object references. For shortcuts, this count is always one.
- **Cross Project Link** is true if the shortcut references an object in a different project and is false if the referenced object is in the same project.
- **Project** is the name of the project that owns the shortcut.
- **Full Name** is the full name of the shortcut, including the object that owns it.
- **Number** is the hierarchy number of the shortcut. For example, **2.3.7**.
- **Local Number** is the last portion of the hierarchy number of the shortcut. For example, **7**.

Some **getList** keywords are processed directly by the shortcut. All other **getList** keywords are retrieved from the original object.

- **MEMBER_LIST** is empty if the shortcut does not inherit children. Otherwise, the **getList** method is forwarded to the primary object to retrieve its members.



A shortcut never has children of its own. It can (optionally) inherit the children of the primary object that it references. In that case, there are no extra shortcut objects for those children. The actual children of the primary object are displayed below the shortcut in

the Systems Architect/Requirements Management client. The small red-arrow icon indicates they were reached via a shortcut. If a shortcut inherits the primary object's children, the **getList** method with **MEMBER_LIST** keyword on a shortcut returns the exact same objects as the **getList** method on the primary object.

- **WHERE_USED_LINK_LIST** is a list of where-used links to the object. This list is typically empty for shortcuts.
- **WHERE_USED_OBJECT_LIST** is a list of objects from following the where-used links from the object. This list is typically empty for shortcuts.
- **USES_LINK_LIST** is a list of reference links from the object. This list always has one entry for shortcuts, that is the link to its primary object.
- **USES_OBJECT_LIST** is a list of objects referenced by the object. This list always has one entry for shortcuts, that references its primary object.

Activators are not triggered on shortcut objects because there is no type definition for the shortcut type. However, many actions on a shortcut are applied to the original object. These actions trigger events on the original object, not the shortcut.

Shortcuts cannot handle the Create, Before Delete, and After Delete events because shortcuts cannot have their own activators. However, when shortcuts are created and deleted, there are some observable events on other objects. They are:

- When a shortcut is created, the After Relation (Add Member) event on the shortcut's owner and the After Relation (Add Where Used) event on the object the shortcut references.
- When a shortcut is deleted, the After Relation (Remove Member) event on the shortcut's owner.

There is no event on the referenced object when a shortcut is deleted. There is also no event on the shortcut's owner if it is restored from the recycle bin.

Working with Reference Links

When a reference is inserted into the text of a requirement or note using the **Copy Reference Link** command, a symbolic reference link object is created. This section describes how to access and manipulate these references.

The **getList** keywords related to references are (see `DataBean.LIST`):

- **WHERE_REFERENCED_LIST** is the list of the requirements and notes that have references to the input (this) object.
- **WHERE_REFERENCED_LINKS_LIST** is the list of symbolic links that reference the input (this) object.
- **REFERENCING_LIST** is the list of objects the input (this) note or requirement references.
- **REFERENCING_LINKS_LIST** is the list of symbolic reference links from the input (this) note or requirement.

The **getValue** keywords related to symbolic link objects are (see `DataBean.PROPERTY`):

- **START_OBJECT** is the requirement or note containing the reference.
- **END_OBJECT** is the referenced object.
- **PROPERTY** is the name of the referenced property.

References to Versioned Objects

When a reference is made to a versioned object, Systems Architect/Requirements Management decides which version of the object to use. There are two modes of behavior. If the **Promote References** option is selected in the project's **Project Settings** property, then references use the effective version. Otherwise, references use a specific version. In either case, there is one symbolic link object that points to a specific version. When **Promote References** is enabled, the reference is shifted to the effective version dynamically.

The **getValue** and **setValue** keywords that apply to symbolic links when versions are involved are:

- **USED_OBJECTS**: Gets the referenced object (like **END_OBJECT**). **END_OBJECT** returns the specifically referenced version. **USED_OBJECTS** returns the effective version of the referenced object if **Promote References** is enabled.
USED_OBJECTS is also supported in **setValue** which allows adjusting which version is specifically referenced.
- **REFERENCE_SETTINGS**: By default, reference links use the behavior specified in the **Project Settings** property. The default behavior can be overridden for a specific symbolic link by setting the **REFERENCE_SETTINGS** property with **Promote References** or **Fixed References** as value. A blank value indicates the default project behavior.

The **getList** keywords that apply when versions are involved are:

- **WHERE_REFERENCED_REAL**: Returns the requirements and notes that reference the input (this) object (like **WHERE_REFERENCED_LIST**). **WHERE_REFERENCED_LIST** takes the **Promote References** option into account where as **WHERE_REFERENCED_REAL** returns specific references.
- **WHERE_REFERENCED_LINKS_REAL**: Returns symbolic links that reference the input (this) object (like **WHERE_REFERENCED_LINKS_LIST**).

WHERE_REFERENCED_LINKS_LIST takes the **Promote References** option into account where as **WHERE_REFERENCED_LINKS_REAL** returns specific references.

- **REFERENCING_REAL**: Returns the objects the input (this) note or requirement references (like **REFERENCING_LIST**). **REFERENCING_LIST** takes the **Promote references** option into account where as **REFERENCING_REAL** only returns specific references.

Chapter 7: Using Icon Overlay

Icon overlay functionality provides a mechanism to add custom overlays to icons. You can use the functionality to overlay an existing icon or completely change an icon.

Container objects such as folders, building blocks, requirements, and projects have a new **hidden** property.

If there is a value on this property, and if it is the LOID of a type definition, then the icon that is set for that type definition will be used as an overlay to the current icon. The custom overlay is the first overlay, followed by any possible out of the box overlays such as subtypes, variants, or frozen.

A new **icon** type definition has been added. You can create subtypes of the **icon** type definition for your icons.

Following is a sample code for Icon Overlay using Tcl:

```
# get the LOID of the icon overlay
set name "BigRedEx"
# find type def by iterating over all type defs
foreach def [getList $currentProject OBJECT_TYPE_LIST] {
    if {[getValue $def Name] == $name} {
        set iconOverlay $def
        break
    }
}
# the code must ensure that icon was found before continuing
setValue $selected "Icon Overlay" $iconOverlay
```

You can create activators (after modifying) that can set the **Icon Overlay** property based on the criteria that you determine.

For example:

- If it is assigned to Jerry and it is complete then make it a big green **J**.
- If it is overdue, then mark it with a big red exclamation mark.

A sample icon overlay is given in Figure 7-1.

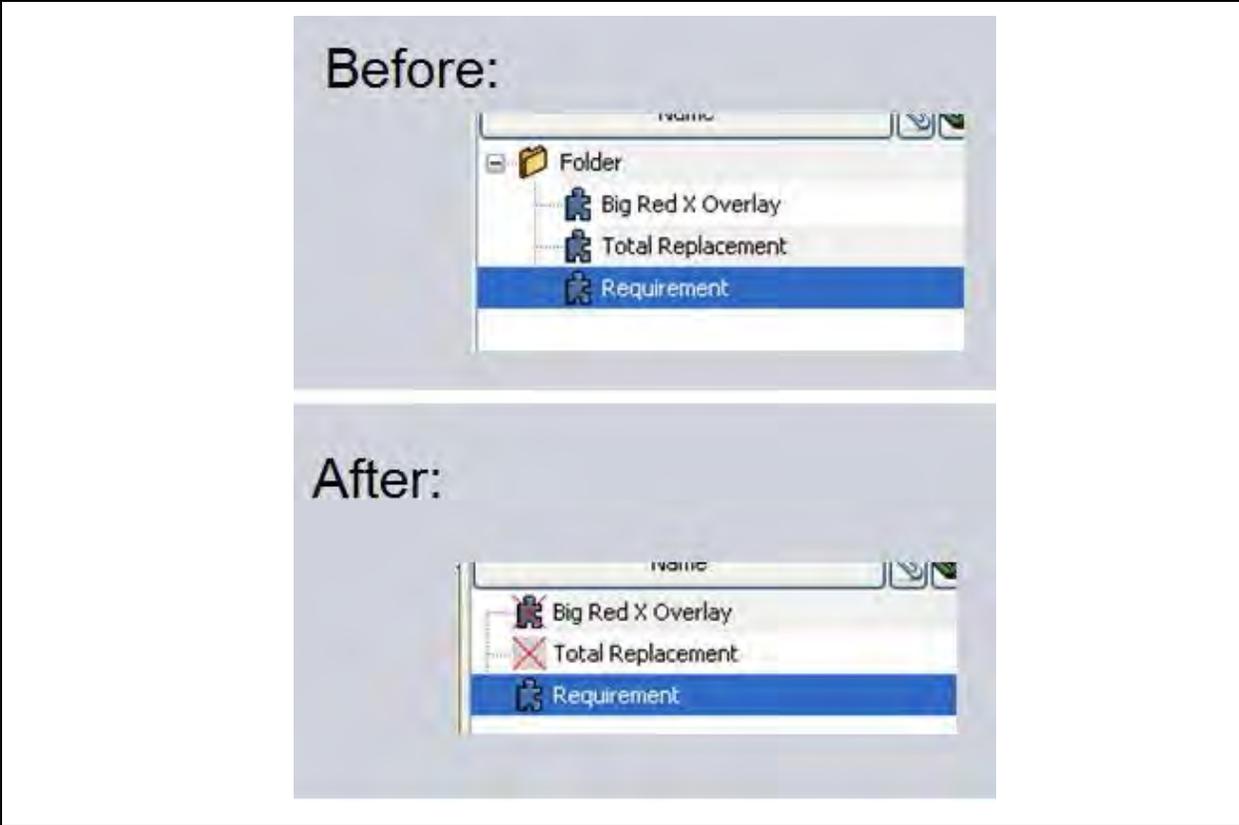


Figure 7-1. Icon Overlay sample

In Figure 7-1, the first requirement has a red X overlaid on the requirement. The second requirement is completely replaced with a red X.

-  Any overlay change is always a change on the actual object. The overlay is saved in the database and shared with all users in the Architect/Requirements project.

Chapter 8: Using Generic Links

This chapter describes generic links, and explains how to use them.

Introduction

The Systems Architect/Requirements Management links have built-in behavior that cannot be customized for new applications. New interfaces that require novel link behavior need a link type whose behavior can be completely controlled. Generic links provide the ability to control and customize the behavior of links between the Systems Architect/Requirements Management objects.

Generic link is a new database object type that can be subtyped to support application-specific linking needs. The default behavior and supported operations are similar to that of the trace links. For more information about the generic links, see the *Systems Architect/Requirements Management User's Manual*.

Support for Generic Links

Generic links are supported by attributes similar to those of the trace links. Table 8-1 lists the attributes supported for generic links and their equivalent trace link attributes. For more information about the trace links and their attributes, see the *Systems Architect/Requirements Management User's Manual*.

Table 8-1. Attributes Supported for Generic Links

Generic Links	Trace Links
Incoming Objects	Complying Objects
Incoming Link List	Complying Link List
Incoming Object Count	Complying Objects Count
Generic Link Count	Trace Link Count
Outgoing Objects	Defining Objects
Outgoing Link List	Defining Link List
Outgoing Objects Count	Defining Objects Count

Examples of API Methods

You can use the application-specific API methods to create, delete, modify, and navigate generic links. Generic links are supported in the [createLinks](#), [deleteLinks](#), [setValue](#), [getValue](#), and [getList](#) API methods.

- To create a generic link between two objects:

```
createLinks $start $end "Outgoing Generic Link"
```

You can also use "Incoming Generic Link", however, the direction of the link is reversed.

- To create a subtype of a generic link:

```
createLinks $start $end "Outgoing Generic Link" "mysubtype"
```

- To get the list of incoming objects:

```
getList[$obj "INCOMING_OBJECT_LIST"]
```

This command is similar to the `getList[$obj "COMPLYING_OBJECT_LIST"] trace link` command.

- To get the outgoing link list:

```
getList[$obj "OUTGOING_LINK_LIST"]
```

This command is similar to the `getList[$obj "DEFINING_LINK_LIST"] trace link` command.



When used with the [getList](#) API method:

- `LINK_TO_LIST` and `LINK_FROM_LIST` get the `INCOMING_OBJECT_LIST` and `OUTGOING_OBJECT_LIST` objects, respectively.
- `TO_LINKS` and `FROM_LINKS` get the `INCOMING_LINK_LIST` and `OUTGOING_LINK_LIST` objects, respectively.

Chapter 9: Cross-Product Messaging

This chapter describes the new `proxyAction` service which has been added to the set of SOAP services implemented for proxy objects in both Systems Architect/Requirements Management and Teamcenter Engineering. It uses the same URL, SOAP message format, authentication, session management and infrastructure as the existing set of services.

proxyAction

Each `proxyAction` request sends a single string argument to a foreign application. Based on the URL found in Application Registry for an Application ID, the string argument is sent to the foreign application. The response is a single string result, and a success/error indicator.

The applications must not place any restrictions on the content of the message or response strings, but they must be encoded properly to pass across the SOAP transport.

The `proxyAction` request message is of this form:

```
operationID – proxyAction
inParam1, “key” STRING – authentication key
inParam2, “appGuid” STRING – the sending application’s GUID
inParam3, “inputMsg” STRING – the action input string
```

The `proxyAction` response is of this form:

```
outParam1, “errMsg”, string – expected value Success
outParam2, “outputMsg”, string – the formatted output string
```

Systems Architect/Requirements Management and Teamcenter Engineering must each be able to send and receive `proxyAction` requests.

Setting Up proxyAction in Systems Architect/Requirements Management

Incoming proxyAction Requests

Incoming `proxyAction` requests are handled by first performing the usual authentication steps, and then connecting to an entry from the Systems Architect/Requirements Management session pool, as with any incoming Proxy SOAP request.

The incoming `proxyAction` string is sent to the Tcl interpreter to be evaluated. That Tcl may ultimately call any of the Tcl API functions, although it most often simply runs an existing Activator. The Tcl string result is returned to the originator of the `proxyAction` request, along with a success indication. If the Tcl interpreter exits with an unhandled error, the Tcl error string is returned as the string result, along with an error indication.

Sending proxyAction Requests

Sending `proxyAction` requests must be supported through a new Tcl API call:

```
proxyAction appID message
```

Where:

`appID` – A valid WOLF Application ID, likely found by reading the value from a Remote Proxy object within Systems Architect/Requirements Management.

`message` – The string to be sent to the remote application.

If the `appID` is not registered with WOLF, or if the remote application returns an error, a Tcl error should be thrown, with the Tcl error string set with the result from the remote application. If the call succeeds, the result string from this procedure should be the response string returned from the remote application.

Setting Up proxyAction in Teamcenter Engineering

Refer to Teamcenter Engineering documentation for details.

Remote Proxy Objects and Tcl API

Get Remote Proxy Properties

Tcl API callers must be able to get the Remote Proxy list of **ApplicationIDs** and **ObjectIDs** for any Systems Architect/Requirements Management object.

```
getValue object REMOTE_PROXY
```

The result is a list of lists of strings. The outer list has an entry for each Remote Proxy for the target object, or an empty list if it has none. Each inner list has two string elements, an **ApplicationID** and an **ObjectID**.

Appendix A: Word Content Activators

The text content of requirements and notes is processed using Microsoft Word's single file web page (**mhtml**) format. A fragment of the **mhtml** file, containing only the content of the requirement or note, is stored in the database with each object. When Word export is run, **mhtml** fragments from the exported objects are concatenated together, along with other data such as the style sheet, to form the **mhtml** file. An **mhtml** fragment can have references to objects outside the fragment. For example, cross references, which point to information in another Architect/Requirements object, and style names, which reference a style definition in the stylesheet.

As the export file is built, some of these references may require adjustments to function correctly. For example, the internal names used for graphic references. The graphic names include a sequential number to keep them unique. For example, **image001.gif**. The same graphic name can be used for different graphics in different Architect/Requirements objects. The graphic names are modified during export to ensure uniqueness.

In certain cases, it is not possible to resolve the external references. For example, a cross reference is not functional if the target for the reference is not included in the export. A custom style name does not work if that style is not defined in the style sheet used for the export.

Architect/Requirements does not correct external reference for numbered and bulleted lists. This is due to the unusual way in which numbered and bulleted list styles are defined in MS Word **mhtml** format. List types are arbitrarily assigned a list number and the mapping from list number to list style is not always clear. This causes the wrong list type to be used with numbered and bulleted lists in exports.

As a work around, activator events are defined to enable customized MS Word content correction. Activators can fire both during and after **mhtml** file generation. For specific use cases it is possible to correct list styles or MS Word content issues.

The new activator events are:

- **Word Edit Pre-Processor**
- **Word Edit Post-Processor**
- **Word Export Pre-Processor**
- **Word Export Post-Processor**

Unlike the other activators, these activators trigger based on the names. If an activator with the event name exists in a project, it triggers at the appropriate time during **mhtml** file generation for exports and edits in that project.

Word Edit Pre-Processor

This activator triggers when a requirement or note is opened for edit in MS Word. This allows modifying the **mhtml** file content that is displayed in MS Word.

The following information passed to the activator:

currentObject – Object ID of the requirement or note being opened for edit.

selected[0] – the **mhtml** file content.

The activator return value is used as the object's **mhtml** file content. If an error occurs in the activator or the return value is blank then the activator result is ignored.

Word Edit Post-Processor

This activator triggers after a requirement or note is saved in MS Word and before the content is saved in the database. This allows modifying the HTML content that gets saved in the database.

The following information passed to the activator:

currentObject – Object ID of the requirement or note being saved.

selected[0] – the HTML content being saved.

selected[1] – the full **mhtml** file that was edited.

The activators return value is stored as the object's HTML content. If an error occurs in the activator or the return value is blank then the activator result is ignored.

The **mhtml** file is provided so that you can examine the stylesheet information.

Word Export Pre-Processor

This activator triggers for each database object before it is added to the exported MS Word document. This allow modifying object content in the document.

The following information passed to the activator:

currentObject – ID of the object being exported.

selected[0] – HTML content of the object.

selected[1] – ID of the style sheet used for the export.

The returned value is added to the export document instead of the original content. If an error occurs in the activator or the return value is blank then the activator result is ignored.

The HTML content passed has already been processed using the object template and includes the heading.

This activator can trigger for all objects types and not for requirements and notes only.

Word Export Post-Processor

This activator triggers once for an exported MS Word document. This allows modifying document file content before it is downloaded to the client and opened.

The following information passed to the activator:

currentObject – ID of the object being exported. If multiple objects are exported then the first object is used.

selected[0] – Path name of the exported MS Word document.

The exported file name is provided so that you can examine and modify the **mhtml** file.

The activator return value is not used. Errors in the activator are logged and ignored.

Appendix B: Examples for using C# API's

This chapter contains examples of using the Architect/Requirements API's. You can use the API using VBA and C#.

Accessing Using VBA

An example of using API's to access Architect/Requirements using VBA is provided with the kit. The example uses API's to perform the following functions:

- Connection to Architect/Requirements
- Get a list of projects
- Create folder in the selected project
- Create building block in a folder
- Modify an object
- Delete an object from Architect/Requirements
- Searching for requirements

To view the example:

1. Extract the <DVD Root>\ **ClientAPIExample\ClientAPIExample.zip** file from the product DVD to a temporary folder on the machine where you are running the client.
2. Double click the **ClientAPIExample.xlsm** to open it in Microsoft Excel.
3. If you get a security warning, click **Enable Content** to enable the macros in the project.
4. Click the **Launch Example** button.
5. Enter the port number on which the client can be accessed and click **OK**.
You can also accept the default port **4002**.
6. Enter the name of the machine on which the Architect/Requirements server is running and click **OK**.
7. Enter the Architect/Requirements user name to access the application and click **OK**.
8. Enter the password for the account and click **OK**.

If you have entered the correct logon credentials, a connection successful message is displayed.

9. Click **OK**.
10. The **Select Project** dialog box displays a list of projects in Architect/Requirements. From the drop down list, select the project in which you want to create the objects and click **OK**.
11. Various Architect/Requirements objects are created and corresponding success or failure messages are displayed.
12. Architect/Requirements is searched and the requirements are displayed.
13. The example launches automatically and connects the Architect/Requirements application, creates a project, creates a requirement.

To view the VBA code, enable the **Developer** ribbon.

1. Click **File**→**Options**→**Customize Ribbon**, select the **Developer** check box, and click **OK**.
2. Click the **Developer** ribbon and click Visual Basic.
3. Double-click **ThisWorkbook** under the Microsoft Excel Objects in the left pane.
4. To close the example, exit Excel.

Following functions are defined in the example to perform various operations:



You must provide the parameters corresponding to your setup for the functions to work correctly.

- **connectToTcR()**

Function to connect to Architect/Requirements. This function runs once when the example is launched. The following parameters are collected from the user to connect to Architect/Requirements:

- Port on which the client/server is running

The **connectionService.tcr_client_socket_port = socketNum** paramter is used to set the port number.

- Name of the server machine

The **connectionService.tcr_client_socket_url = machineNum** paramter is used to set the name of the Architect/Requirements server.

- Valid Architect/Requirements username and password

The **connectionService.user_name = uName** paramter is used to set the username of the Architect/Requirements user.

The **connectionService.user_password = pass** paramter is used to set the password for the user.

The function attempts to connect to the Architect/Requirements client. If the client is not running, it connects to the server.

- **getProjects()**

Function to get a list of projects from Architect/Requirements.

- **getSelectedProjectId(projectName As String)**

Function to get the ID of the selected project. This function accepts the project name and gets the project ID.

- **createObjectInTcR(owner As String, typeName As String)**

Function to create any Architect/Requirements object. This function accepts the ID of the object that you want to create and the type of the object.

- **modifyObjectInTcR(modiftObject() As TcR.dataBean)**

Function to modify objects in Architect/Requirements. This function accepts the object to be modified. It modifies the name and text of the requirement, however, this function can be used to modify any object created in Architect/Requirements.

- **deleteObjectInTcR(objectId() As String)**

Function to delete objects in Architect/Requirements. This function accepts the ID of the object that you want to delete.

Accessing Using C#

An example of using API's to access Architect/Requirements using C# is provided with the kit. The example uses API's to perform the following functions:

- Connection to Architect/Requirements
- Get a list of projects
- Create folder in the selected project
- Create building block in a folder
- Modify an object
- Delete an object from Architect/Requirements
- Searching for requirements

You must have Microsoft Visual Studio 2010 installed to view the example.

To view the example:

1. Extract the *DVD Root\ClientAPIExample\ClientAPIExample.zip* file from the product DVD to a temporary folder on the machine where you are running the client.
2. Navigate to **Example in c#** folder.
3. Double click **API Reference Solution.sln**.
4. In the **Solution Explorer**, click the **MainClass.cs**.
5. Modify the following paramters to match your installation:
 - **connectionService.tcr_client_socket_port**
 - **connectionService.tcr_client_socket_url**
 - **connectionService.tcr_server_controller_url**
 - **connectionService.user_name**
 - **connectionService.user_password**
6. Click the **Start** button on the toolbar to view the functions of the example.
7. To close the example, exit Visual Studio.

Index

Access Privileges for an Activator.....	141
Activator	
Login.....	148
Logout.....	148
Pick list.....	154
Activators.....	141
Defining Events.....	145
Deleting.....	130
Release reservation.....	143
Transaction Management.....	79
Activators	
Creating.....	143
Activators	
Deleting Objects.....	163
Activators	
Deleting Objects.....	163
Activators	
Discarding Objects.....	163
Activators	
Examples.....	167
Activators	
Examples	
Same Name.....	167
Activators	
Examples	
Creating a Note.....	167
Activators	
Examples	
After Delete Activator Event.....	169
Activators	
Examples	
createAction FileDownload.....	170
Activators	
Examples	
createAction RunJava.....	171
Activators	
Examples	
Running from command line, tcradmin	
script.....	179
Activators Objects.....	142
Actual Name	
Schema Object.....	22
Adding reference to TcR.tlb.....	64
API	
Change List.....	25
Database Transactions.....	25
DataBeans.....	29
Describing API Functions.....	24
Error Handling.....	24
Introduction.....	15
Logging in.....	23
Message list.....	24
ResultBeans.....	24
Schema List.....	25
Using.....	23
C# API.....	63
calculateProperties.....	80
Change flags.....	152
Change List.....	25
changeApproval.....	81
Code conventions.....	12
Command line	
Running an activator.....	179
Command line entry conventions.....	12
Configuring COM Client.....	64, 65
VBA.....	64
Connecting to TcSE.....	65
Conventions	
Code.....	12
Command line entries.....	12
File contents.....	12
Names.....	11
Values.....	11
copyObjects.....	82
createAction.....	83
createAction FileDownload.....	86
createAction RunJava.....	88
createBaseline.....	91
createExternalLink.....	92
createLinks.....	93
createLinksCOM.....	71
createObject.....	94

createObjectCOM	68	getResultCOM	72
createProject.....	95	getValue.....	117
createUser	97	Icon Overlay	187
createVariant.....	98	importDocument.....	119
createVersion	99	Introduction	141
Database Transactions	25	Java API examples.....	30
DataBeans	29	Java API methods	29
deleteLinks.....	100	Logging in to the API	23
deleteObjects.....	101	Login activator.....	148
Deleting activators	130	Logout activator.....	148
Display Name		LOID	
Schema Object	22	Passing	
displayMessage.....	101	setObject	132
emptyTrashcan.....	103	LOID of a Pick list	
Error handling	24	obtaining	
Events		getObject.....	111
After import	150	getValue.....	117
Before import.....	150	Macro	
Change Approval Routing	149	Creating	180
Change List.....	151	Examples	181
Import.....	150	Form Values	180
Object Modify Events	145	Running	181
Session Modify Events	148	Tcl environment.....	180
Storing Event Context.....	150	Using.....	180
export2Excel	104	Message list	24
exportDocument.....	106	moveObjects	121
exportXML	108	Name conventions	11
File contents conventions.....	12	Overlay	
Flags		Icon	187
Create flags	153	Pick list activator	154
Delete flags	153	Prerequisites.....	64
Modify flags.....	153	proxyAction.....	191
Relation flags	152	Incoming requests.....	191
Form Value	180	Sending requests	192
Freezing object, Static property, setValue	136	Reference Links	
Generic links	189	Properties	185
Attributes	189	restoreFromTrashcan	122
Examples of API methods	190	ResultBeans	24
Introduction.....	189	runActivator.....	123
Support.....	189	Running activator from command line, tcradmin	
getDataBeansCOM	74	script	179
getEnvironment.....	109	runReport.....	124
getList	110	Schema List	25
getObject.....	111	search.....	125, 126
getObjectCOM.....	69	sendEmail	127
getProjects.....	112	setEnvironment	128
getPropertiesCOM	73	setObject	96, 132
getPropertiesWithFormula.....	113	setObjectsCOM	70
getPropertyDefinition	114	setPassword	133
getPropertyDefinitions.....	115	setUserPreferences.....	134
getRemoteObjectTraceReport.....	116	setValue	135

Shortcut Objects	
Children	183
Keywords	183
Properties	183
Standard input parameters.....	17
Static property, setValue.....	136
Tcl	
Defined.....	78
Resources	78
Tcl Scripts	
Errors	79
Executing	79
Methods	79
tcradmin script	
Activator, running from command line	179
Transaction Control	158
Unfreezing object, Static property, setValue ..	136
Using API	197
Value conventions	11
VBA Examples	67
Working with Reference Links.....	185
Working with Shortcut Objects	183
writeLog.....	139, 140

Teamcenter 11.1 Systems Engineering and Requirements Management

Systems Architect/ Requirements Management Server Installation Manual for Windows

Manual History

Manual Revision	Teamcenter Requirements Version	Publication Date
A	3.0.1	May 2003
B	4.0	December 2003
C	4.1	February 2004
D	5.0	July 2004
E	6.0	March 2005

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
F	2005	September 2005
G	2005 SR1	June 2006
H	2007	December 2006
I	2007.1	April 2007
J	2007.2	September 2007
K	2007.3	January 2008
L	8	January 2009
M	8.0.1	June 2009
N	8.1	October 2009
O	8.2	October 2010
P	9	July 2011
Q	9.1	May 2012
Q1	9.1.5	April 2014
R	10.0	January 2015

Manual Revision	Teamcenter Systems Engineering and Requirements Management Version	Publication Date
S	10.1	September 2016
T	11.1	March 2018

This edition obsoletes all previous editions.

Server Installation Manual Contents

Manual History	4
Contents	7
Preface.....	11
Audience	11
Conventions	12
Revision Marks	12
Browser and Dialog Window Examples.....	12
Names and Values	12
Command Line Entries, File Contents, and Code.....	13
Submitting Comments.....	14
Proprietary and Restricted Rights Notice.....	14
Chapter 1: Preparing for the Installation.....	15
Components	15
Hardware Requirements.....	17
Software Requirements	18
Requirement for Versant 64-bit Installation on Windows	18
Oracle JDK 64-bit.....	18
Chapter 2: Installing Architect/Requirements.....	21
Installation Overview	21
Installing Architect/Requirements and Versant on a Single Server.....	22
Installing Architect/Requirements and Versant on Multiple Servers	23
Installing Architect/Requirements Client in Silent Mode.....	24
User Privileges for the Installations	24
Installing the Versant Object Database Application	25
Installing Versant 9	25
Post-Versant Installation Steps	35
Verifying the Versant Installation.....	35
Installing the Architect/Requirements Server	37
Installing the Server	38
Verifying Versant Environment Variables.....	43
Copying the Versant License File.....	43

Initializing the Versant Database	43
Testing the Database	44
Next Steps	45
Chapter 3: Upgrading the Installation.....	47
Upgrade Overview	47
User Privileges for Running the Installations	47
Upgrade Planning	48
Upgrading From Architect/Requirements 10.1 or Later Versions.....	48
Upgrade Prerequisite Tasks	49
Backup Customized JSP and Schema Files	49
Exporting the Customized Schema Objects.....	49
Cleaning the Database and Taking a Backup.....	50
Upgrading the Operating System.....	51
Uninstalling Architect/Requirements Web Applications.....	51
Uninstalling Architect/Requirements 10.1 or later patch.....	51
Undeploying the Existing WAR File	52
Uninstalling the the Versant Object Database 8	53
Uninstalling Versant Object Database 8	53
Installing the Versant Object Database 9.....	54
Installing Versant 9.....	55
Verifying the Versant Installation.....	63
Post-Versant Installation Steps	64
Installing Architect/Requirements Server.....	65
Installing the Server	65
Copying the Versant License File.....	71
Upgrading the Database for use with Architect/Requirements 11.1	71
Upgrading the Architect/Requirements Schema.....	71
Completing the Upgrade	71
Running the maintainDB Utility	72
Next Steps	72
Chapter 4: Post-installation Tasks	75
Overview.....	75
Restoring Custom JSP Files.....	77
Deploying the tcr.war File	77
Deploying tcr.war File on Oracle WebLogic.....	77
Deploying tcr.war File on IBM WebSphere	80
Deploying the tcr.war File on Apache Tomcat.....	82
Updating the Web Application Configuration Parameters	83
Entering Architect/Requirements License Information	83
Entering Your Customer Number.....	84
Entering Your License Key	84
Upgrading the Architect/Requirements Client.....	85
Verifying the Installation	86
Verifying the Installation on the Server.....	86
Verifying the Installation on the Client	87

Restoring Customized Schema	89
Importing Schema for Folder Move Confirmation	89
Chapter 5: Uninstalling Architect/Requirements	91
Uninstalling the Server Software	91
Appendix A: Directories and Files Created During Server Installation.....	93
Appendix B: Upgrading the Architect/Requirements Schema while moving the database server ..	94
Ensuring unique database numbering	95
System 1 running on Solaris Server	96
System 1 running on Windows Server.....	98
Index.....	100

Preface

This manual is a server installation reference for Systems Architect/Requirements Management (Architect/Requirements). Architect/Requirements belongs to the Siemens PLM Software portfolio of digital product lifecycle management software and services.

Audience

This manual is for application server administrators who are responsible for installing the Architect/Requirements Web component and database server component. The manual provides an overview of installation requirements and instructions for installing the Web and database server components. For information about installing the Architect/Requirements client component, see the *Systems Architect/Requirements Management User's Manual*.

Conventions

This manual uses the conventions described in the following sections:

Revision Marks

Technical changes are marked by a bar adjacent to the changed text.

Browser and Dialog Window Examples

The examples of browsers and dialog windows in this manual may appear different from those you see on your screen:

- The examples reflect Systems Architect/Requirements Management as initially installed at your site. Your enterprise may customize the browsers and dialog windows such that they appear different from those in the examples.
- The examples reflect individual Systems Architect/Requirements Management modules. If you install additional modules, your dialog windows and browsers reflect the additional modules.
- The examples reflect Systems Architect/Requirements Management installed on a Windows platform.

Names and Values

This manual represents system names, file names, and values in fonts that help you interpret the name or value. For example:

Change or add the parameter to the **initsid.ora** file.

The conventions are:

Bold	Bold font represents unvarying text or numbers within a name or value. Capitalization is as it appears.
<i>Italic</i>	Italic font represents text or numbers that vary within a name or value. The characters in italic text describe the entry. Letters are shown in lowercase, but the varying text may include uppercase letters. In initsid.ora , <i>sid</i> identifies a varying portion of the name (a unique system ID). For example, the name of the file might be: initBlue5.ora
<i>text-text</i>	A hyphen separates two words that describe a single entry.

Command Line Entries, File Contents, and Code

This manual represents command line input and output, the contents of system files, and computer code in fonts that help you understand how to enter text or to interpret displayed text. For example, the following line represents a command entry:

```
msqlora -u system/system-password
```

The conventions are:

Monospace	<p>Monospace font represents text or numbers you enter on a command line, the computer's response, the contents of system files, and computer code.</p> <p>Capitalization and spacing are shown exactly as you must enter the characters or as the computer displays the characters.</p>
<i>Italic</i>	<p>Italic font represents text or numbers that vary. The words in italic text describe the entry.</p> <p>The words are shown in lowercase letters, but the varying text may include uppercase letters. When entering text, use the case required by the system.</p> <p>For the preceding example, you might substitute the following for <i>system-password</i>:</p> <pre>KLH3b</pre>
<i>text-text</i>	<p>A hyphen separates two words that describe a single entry.</p>

Submitting Comments

Portions of Teamcenter software are provided by third-party vendors. Special agreements with these vendors require Siemens PLM Software to handle all problem reports concerning the software they provide. Please submit all comments directly to Siemens PLM Software.

Please feel free to share with us your opinion on the usability of this manual, to suggest specific improvements, and to report errors. Mail your comments to:

Siemens PLM Software Technical Communications
5939 Rice Creek Parkway
Shoreview, MN 55126
U.S.A.

To submit an incident report, you can use the Siemens PLM Software GTAC online support tools at the following URL:

http://www.plm.automation.siemens.com/en_us/support/gtac/

Proprietary and Restricted Rights Notice

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

© 2018 Siemens Product Lifecycle Management Software Inc. All Rights Reserved.

All trademarks belong to their respective holders.

Chapter 1: Preparing for the Installation

This chapter describes the Architect/Requirements components and the requirements for the installation of the Systems Architect/Requirements Management server.

Components

Architect/Requirements is designed for multi-tier Web deployment. It consists of the following main components:

- *Web* component

The Architect/Requirements business logic resides in the Web component. This component runs within the Java environment of an application server, or Web server with a servlet engine. The Web component consists of one binary file, **tcr.war**.

- *Database server* component

Architect/Requirements uses an object-oriented database system from Versant Corporation to store and manage data. The database server component consists of several binary files and must be installed separately before installing the Systems Architect/Requirements Management server.

Architect/Requirements is certified and ships with Versant Database.

For information about the version of Versant Database, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- *Client* component

This component presents the Architect/Requirements user interface on a local computer.

The Web component and database server component can be installed on the same computer or on two separate computers.

When the Web and database servers are on different computers, they must be within the same local area network (LAN) and never connected over a wide area network. Even within a LAN, performance can be adversely affected by latency introduced by routers and firewalls, or by network traffic from other sources. For best performance, place the database server and Web server on a dedicated, private LAN, that is isolated from other traffic and has near-zero latency.

This arrangement is not a minimum requirement. However, it is recommended when there are enough users that multiple Web server instances become necessary for scalability. That level of user activity generates a high volume of small communications to the database server.

Hardware Requirements

The Architect/Requirements can be deployed in multiple configurations. Following are the recommendations for the application server and the database server.

Application Server hardware requirements (per Architect/Requirements instance)

For an anticipated 15 concurrent users

- RAM
2 GB
- Processor
2 CPUs (or cores) per instance

For an anticipated 50 concurrent users

- RAM
8 GB for 64-bit applications
- Processor
4 CPUs (or cores) per instance

Versant Database Server hardware requirements

- RAM
Memory requirements are driven by the size of the database and the desire to cache all or a significant portion of the database.
The minimum requirement for Windows is 4 GB of RAM.
- Processor
2 CPUs (or cores) are required but 4 CPUs (or cores) are recommended for better performance
- Disk space
The disk space requirement depends on the size of the database. If you are using a default database of 4 GB, you must have the following:
 - 10 GB for database area
 - 20 GB for backup area
 - 10 GB for working files
 - 50 GB for operating system swap and other applications

Temporary installation space (not to exceed the requirements above) is also required in your **Temp** directory during the installation.

RAM and disk space requirements must be adjusted upward depending on the resources required by other applications on the same servers.

These requirements do not include the disk space needed to create the database or to store back ups. For more information, see the *Systems Architect/Requirements Management System Administrator's Manual*.

Software Requirements

Architect/Requirements 11.1 is a 64-bit application. Both the application server and the database server must be a 64-bit deployment.

For a 64-bit Architect/Requirements installation, 64-bit hardware, a 64-bit operating system, and a Web server such as WebLogic, WebSphere, or Tomcat are required.

The Architect/Requirements components can be installed on the following operating systems:

- Microsoft Windows
- Sun Solaris

For information about versions of operating systems, third-party software, and Teamcenter software that are certified for your platform, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

Requirement for Versant 64-bit Installation on Windows

The installation of Versant 64-bit on a Microsoft Windows system requires Microsoft Visual C++ 2015 Redistributable Package (x64). You can download the Microsoft Visual C++ 2015 Redistributable Package (x64) from the following link:

<https://www.microsoft.com/en-us/download/details.aspx?id=53840>

Oracle JDK 64-bit

The installation procedure requires deployment of the Architect/Requirements **war** file on a Windows computer. IBM WebSphere provides the required Java runtime support, with the IBM JVM version 1.7 or 1.8. But if you are not planning to deploy Architect/Requirements on WebSphere, you must install either the Java Development Kit (JDK) or the Server JRE (Java SE Runtime Environment) on your Windows computer. You can obtain the JDK or the Server JRE from your IT department or download it directly from the Oracle Web site:

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>



You must use the 64-bit version only.

For information about version of Oracle JDK certified for your platform, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

Setting the Java environment variables

You must set the following Java variables:

- **JAVA_HOME**: Set it to the Java 8 JDK directory. For example,
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_151
- **PATH**: Add the Java bin directory to the PATH variable. For example,

PATH=%JAVA_HOME%\bin;%PATH%

Chapter 2: Installing Architect/Requirements

This chapter contains an overview and instructions for installing Architect/Requirements and its components. It also describes how to set environment variables and configure the Versant database. If you are upgrading an existing installation of Architect/Requirements, skip to [Upgrading the Installation](#).

Installation Overview

Architect/Requirements and its components can be installed in different configurations. Single-server and multiple-server installations are described in brief in the following sections. You may refer to this overview and follow the steps explained in detail in the subsequent sections.

You should verify the hardware and software requirements for Architect/Requirements prior to the installation.

Installing Architect/Requirements and Versant on a Single Server

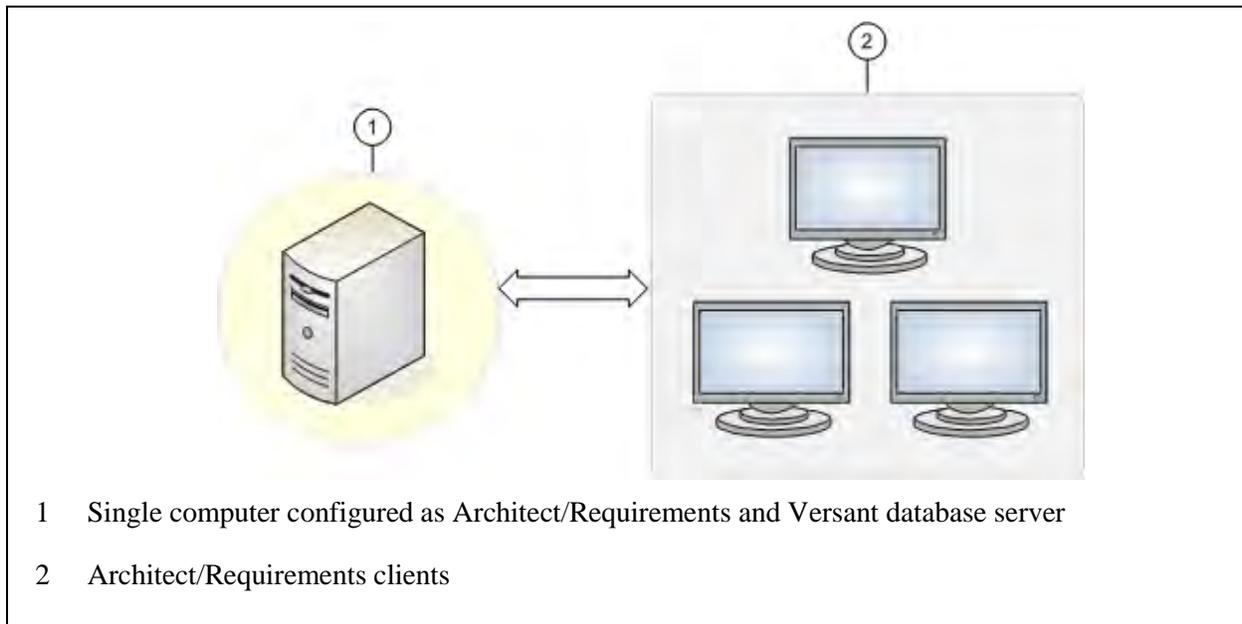


Figure 2-1. Architect/Requirements and Versant Installed on a Single Server

In a single server installation, the Systems Architect/Requirements Management server and Versant object database are installed on the same computer. At a high level, the steps involved are:

1. Install a Web application server supported by Architect/Requirements.
2. Install the Versant object database.
3. Install the Architect/Requirements server software.
4. Set environment variables.
5. Copy the Versant license file and initialize the database.
6. Perform post-installation procedures, such as deploying the Web component (**tcr.war** file), and entering the licensing information.

Installing Architect/Requirements and Versant on Multiple Servers

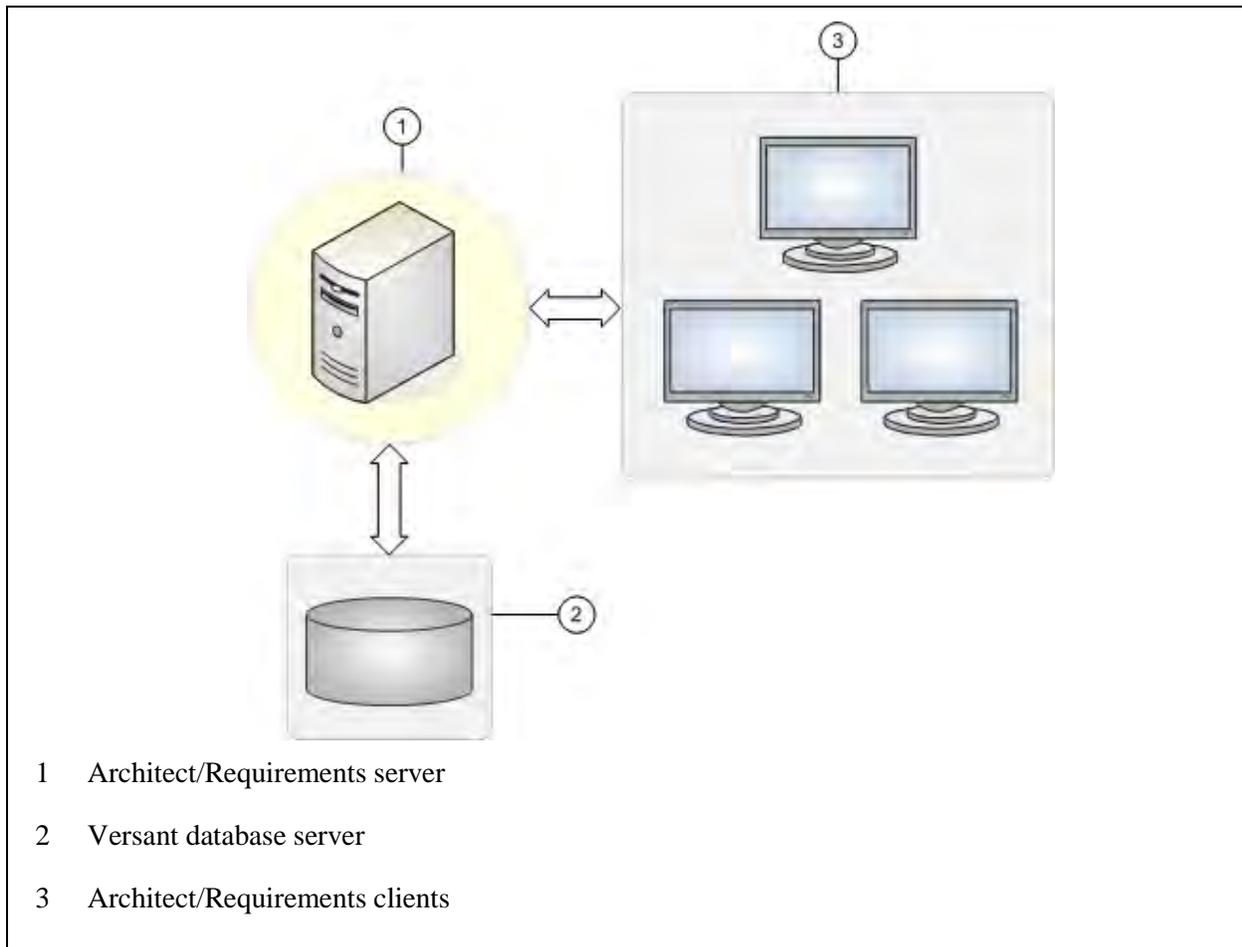


Figure 2-2. Architect/Requirements and Versant Installed On Multiple Servers

In a multiple server installation, the Systems Architect/Requirements Management server and the Versant object database are installed on separate computers.

If you are installing and configuring an additional Architect/Requirements server or a Versant database server, you should install both Versant and Architect/Requirements server software on every server.

- On the computer to be configured as the Versant database server:
 - . Install the Versant object database.
 - . Copy the Versant license file.
 - . Initialize the Versant database.
- On the computer to be configured as the Architect/Requirements Server:
 - . Install a Web application server supported by Architect/Requirements.
 - . Install the Architect/Requirements server software.

In the Database Input dialog window, ensure that you enter the host name of the Versant database server (installed and configured on a separate computer) and the name of the database.

- . Set environment variables.
- . Perform post-installation procedures, such as deploying the Web component (**tcr.war** file), and entering licensing information.

Installing Architect/Requirements Client in Silent Mode

The Architect/Requirements client installation program provides an option to install the client in silent mode. The IT departments or enterprise administrators can use the silent mode option to roll out command line installations of Architect/Requirements Client with Office Integration on end-user machines.



- Elevated user privileges (power user or administrator) are required for the installation as registry entries are updated with the patch information.
- The client installer does not check for the user privilege level. If the installation is not performed as a privileged user, the registry entries are not created and the client fails to run when launched by a normal user.

To install the Architect/Requirements client in silent mode:

1. The client installation program is located in the Architect/Requirements Web component (the **tcr.war** file).

The **tcr.war** file is located in the **war_file** directory. The typical location of the **war_file** folder is *C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file*.

Extract the **tcr.war** file in a folder. You can use Winzip, or run the command: `jar -xf tcr.war`

2. Locate the Architect/Requirements 11.1 client installer (**setup.exe**) in the **/ugs/tc/req/installs** folder and run the following commands:

```
setup.exe -i silent
```

The client is installed in the default location, such as, **C:\Program Files\SiemensPLM\Teamcenter\SystemEngineering\Release_11.1** folder.

User Privileges for the Installations

Privileges for Installing the Server

Users installing and configuring the Architect/Requirements server and the Versant database server must be logged on to the systems with administrator privileges.

The Architect/Requirements server installer should be run with a user ID that is used to manage both Architect/Requirements and Versant. This user ID may be different from the user ID used to install the Versant database.

Privileges for Installing the Client

Elevated user privileges (power user or administrator) are required for installation and update of the Architect/Requirements client. This is required as Windows registry entries are updated during the patch update.

Installing the Versant Object Database Application

Architect/Requirements uses an object-oriented database system from Versant Corporation to store and manage data. Architect/Requirements is certified and ships with Versant Database.

For information about the version of Versant Database supported, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

The Versant installer is no longer a part of the Architect/Requirements server installer. The Versant database server must be installed before installing the Architect/Requirements server. The instructions are the same for installing the Versant database server. The Architect/Requirements server may be installed either on the same machine, or on a separate machine. If you are running these servers on separate machines, perform this installation only on your database server machine.

Architect/Requirements 11.1 supports the Versant Object Database 9.3 patch 12. The Versant installer is included in the distribution archive in the *Root-Folder***Versant_9****Windows** folder.

The Versant database files must be on a file system that is local to the Versant server processor. Database corruption can occur if you use a remotely mounted drive.

Installing Versant 9

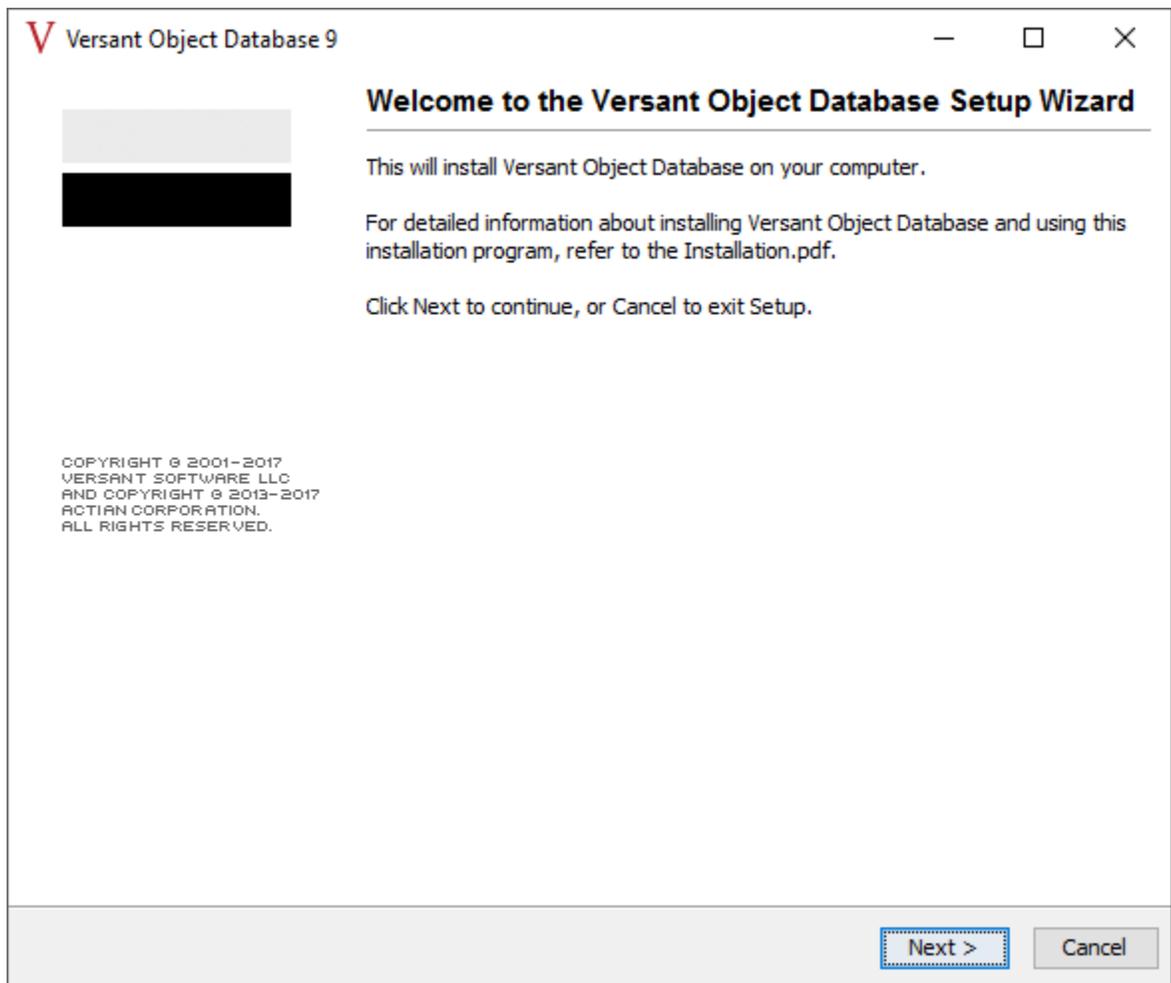
1. Run the Versant installer.

Extract **VOD9.3.0.12_2561_Windows_VS2015_64bit_opt.zip** to a temporary folder.

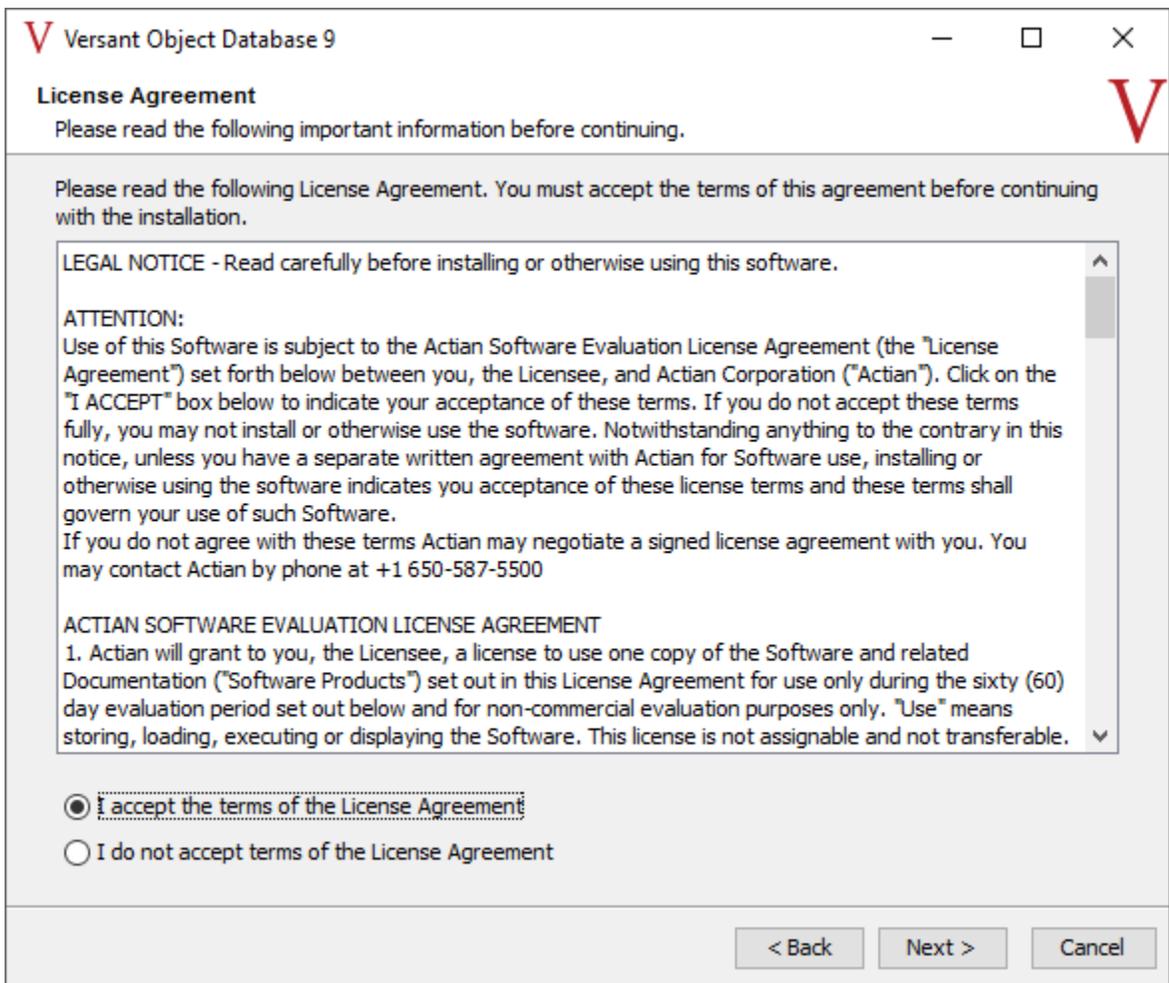
As the Administrator, execute **VOD9.3.0.12_2561_Windows_VS2015_64bit_opt.exe**.

It can take several minutes before the installation window appears.

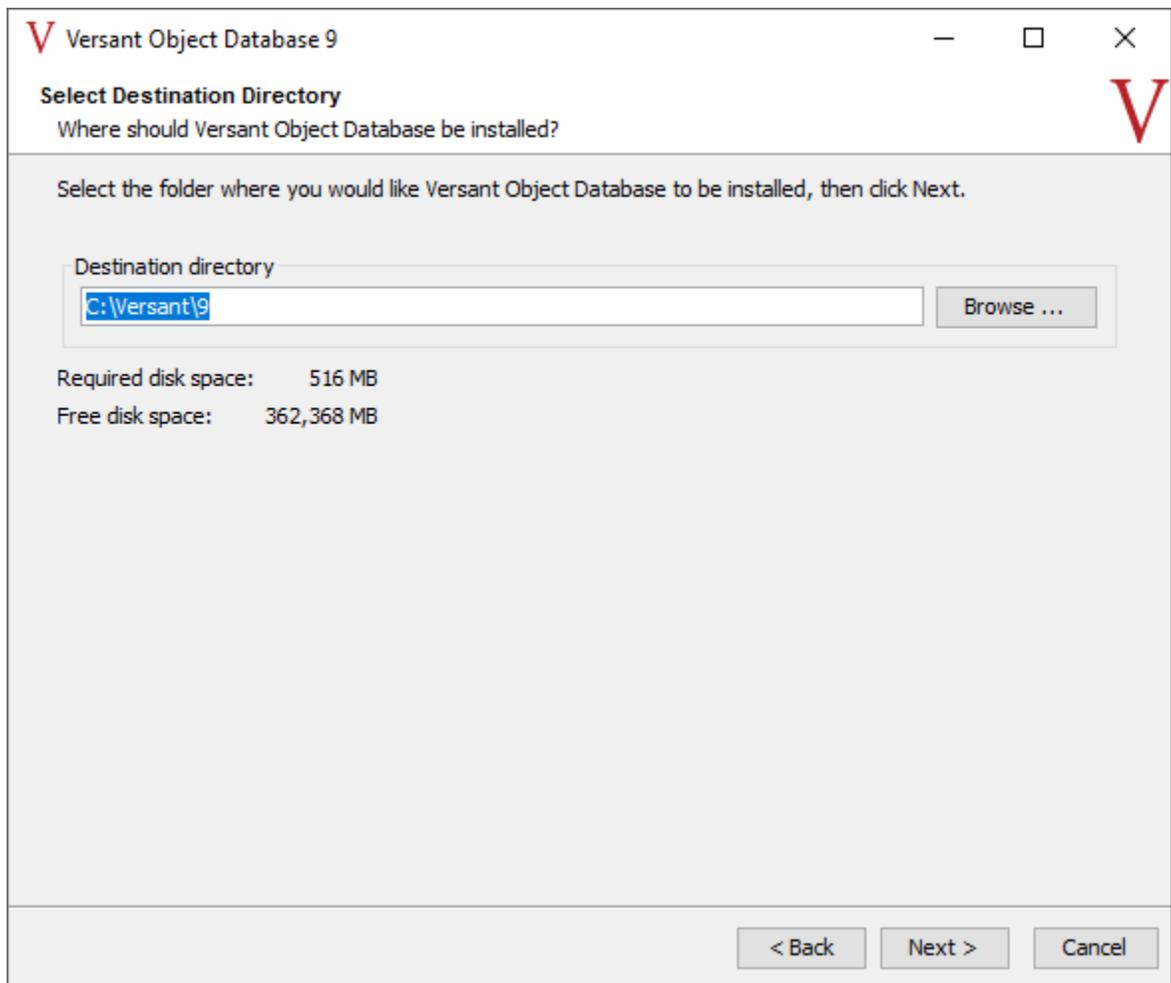
2. On the **Welcome to the Versant Object Database Setup Wizard** screen, click **Next**.



3. On the **License Agreement** screen, select **I accept the terms of the License Agreement** and click **Next**.



4. On the **Select Destination Directory** screen, specify the location to install the Versant binaries and click **Next**.



5. On the **Select Components** screen, select or clear the check boxes as relevant:

Clear the **SDK** check box.

You may need to click twice in the SDK check box to clear it. Clearing the **SDK** check box clears the **C SDK**, **C++ SDK**, **Java**, and **.NET SDK** check boxes.

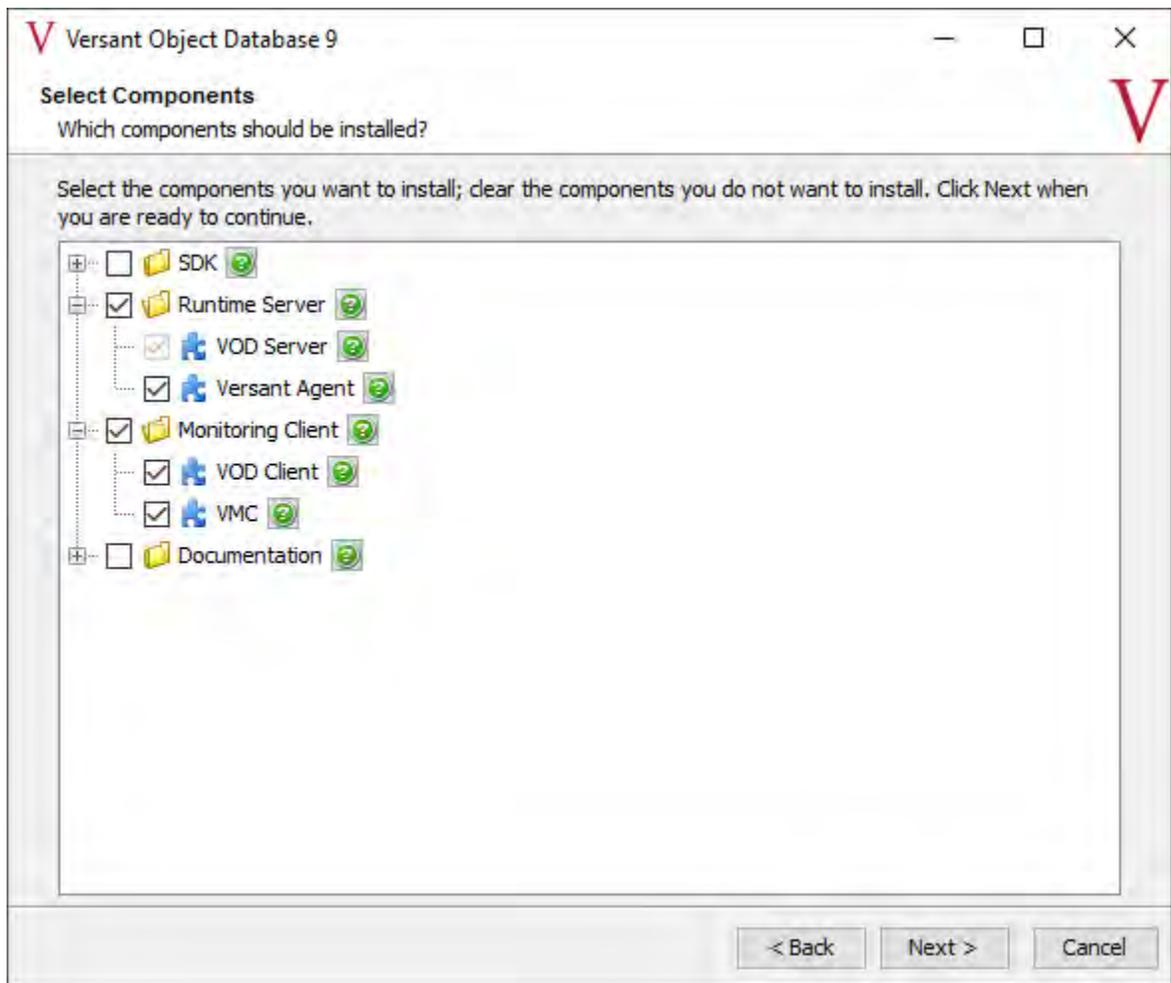
Expand the **Runtime Server** section and clear the **VSQL Server** check box.

Verify that only **VOD Server** and **Versant Agent** are selected.

Expand the **Monitoring Client** section and clear the **VSQL Client** check box.

Verify that only **VOD Client** and **VMC** are selected.

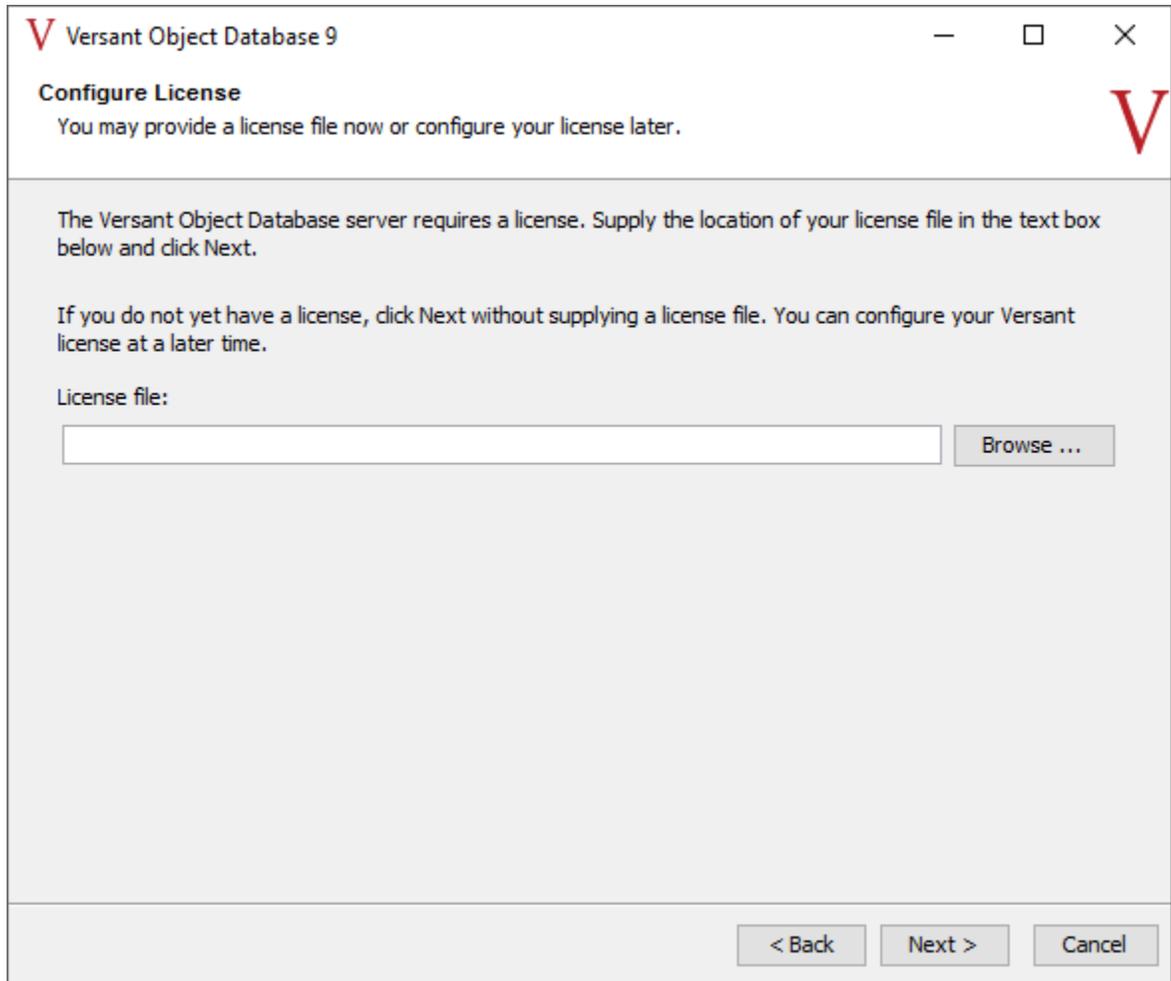
If you do not want the Versant documentation to be installed, clear the **Documentation** check box and click **Next**.



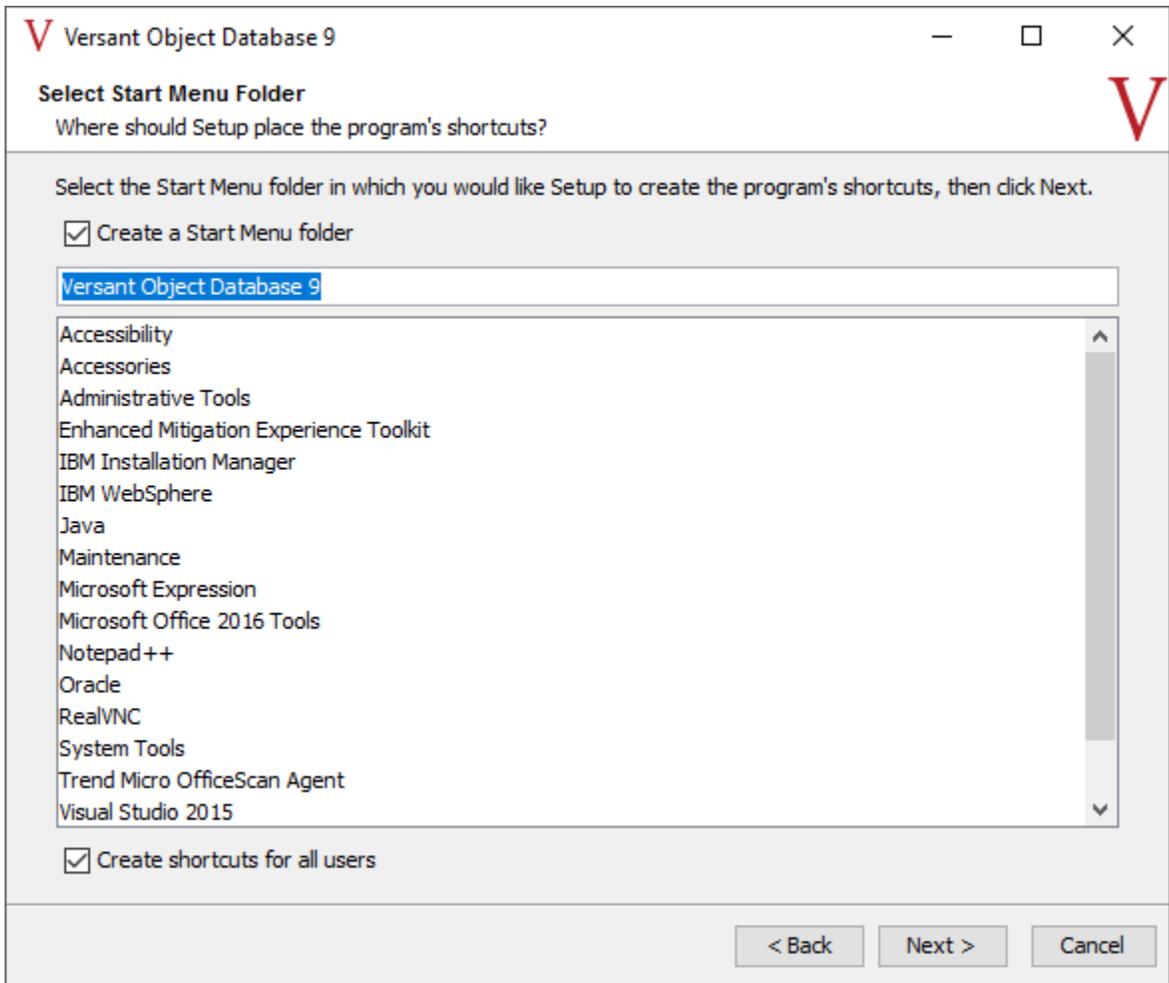
6. On the **Configure License** screen, click **Next**.



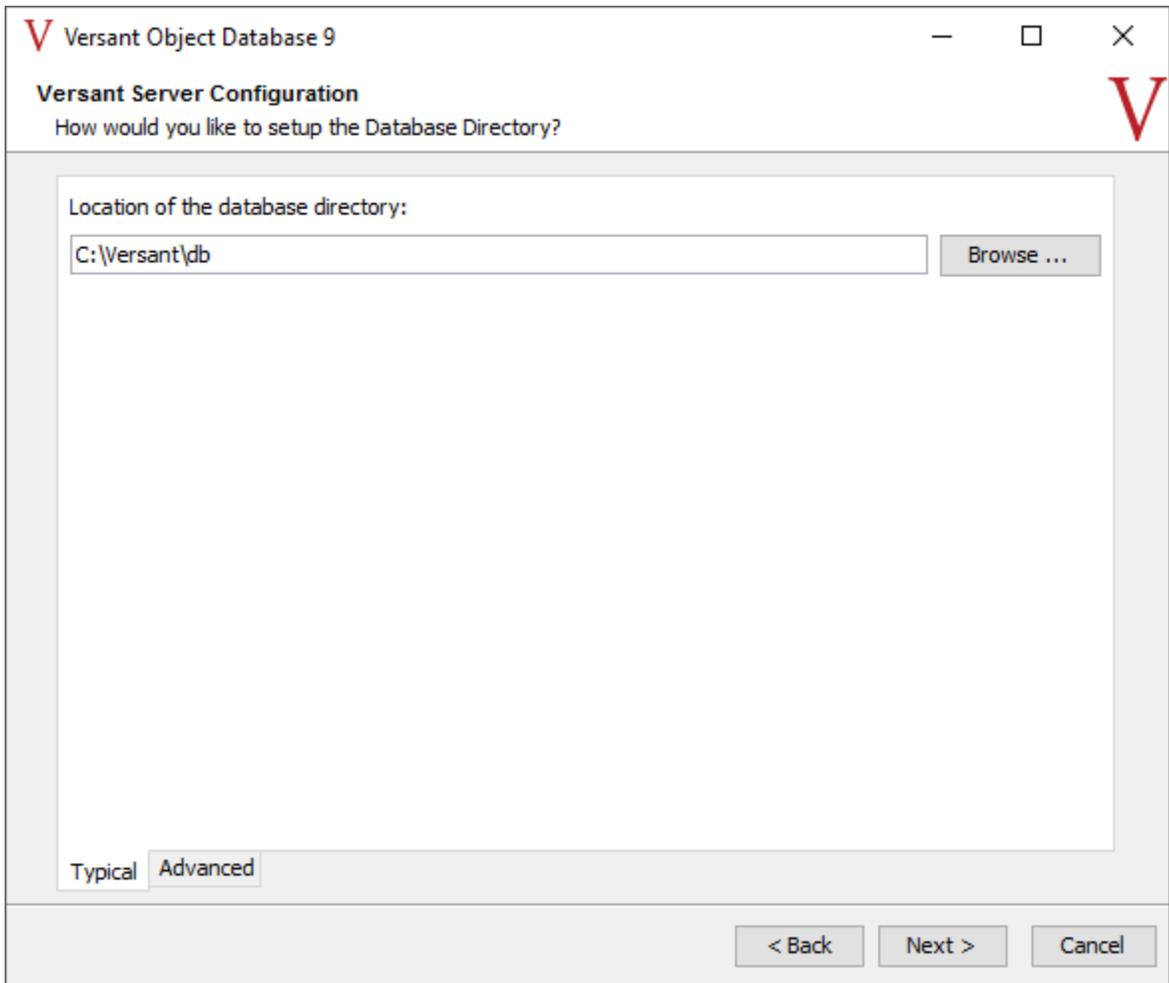
The required Versant license is available after installing Architect/Requirements. You must configure the license after completing the Architect/Requirements installation.



7. On the **Select Start Menu Folder** screen, clear the **Create shortcuts for all users** check box if you do not want the Versant link to be created in the **Start** menu for all the users and click **Next**.

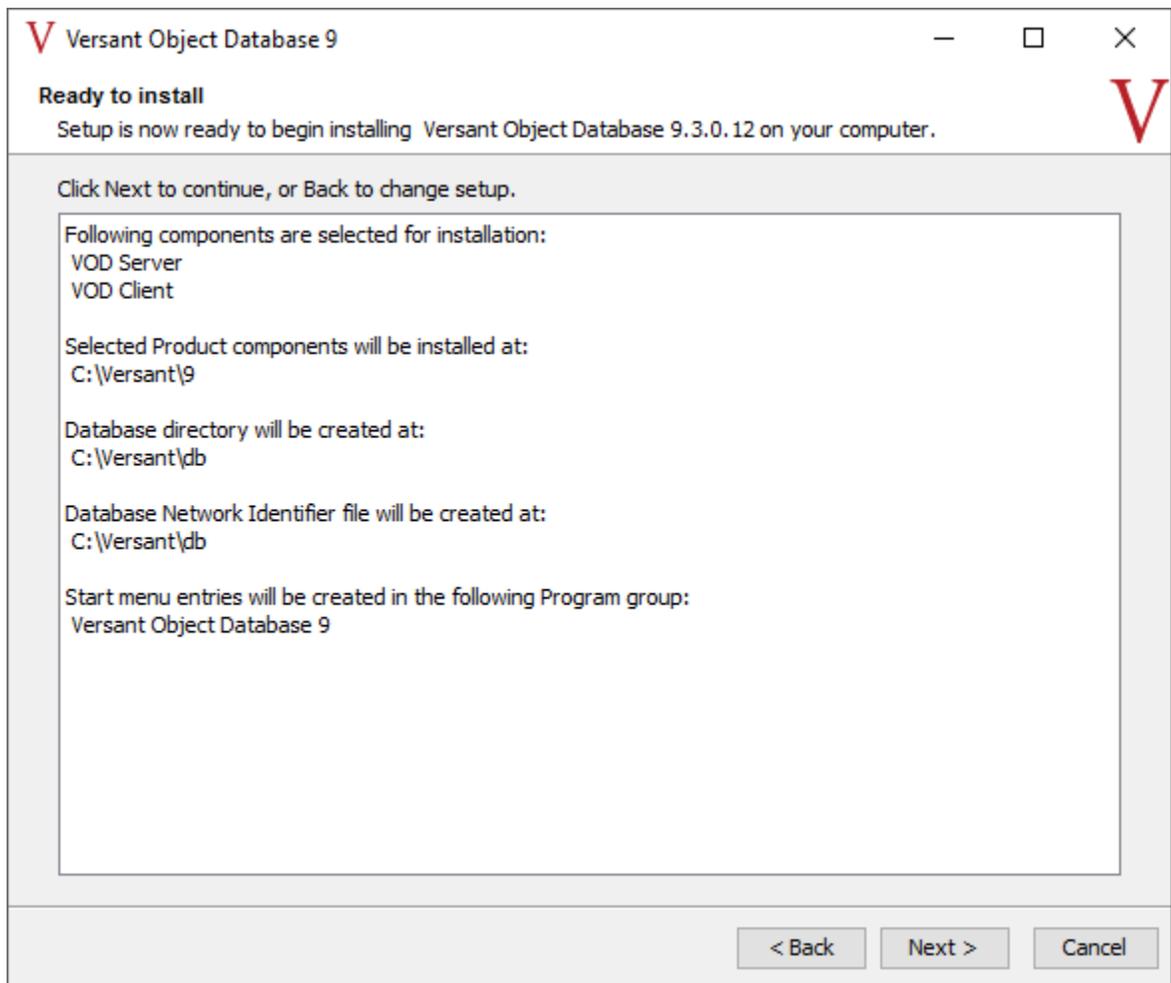


8. On the **Versant Server Configuration** screen, specify the location where you want to set up the database directory. Ensure that you have sufficient disk space for the future growth of your database.
Click **Next**.

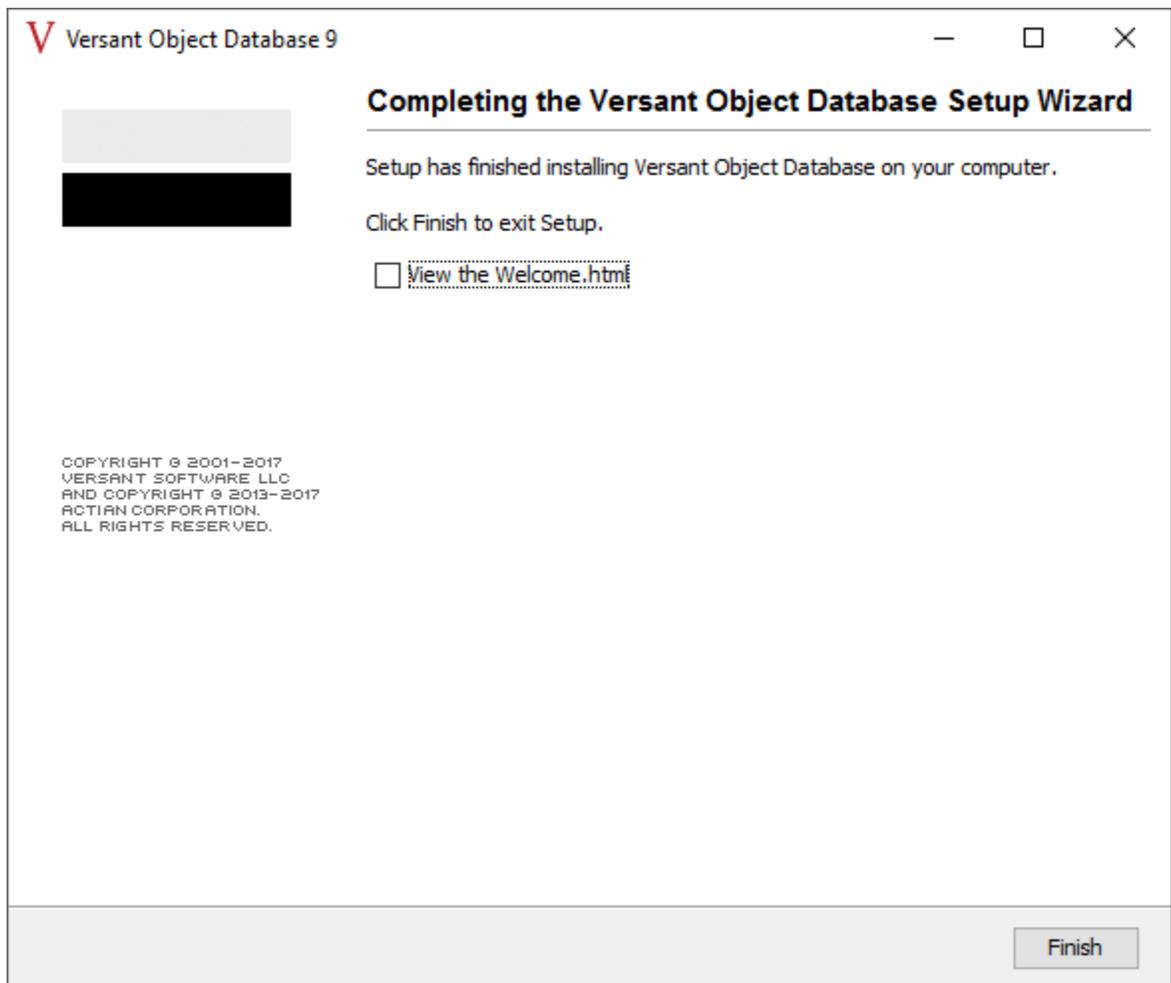


9. On the **Ready to install** screen, review the options that you have selected to install and click **Next**.

Wait for the installation process to complete.



10. On the **Completing the Versant Object Database Setup Wizard** screen, clear the **View the Welcome.html** check box and click **Finish**.



 When newly installing VOD 9.3.0.12 on Windows Server 2016 64-bit, you may encounter an error dialog after clicking this the "Finish" button on the final step of installation. The title of the error dialog would be "Setup", and the message would be "X com.intstall4j.runtime.beans.actions.finish,ShowFileAction". This message is of no consequence, and should be disregarded. The error dialog can be dismissed with no further action required.

11. Restart your computer.

 Although Versant does not prompt you to restart your computer, you must restart it for proper functioning of Versant.

The following services are created on installing Versant.

- **Versant Agent**
- **Versant Database Connector**

Continue installing Architect/Requirements according to the instructions in [Installing the Architect/Requirements Server](#).

After installing Architect/Requirements, you must install the Versant 8 license file.

Post-Versant Installation Steps

The last step in the Versant installation is to copy the Versant license file. This step should be completed after installing the Architect/Requirements server because the license file is included in the Architect/Requirements server installation.

Verifying the Versant Installation

Run the following commands by logging on to the Versant database server computer with the same user ID that you used to perform the Versant installation.

If Versant is installed properly, you should see several lines of output indicating the various Versant-related paths that you entered during the installation, such as Versant root path, Versant runtime path, and Versant database directory.

If you see an error message, restart the Versant service.

- **Database Server:**

Run the following commands to test the Versant installation:

```
o oscp -i @localhost
```

Sample output for `oscp -i @localhost`:

```
Versant Product Version: 9.3.0
Versant Root Path: D:\Versant\9
Versant Runtime Path: D:\Versant\9
Versant DB Directory: D:\Versant\db
Versant osc-dbid node name: hostname
Versant osc-dbid path: D:\Versant\db
```

```
o itest -v <dbserverhostname>
```

Sample output for `itest -v <dbserverhostname>`:

```
host name: localhost IP addr: xxx.xxx.xxx.xxx
Connection to <dbserverhostname> successful.
```

```
o vinfo -l
```

Sample output for `vinfo -l`

```
PRODUCT: Versant Object Database
-----
VERSION          9.3.0.12.2561
OS                Windows
BUILD INFO       VC 14.0 JDK 1.7 64bit
PATCH DATE      Feb 09, 2018
```

Included Components and Versions

```
-----
ODB              9.3.0.12.2561
JDO              9.3.0.12.2561
JVI              9.3.0.12.2561
VAR              9.3.0.12.2561
GUI              9.3.0.12.2561
```

- **Architect/Requirements Server:**

The following commands can be used to ensure that the Architect/Requirements Server can communicate with the database server across the network:

- o `oscp -i @<dbserverhostname>`

Sample output for `oscp -i @localhost`:

```
Versant Product Version: 9.3.0
Versant Root Path: C:\Versant\9
Versant Runtime Path: C:\Versant\9
Versant DB Directory: C:\Versant\db
Versant osc-dbid node name: localhost
Versant osc-dbid path: C:\Versant\db
```

- o `itest -v <dbserverhostname>`

Sample output for `itest -v <dbserverhostname>`:

```
host name: <dbserverhostname> IP addr: xxx.xxx.xxx.xxx
Connection to <dbserverhostname> successful.
```

Installing the Architect/Requirements Server

If you are upgrading your Systems Architect/Requirements Management installation from an earlier version, follow the instructions in [Upgrading the Installation](#).

Before beginning the Systems Architect/Requirements Management server installation process, ensure that you install the Versant object database software. The Versant installation is no longer a part of the Architect/Requirements server installer.

The installers are on the Architect/Requirements distribution archive.

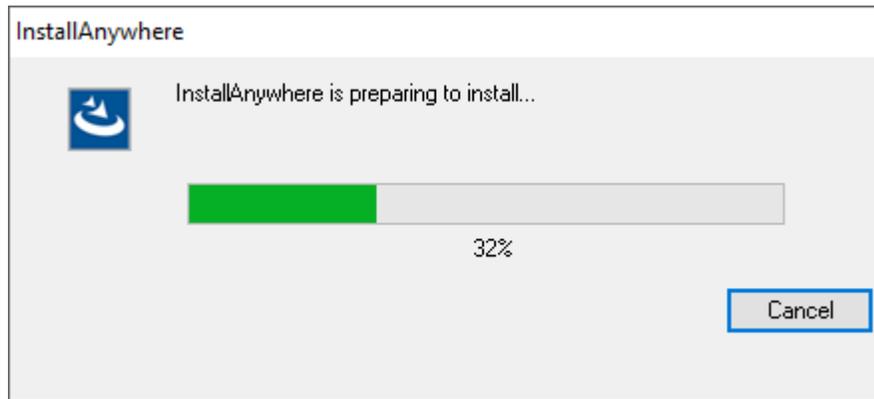
The installer is located in the *Root-Folder\Server\Windows folder*.

Installing the Server

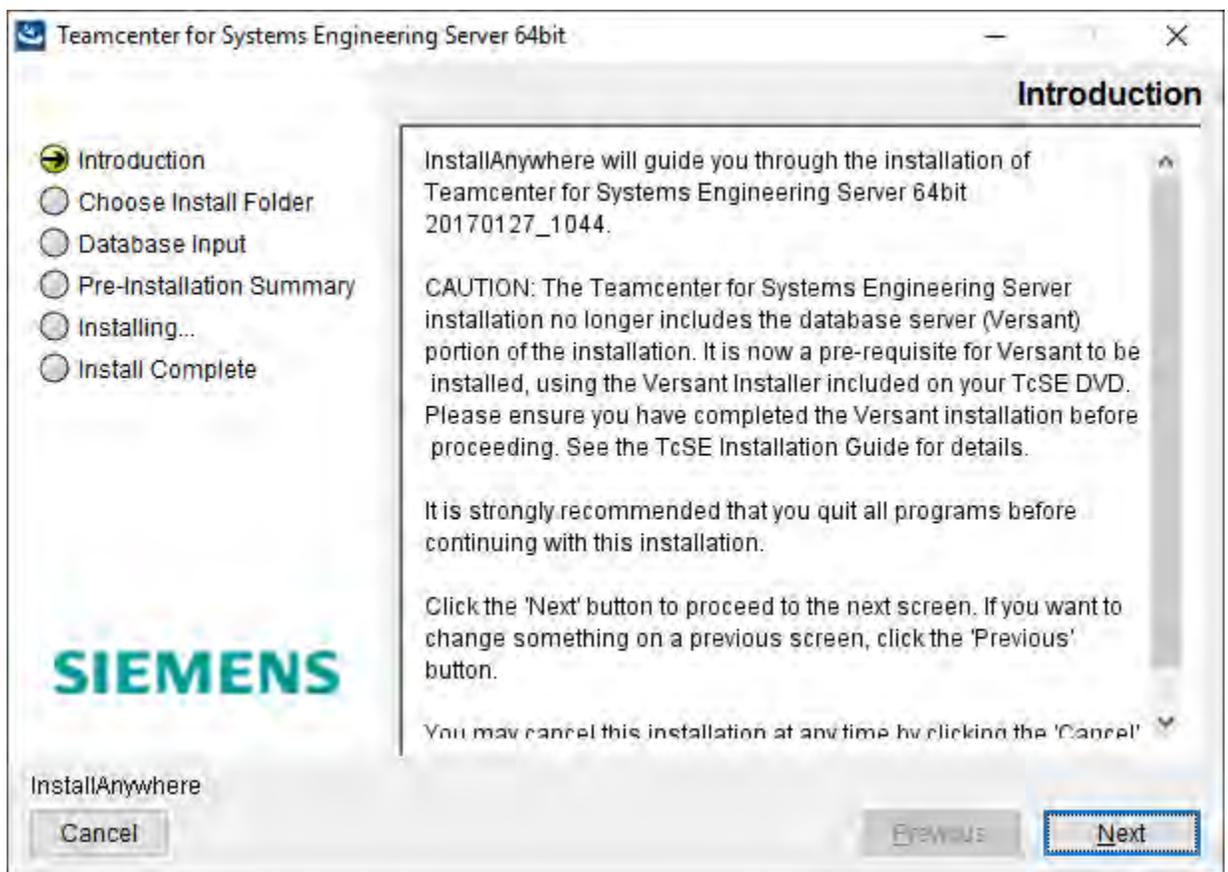
To start the Systems Architect/Requirements Management Server installation process, perform the following steps:

1. The installer file is **TcSE_Server_64.exe**. Execute the installer file.

It may take several minutes before the installation window appears.



2. In the **Introduction** screen, review the notes and click **Next** to begin the installation.



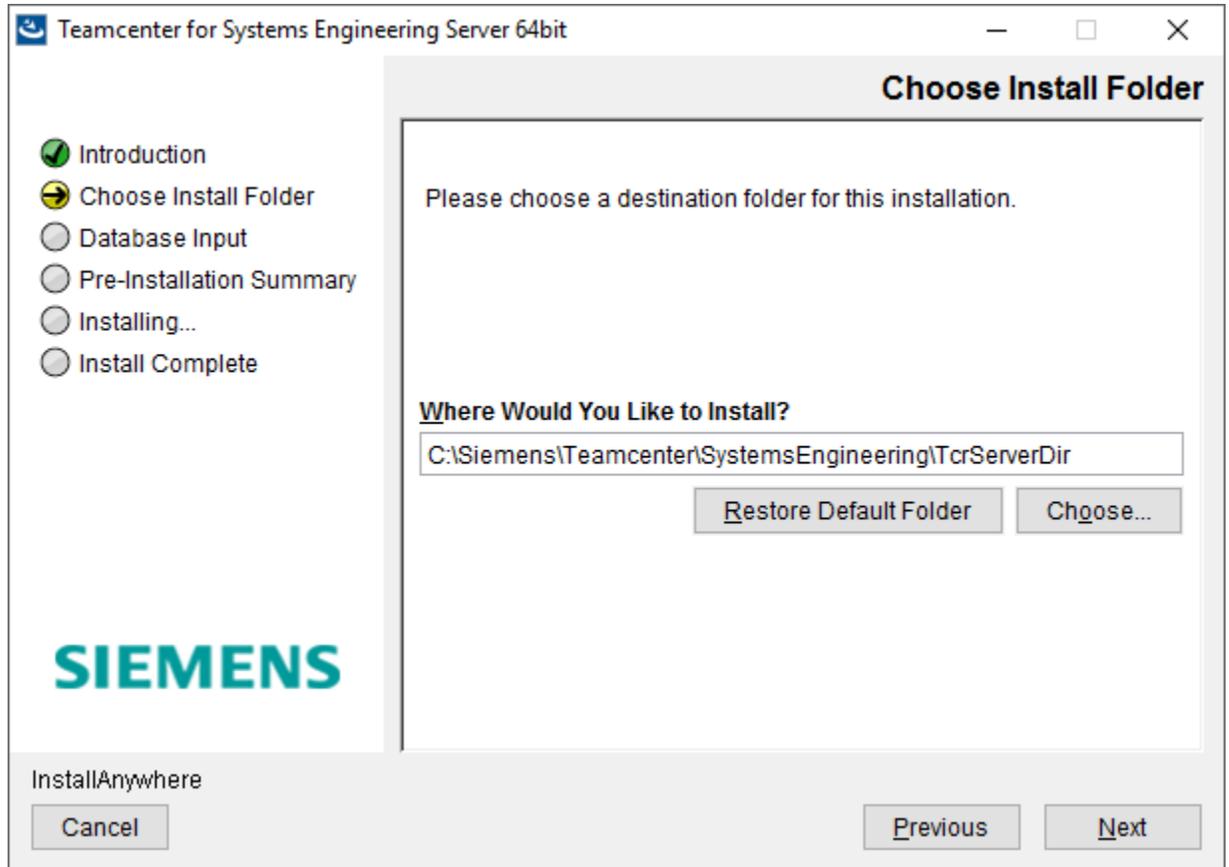
3. In the **Choose Install Folder** screen, click **Choose** to select the installation directory.

The installation path must not contain spaces.

To revert to the default location displayed by the installer, click **Restore Default Location**.

To revisit the previous options presented by the installer, click **Previous**.

Click **Next**.



4. In the **Database Input** screen, enter the name of the database server's host name and the name of the database.

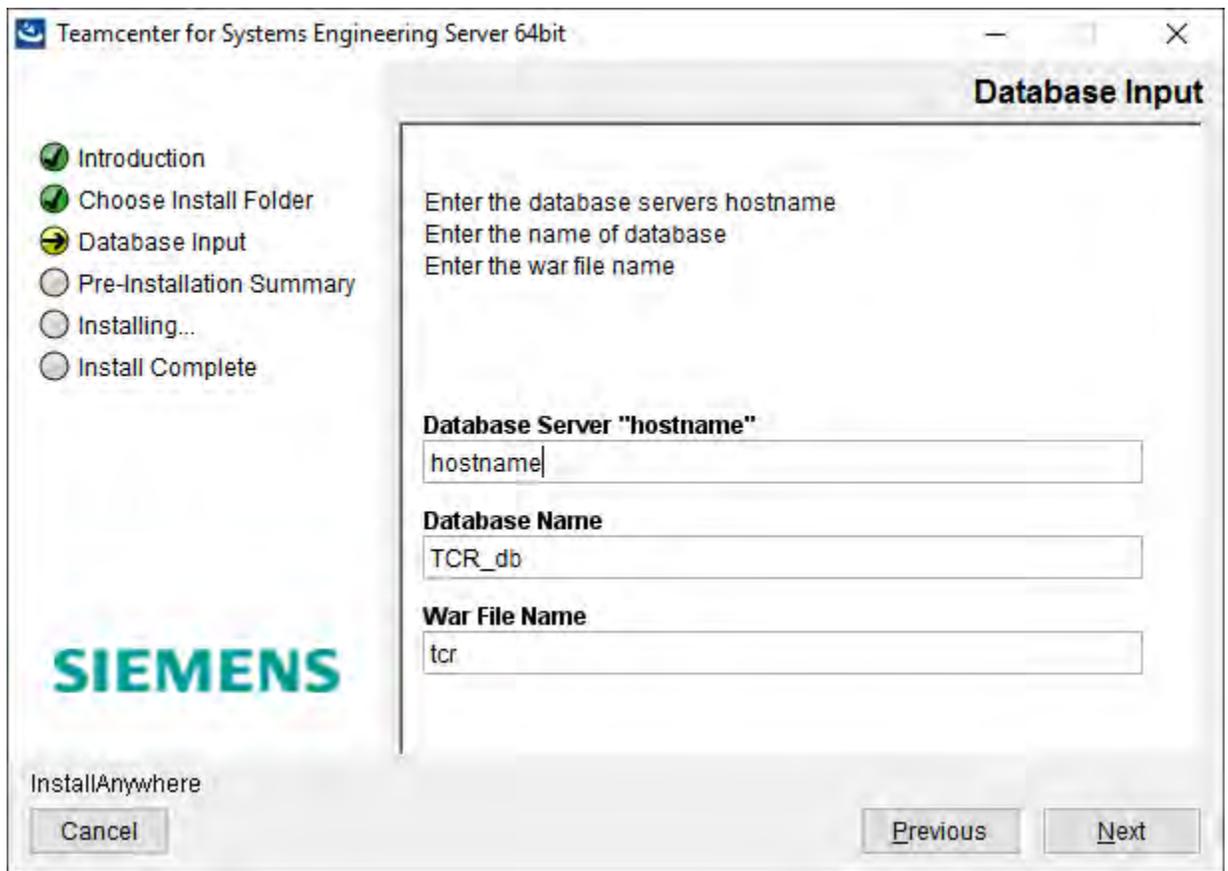
The installer defaults to the local host name and **TCR_db**. This should be correct if you are installing the application server and database server on one machine. If the database is on a separate machine, enter the information applicable to that machine.

It is not necessary for the database to already exist at this point in the installation process. The installer needs this information to update the **war** file with the information needed to connect to the database after Architect/Requirements is deployed.

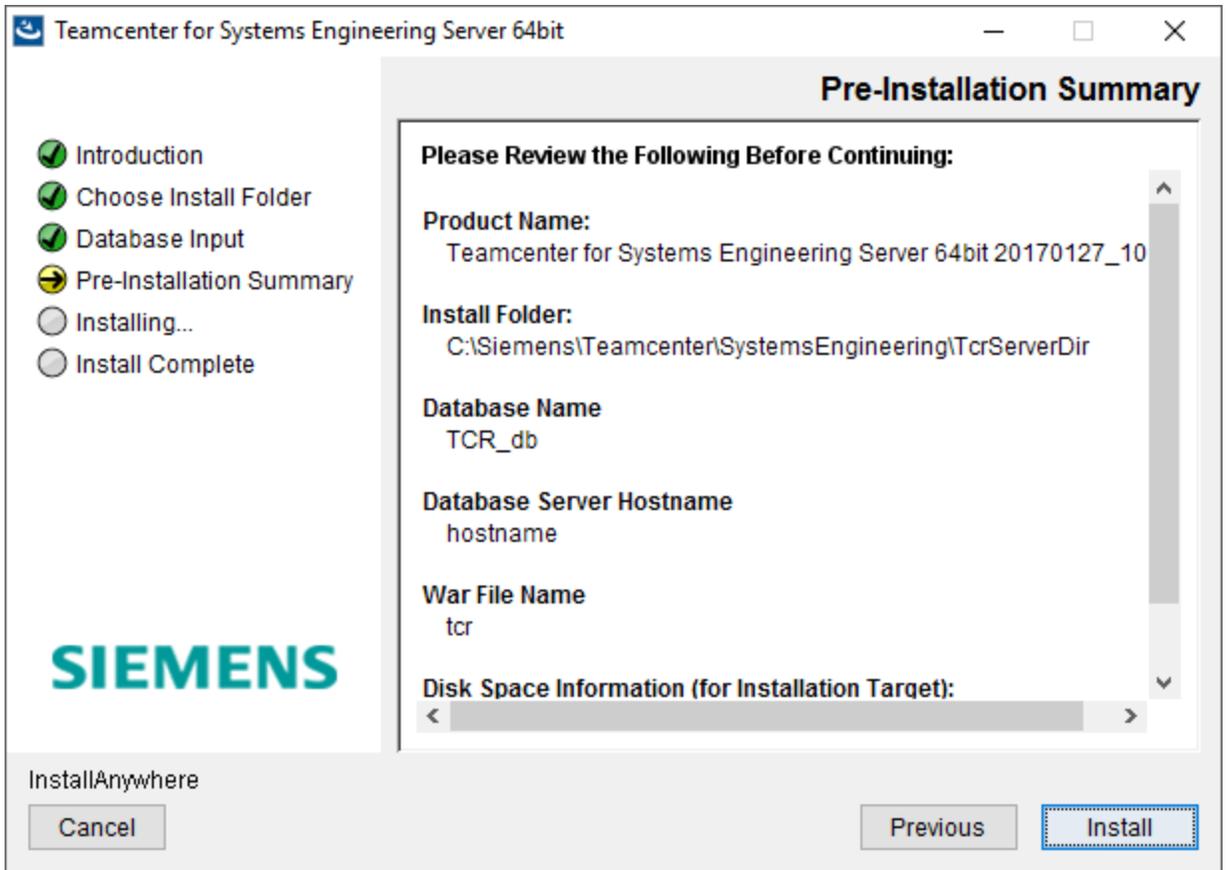


The **Database Server "hostname"** that you specify must be the same name that is used for the Versant installation.

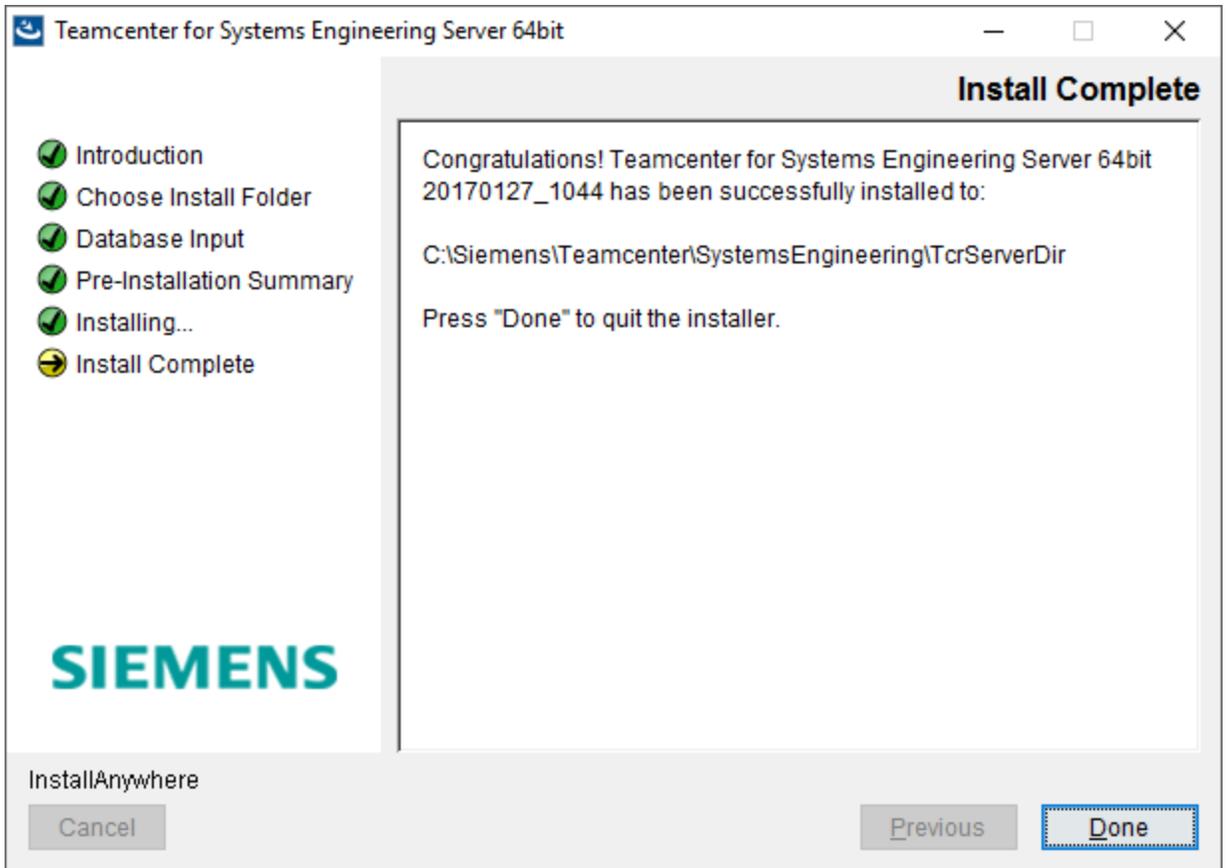
Click **Next**.



The **Pre-Installation Summary** screen displays information such as the product name, installation folder, database name, database server's host name, and disk space availability. Click **Install** to begin the installation.



5. After the installation is successful, click **Done** to exit the installer.



6. Restart your computer to ensure that the updated Path environment variable is propagated for use by you web application server's service process.

Verifying Versant Environment Variables

The Versant installer updates all the required variables during installation. You can refer to the information below to ensure that the environment variables are set correctly on the Versant Database server.

The following environment variables must be set for the user who creates and manages the Versant database (typically, this same user also installs and manages Architect/Requirements Server) on Windows machines:

- Verify that **VERSANT_ROOT** points to the location of your Versant installation.
For example, *C:\Versant\9*.
- Verify that **VERSANT_DB** points to the Versant database root directory.
For example, *C:\Versant\db*.
- Verify that **VERSANT_DBID** points to the location of the Versant database network identifier file, **osc-dbid**.
For example, *C:\Versant\db*.
- Verify that **VERSANT_DBID_NODE** has the name of the machine hosting the Versant database network identifier file, **osc-dbid**.
For example, *myVersantServer.myDomain.com*.
- Verify that the following are added to the **PATH** system environment variable:
 - Versant client **bin** directory installed with the Architect/Requirements server.
The default location of the Versant client **bin** directory is:
C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\jvi_root\bin
 - Versant server **bin** directory installed with the Versant object database server.
The Versant server **bin** directory is defined as *%VERSANT_ROOT%\bin*.

These variables must be set on all Versant database server computers.

Copying the Versant License File

The default location of the license file (**license.xml**) is *C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\versant_license\license.xml*. Copy the **license.xml** file located in the **versant_license** folder (in the Architect/Requirements installation folders) to the *VERSANT_ROOT* folder (for example, *C:\Versant*). This file is needed to enable Versant.



You cannot create a database if you skip this step.

Initializing the Versant Database

You need to create a new database only in case of a new installation. To create and initialize the database, perform the following steps:

1.

Log on to the application server with the ID of the user who owns and manages the Versant database server.

2. From the command prompt, change the directory to the **schema** directory of your server installation. On the command prompt, enter the following command:

```
tcradmin -action initDB -logToStdOut
```

To test whether the database (for example, **TCR_db**) is successfully created, log on to the database server, and enter the following command on the command prompt:

```
dbinfo -p TCR_db
```

If the database creation is successful, the following output is displayed:

```
VERSANT Utility DBINFO Version 8.0.2  
Copyright (c) 1988-2012 VERSANT Corporation
```

```
Database is in multi-user mode ...
```



At this point, the database is created and initialized. A Systems Architect/Requirements Management user named **tcradm** is created with a blank password and the **Enterprise Administrator** privilege.

When you run the Systems Architect/Requirements Management client to verify the installation or enter licensing information, you must log on as **tcradm**.

Testing the Database

To determine whether the Systems Architect/Requirements Management database is properly created and initialized, log on to the Versant database server, and perform the following steps. Replace **VERSANT_ROOT** with the name of the root directory in which Versant is installed on your system.

1. Run the following command at the command prompt to display all the databases in your Versant installation.

```
dblist
```

Verify that the default database (for example, **TCR_db**) is included in the output. If the output does not include **TCR_db**, the database is not created properly.

2. Run the following command to display information about the database.

```
db2tty -D TCR_db
```



This Versant command line utility can fail if it is not run from a command window launched without using the **Run As Administrator** option.

This command prints several pages of data from the database on the screen if the database is created properly.

In case of any problems with the database, contact support at http://www.plm.automation.siemens.com/en_us/support/gtac/.

Next Steps

Continue with the post-installation procedures such as:

- Deploying the **tr.war** file.
- Entering Architect/Requirements license information.
- Verifying the installation.

The procedures are described in [Post-installation Tasks](#).

Chapter 3: Upgrading the Installation

This chapter describes how to upgrade Architect/Requirements server software from the version 10.0 or later to Architect/Requirements 11.1.

Upgrade Overview

Architect/Requirements requires you to install or upgrade your Versant database separately from the Architect/Requirements installation. The Versant installation files are included on the Architect/Requirements media.



For the upgrade steps that require entering commands, you must type the commands and not copy and paste them into a command prompt from the documentation. If you copy and paste, the command may not work as intended.



The following assumptions are made in this section:

- The name of the Architect/Requirements server war file is **tcr.war**. While the Architect/Requirements installer allows you to use any name, **tcr.war** is the typically used name.
- The name of the Architect/Requirements Versant object database is **TCR_db**. While the Architect/Requirements installer allows you to use any name, **TCR_db** is the typically used name.
- The *system 1* refers to the state of the system before the upgrade procedure, and the *system 2* refers to the state of the system following the upgrade procedure.

User Privileges for Running the Installations

Users installing and configuring the Architect/Requirements server and the Versant database server must be logged on to the systems with administrator privileges.

The Architect/Requirements server installer should be run with a user ID that is authorized to manage both Architect/Requirements and Versant. This user ID may be different from the user ID used to install the Versant database.

Upgrade Planning

Upgrading your existing Architect/Requirements installation to the latest version will include the following activities:

- Upgrading the operating system.

For information about the latest supported version of the operating system, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- Upgrading the application server from Architect/Requirements 10.1 or later to Architect/Requirements 11.1.
- Upgrading the Versant Object Database application to version 9.3.

For information about the latest supported version of the Versant, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

Upgrading From Architect/Requirements 10.1 or Later Versions

If you are installing and configuring additional computers as Architect/Requirements servers or Versant database servers, you should upgrade both Versant and Architect/Requirements server software on each of the computers.

You can follow the steps below that are applicable for your corresponding configuration.

On the computer configured as both the Versant database server and the Architect/Requirements server:

- . Back up any important data
- . Undeploy the existing **tcr.war** file on the application server.
- . Uninstall the existing Architect/Requirements server.
- . Uninstall Versant 8.0.2
- . Install Versant 9.3
- . Install Java.

For information about the supported version of Java, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

- . Upgrade the Web application server (if required).
- . Install the latest version of the Architect/Requirements server software.

In the Database Input dialog window, ensure that the host name of the Versant database server (installed and configured on a separate computer) and the name of the database are correct.

- . Run the Schema upgrade command.

- . Deploy the new **tcr.war** file on the application server.
- . Start the Web application server.
- . Launch Architect/Requirements and update the Web configuration parameters as needed.

On every computer configured as the Versant database server:

- . Uninstall Versant 8.0.2
- . Install Versant 9.3

On every computer configured as the Architect/Requirements server:

- . Back up any important data.
- . Undeploy the existing **tcr.war** file on the application server.
- . Upgrade the Web application server (if required).
- . Install the latest Architect/Requirements server software.

In the Database Input dialog window, ensure that the host name of the Versant database server (installed and configured on a separate computer) and the name of the database are correct.

- . Review and set the environment variables if needed.
- . Deploy the new **tcr.war** file on the application server.
- . Update the Web application configuration parameters, such as **JRE.Version**.
- . Enter the Architect/Requirements licensing information.
- . Restore any backed-up data.

Upgrade Prerequisite Tasks

You must perform the tasks mentioned in this section before starting the upgrade process.

Backup Customized JSP and Schema Files

Before the upgrade, you must back up any customized **JSP** files or custom schema. You can restore the **JSP** and schema after the upgrade process.

Copy Customized JSP Files

If you have created any custom **JSP** files, or modified any example **JSP** files, copy them to a temporary location before the upgrade process. The example files are overwritten during the upgrade process.

Exporting the Customized Schema Objects

If you have modified any schema objects supplied by Architect/Requirements, such as default templates and change approval activators, you must perform XML schema export of those objects from each project. The upgrade process replaces any standard schema objects that were changed from the previous release, and you must therefore perform the XML schema export.

These objects can be restored by importing them as a post-upgrade step.

Cleaning the Database and Taking a Backup

Before beginning the upgrade, you should review the health of your existing database and back it up.

Major version installations of Architect/Requirements often have schema changes that are not compatible with previous versions. To revert to a previous version, you must restore a Versant backup from that release.

Perform the following steps:

1. Run the Architect/Requirements **maintainDB** script to ensure that there are no critical errors in the database before proceeding to the next step.

```
tcradmin -action maintainDB
```

Verify and review the log file (*TcrAdminLog.html*).

2. Ensure that all users are out of the system and stop your application server.
3. Run the following commands to ensure that all database transactions are complete and that all data is flushed from cache.

```
stopdb TCR_db  
startdb TCR_db  
stopdb TCR_db
```



It is important to ensure that the database is properly shut down. If not, the Versant schema conversion fails.

4. Run the following Versant **check** database command to ensure that there are no database integrity issues before beginning the upgrade:

```
dbtool -check TCR_db
```



This Versant command line utility can fail if it is not run from a command window launched without using the **Run As Administrator** option.

If there are no errors, proceed to the next step. If there are errors, you must resolve them before proceeding with the upgrade.

5. Backup your existing Architect/Requirements database.

The backup is for safety purposes only. It is not used in the upgrade process. Your existing database is upgraded as part of the installation process.

To back up the complete database to a file, run the following command:

```
vbackup -level 0 -device fileName -backup dbname
```

Enter variable values as follows:

Replace *fileName* with the name of the backup file.

Replace *dbname* with the name of the database. The default name of Architect/Requirements database is **TCR_db**.



Database backup can be a time-consuming activity for large databases.

For example,

```
vbackup -device PATH\vbackup_VOD20071424.vbk -backup TCR_db
```



PATH is the directory where the backup file is kept.

6. After the backup, repeat the following commands to ensure that all database transactions are complete and that all data is flushed from cache.

```
stopdb TCR_db  
startdb TCR_db  
stopdb TCR_db
```



You must complete this step now. If you do not complete this step, the database conversion that occurs later in the upgrade procedure fails.

7. Back up all your custom **.jsp** files in the custom directory of the application server (such as BEA WebLogic), *application-server-installation-directory\tr\custom*.



The contents of this directory are removed when you undeploy the previous **tr.war** file and deploy the new one. You must use the backup files as replacements after the upgrade.

8. If the Sales and Services **DeltaMaintainDB** is used, remove it.

For more information on removing the Sales and Services **DeltaMaintainDB**, contact the *Global Siemens Sales and Services* team.

Upgrading the Operating System

If you do not currently have a Windows server operating system supported for Architect/Requirements server, contact your IT department for operating system deployment recommendations. The upgrade procedure supports either your existing system with an upgraded operating system, or the use of a separate server.

Uninstalling Architect/Requirements Web Applications



If system 1 and system 2 are not on the same machine,, do not uninstall your Architect/Requirements Web applications until you have verified your system 2 configuration.

Uninstalling Architect/Requirements 10.1 or later patch

1. Log on as a system administrator.
2. If the computer is also configured as Architect/Requirements server, stop the application server on which Architect/Requirements is installed.
3. If the computer is also configured as a Versant database server, run the following command to stop the Architect/Requirements database (for example, **TCR_db**) at a command prompt:

```
stopdb TCR_db
startdb TCR_db
stopdb TCR_db
```

4. Remove the Architect/Requirements server software.

Run the uninstaller located under the **Uninstall** folder of the server installation folder (for example, **Uninstall Teamcenter for Systems Engineering Server 64bit**

Release_*version_number***.exe** located in the

C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\Uninstall folder). *version_number* is 10.1 for Architect/Requirements 10.1.

Alternatively, click **Start**→**Control Panel** and double-click **Add/Remove Programs**. Then remove the Architect/Requirements server software (for example, **Teamcenter for Systems Engineering Server 64bit Release_***version_number***)**. *version_number* is 10.1 for Architect/Requirements 10.1.



The Architect/Requirements 10.1uninstaller fails with a Windows error 2 message if there is a problem in locating a JRE. In such a case, insert the **LAX_VM** argument to explicitly specify the JRE used to run the uninstaller. For example, the following command uninstalls the Architect/Requirements web application from its default location using **java.exe** under **JAVA_HOME**:

```
"C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\Uninstall\Uninstall
Teamcenter for Systems Engineering Server 64bit Release_10.1.exe" LAX_VM
"%JAVA_HOME%\bin\java.exe"
```

Undeploying the Existing WAR File

As part of the upgrade procedure, you should undeploy the existing **tcr.war** file before deploying the new **tcr.war** file. Follow the instructions from your application server vendor for removing a Web archive.

If you have any customizations in the **custom** folder inside the WAR file, create a backup for them so that you can add them to the new **tcr.war** file.

Uninstalling the the Versant Object Database 8

Prior to installing the new version of Version Object Database, you must first uninstall the previous version.

Uninstalling Versant Object Database 8

To uninstall the Versant Object Database:

1. Log on to the system 1 database server as the same user who originally installed Versant Object Database 8.
2. Launch the Versant Object Database 8.0.2 Uninstaller

Option 1: From the Control Panel

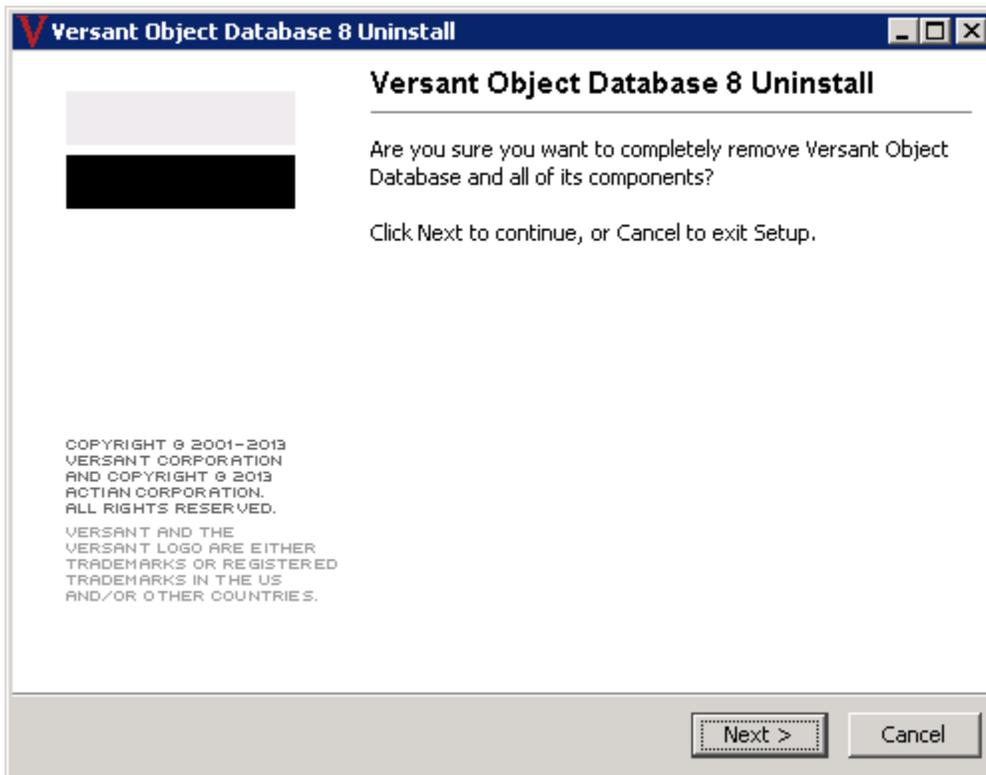
- a) Navitate to Programs and Feature.
- b) Select Versant Object Database
- c) Click Uninstall/Change

Option 2: From the command line

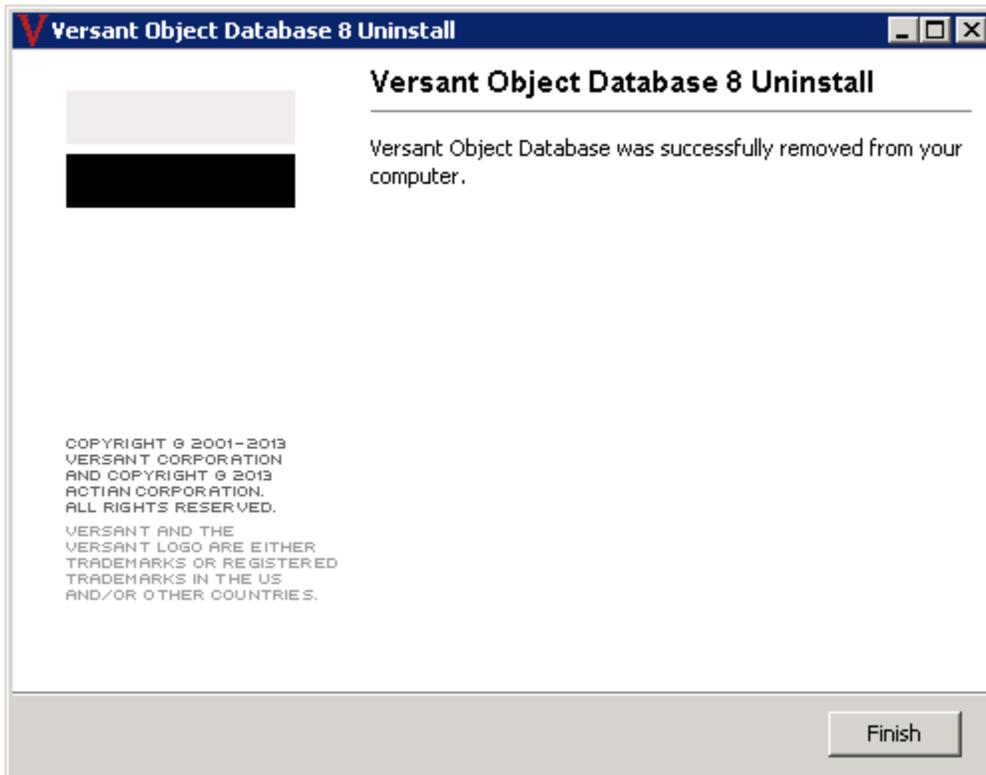
- a) Open a command prompt
- b) Enter the following command:

`%VERSANT_ROOT%\uninstaller\uninstall.exe`

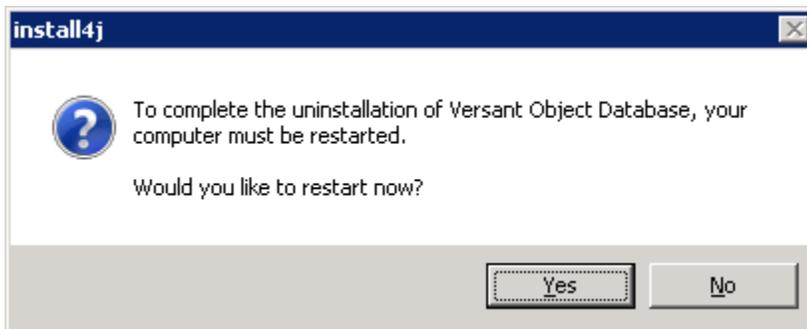
3. On the uninstall wizard, click Next:



4. When the uninstallation is completed, click Finish:



5. When prompted to restart the system, click Yes:



Installing the Versant Object Database 9



Versant Object Database server is required only on a machine configured as a database server.

Architect/Requirements uses an object-oriented database system from Versant Corporation to store and manage data. Architect/Requirements is certified and ships with Versant Database.

The Versant installer is no longer a part of the Architect/Requirements server installer. The Versant database must be installed before installing the Architect/Requirements server. The instructions are same for installing Versant on both the Architect/Requirements server and the database server. If you are

running these on the same machine, perform the installation only once. If you are running these on separate machines, perform this installation on your database server machine as well as each of the Architect/Requirements Server machines.

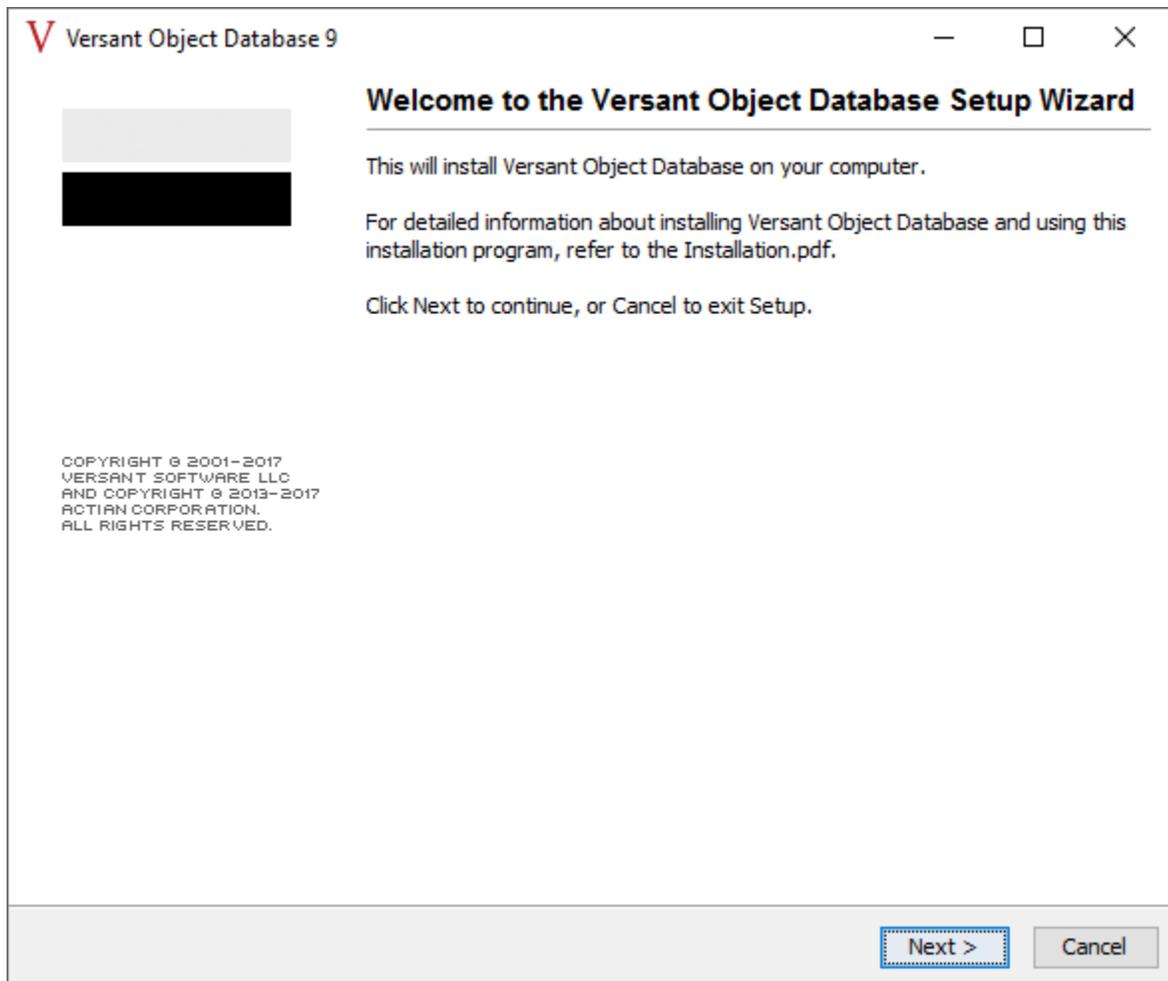
Architect/Requirements 11.1 supports the Versant Object Database 9.3.0 patch 12. The Versant installer is included in the distribution archive in the *Root-Folder***Versant_9**\Windows folder.

The Versant database files must be on a file system that is local to the Versant server processor. Database corruption can occur if you use a remotely mounted drive.

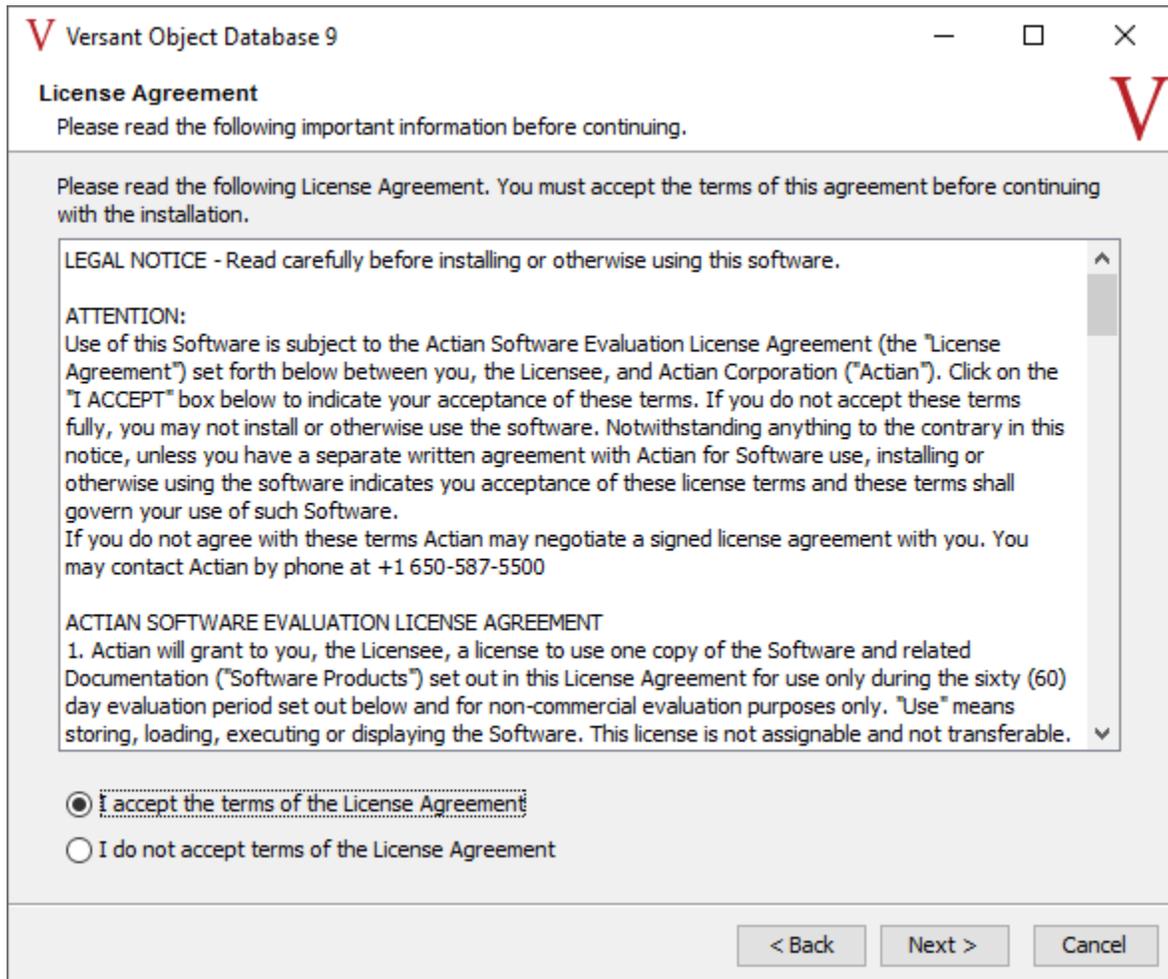
Installing Versant 9

1. Run the Versant installer.
 - Extract **VOD9.3.0.12_2561_Windows_VS2015_64bit_opt.zip** to a temporary folder.
 - As the Administrator, execute **VOD9.3.0.12_2561_Windows_VS2015_64bit_opt.exe**.

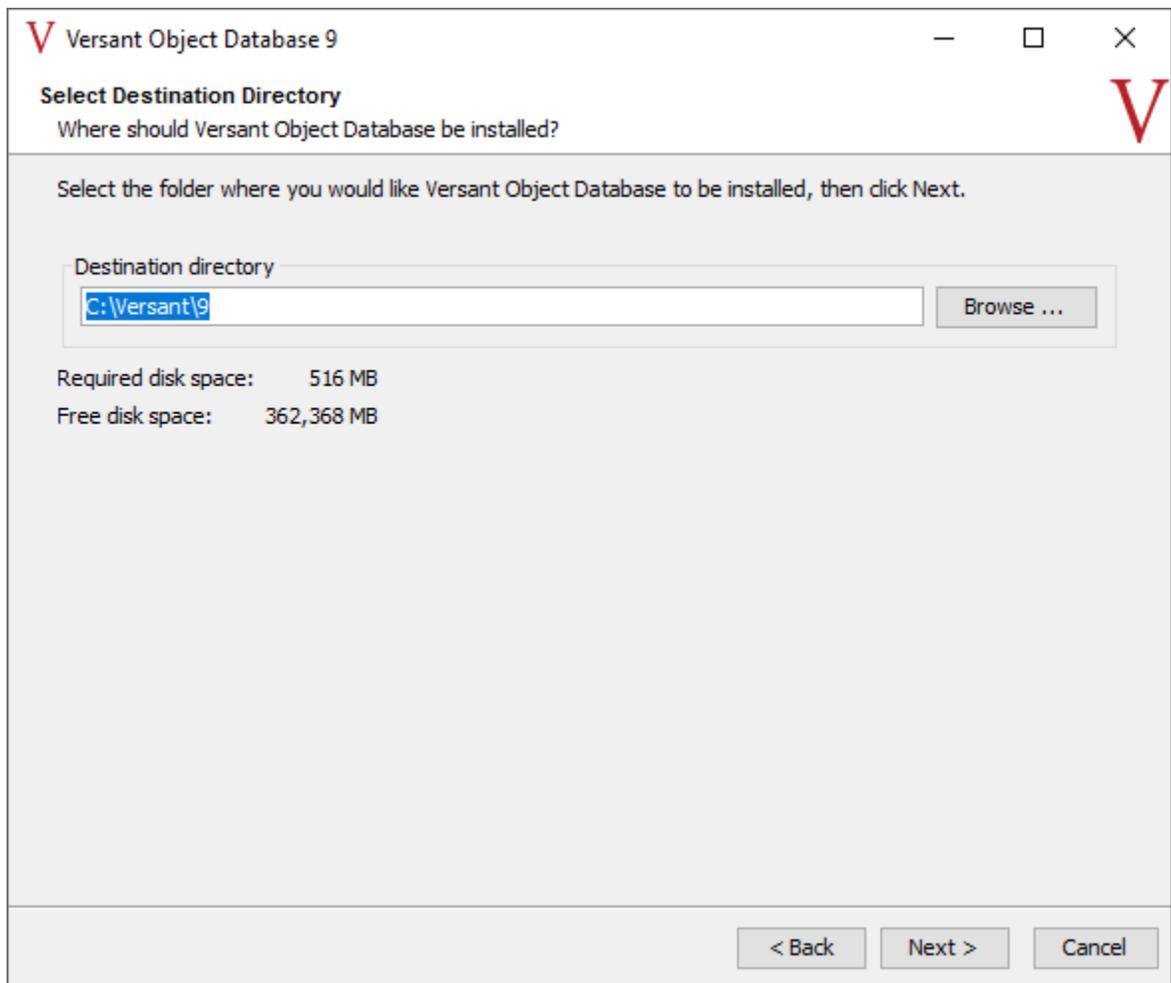
It can take several minutes before the installation window appears.
2. On the **Welcome to the Versant Object Database Setup Wizard** screen, click **Next**.



3. On the **License Agreement** screen, select **I accept the terms of the License Agreement** and click **Next**.



4. On the **Select Destination Directory** screen, specify the location to install the Versant binaries and click **Next**.



5. On the **Select Components** screen, select or clear the check boxes as relevant:

Clear the **SDK** check box.

You may need to click twice in the SDK check box to clear it. Clearing the **SDK** check box clears the **C SDK**, **C++ SDK**, **Java**, and **.NET SDK** check boxes.

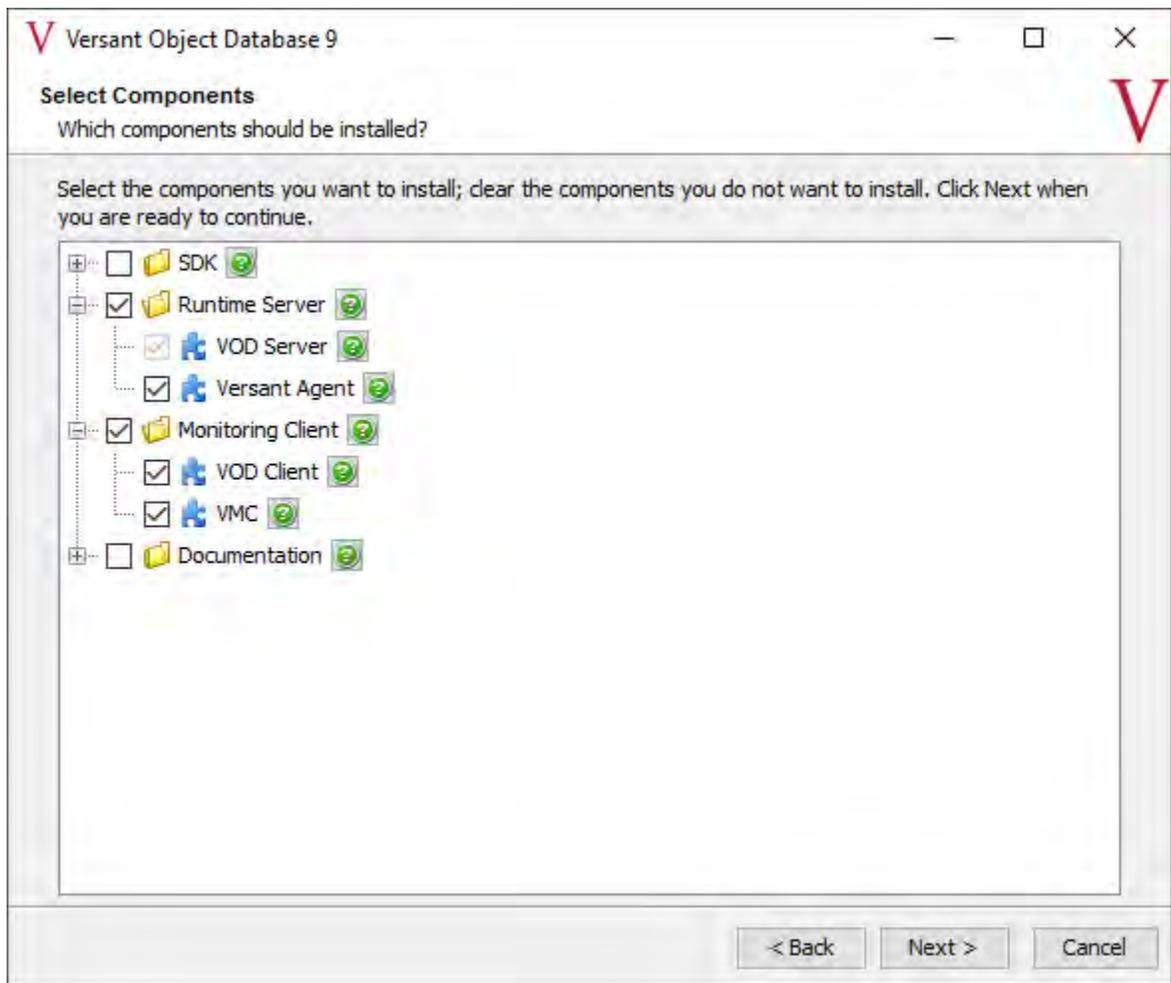
Expand the **Runtime Server** section and clear the **VSQL Server** check box.

Verify that only **VOD Server** and **Versant Agent** are selected.

Expand the **Monitoring Client** section and clear the **VSQL Client** check box.

Verify that only **VOD Client** and **VMC** are selected.

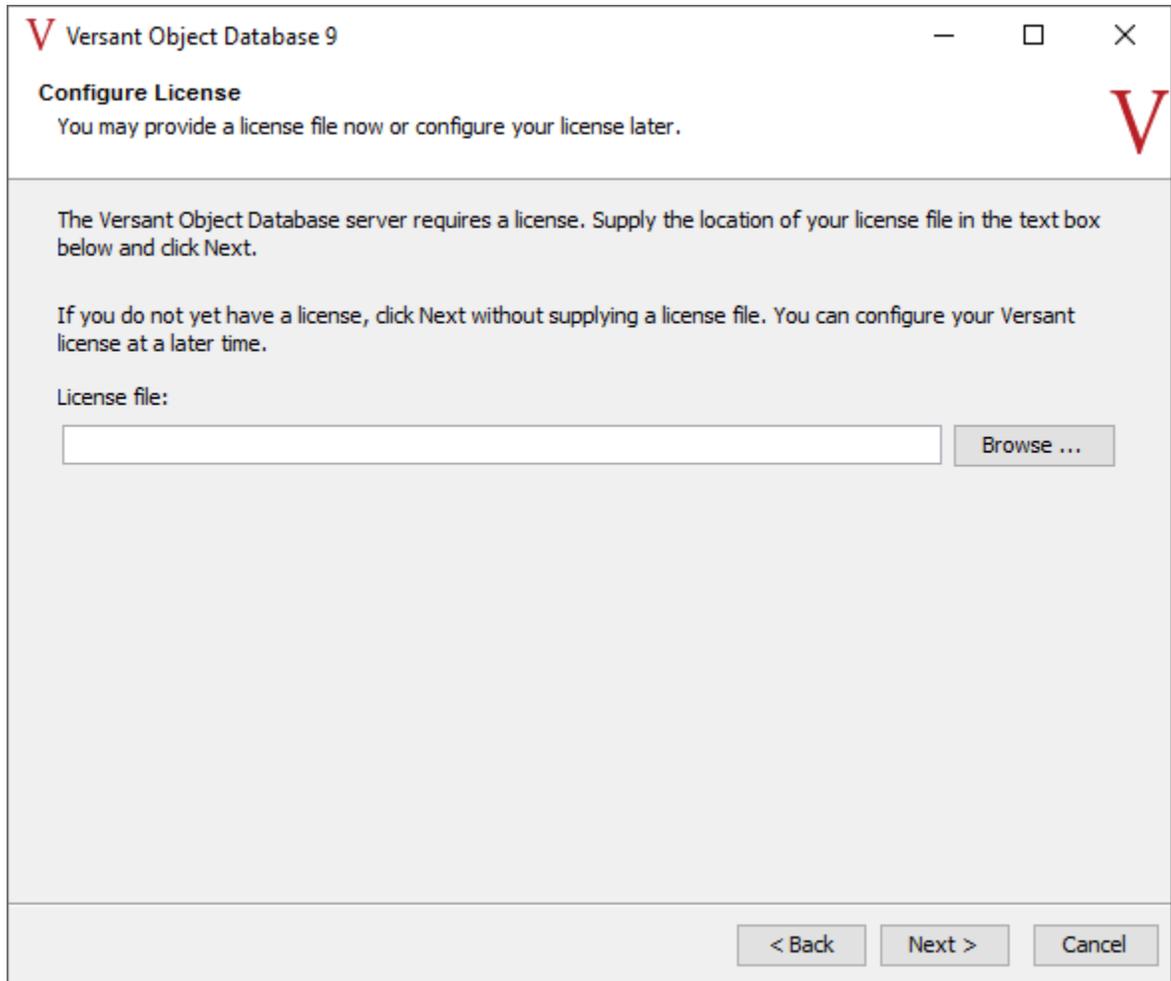
If you do not want the Versant documentation to be installed, clear the **Documentation** check box and click **Next**.



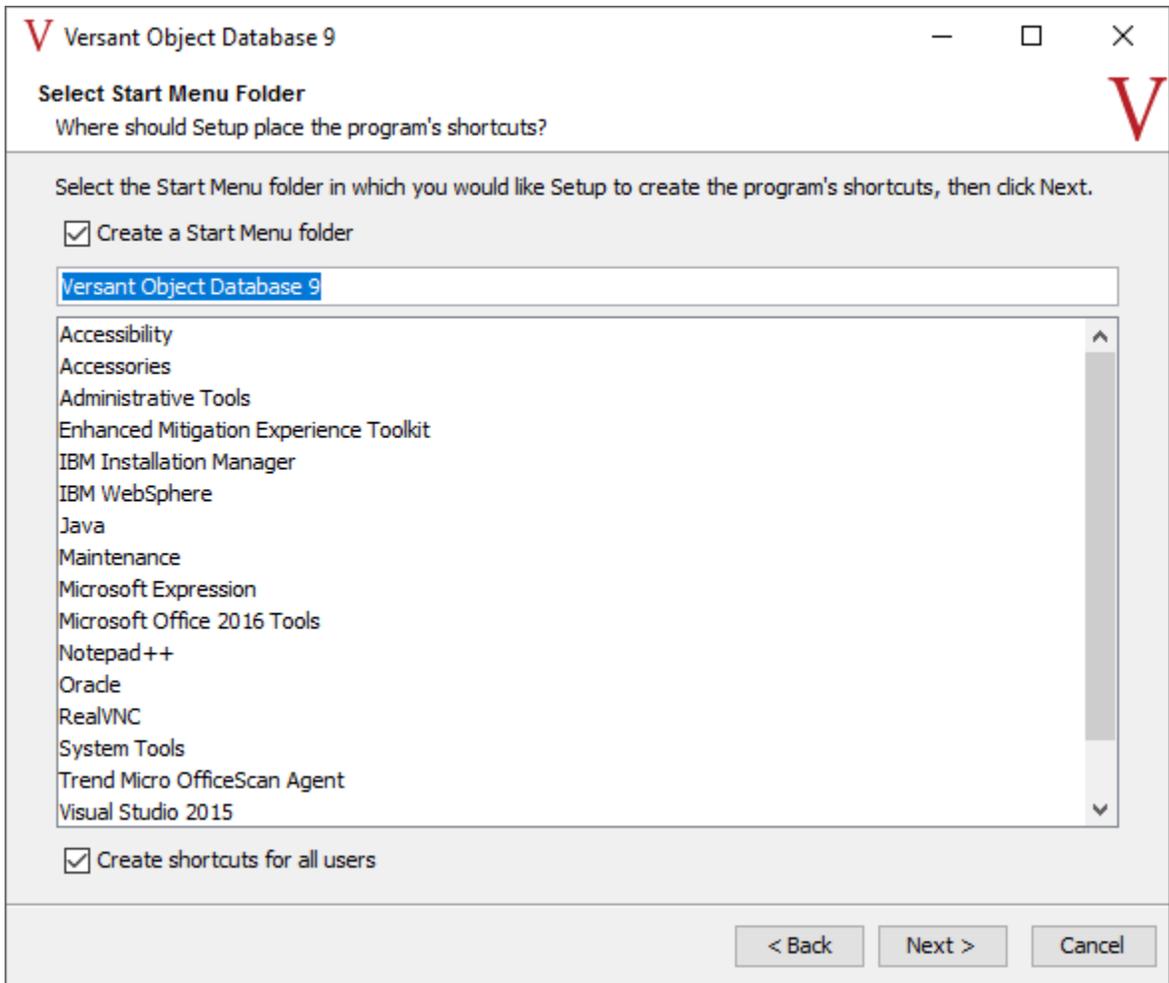
6. On the **Configure License** screen, click **Next**.



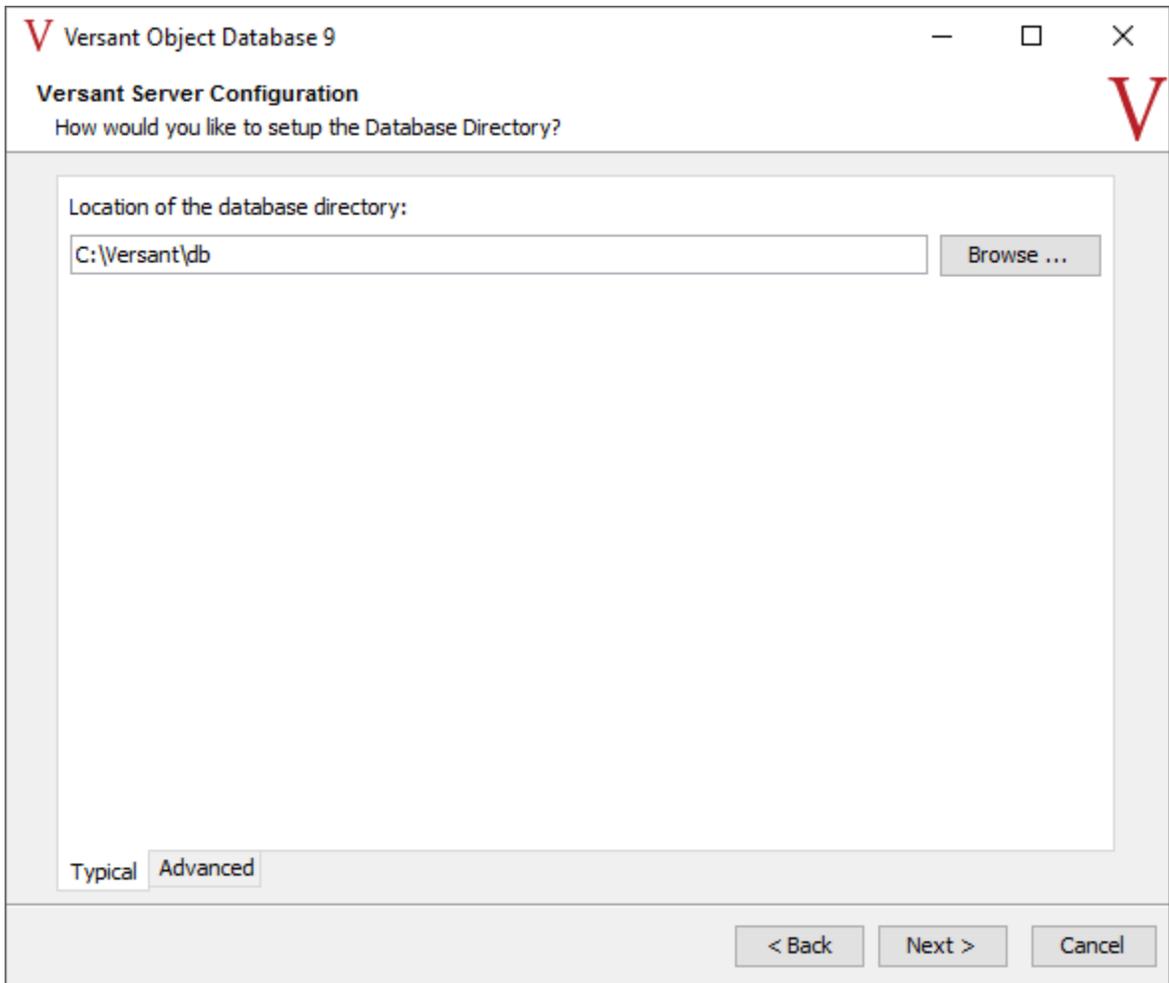
The required Versant license is available after installing Architect/Requirements. You must configure the license after completing the Architect/Requirements installation.



7. On the **Select Start Menu Folder** screen, clear the **Create shortcuts for all users** check box if you do not want the Versant link to be created in the **Start** menu for all the users and click **Next**.

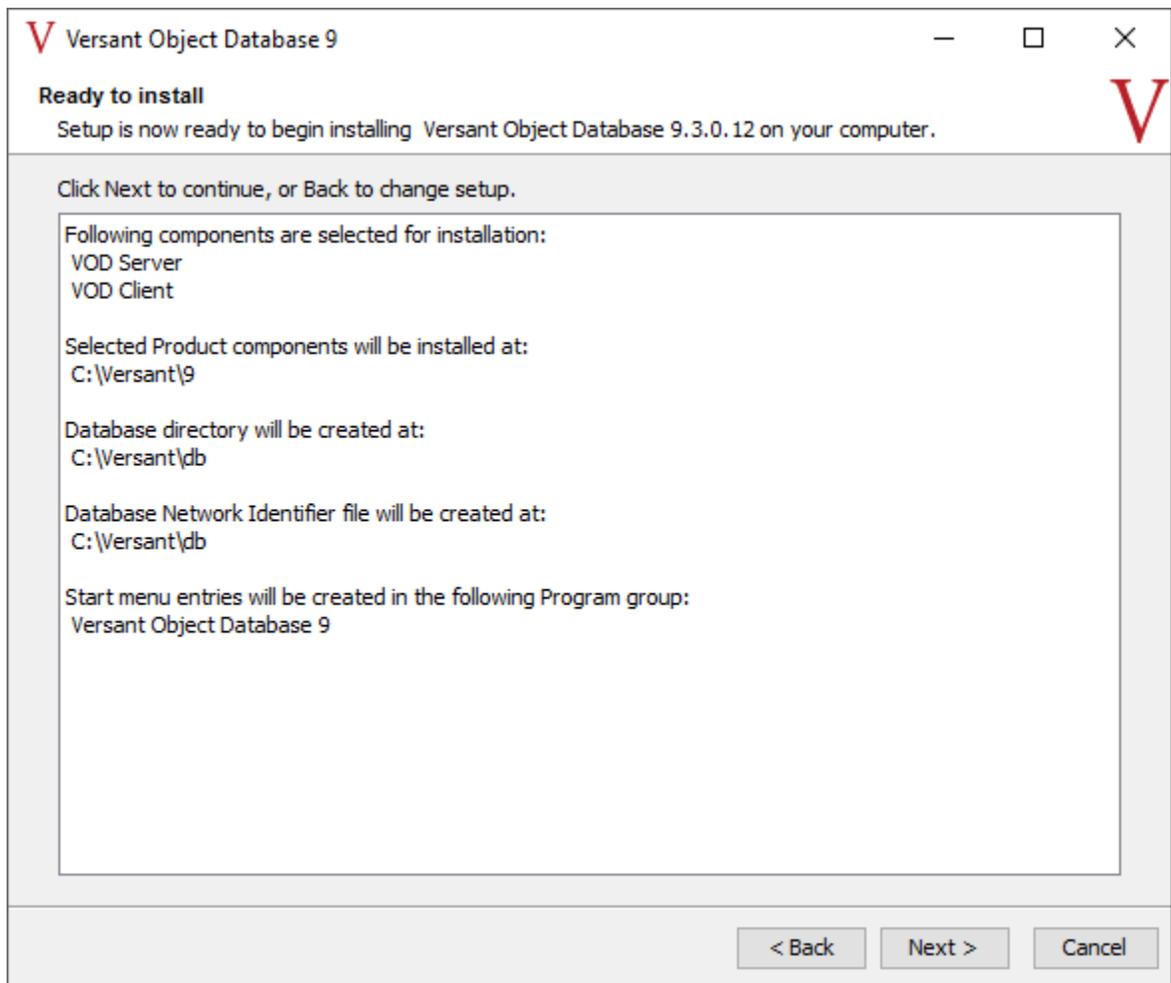


8. On the **Versant Server Configuration** screen, specify the location where you want to set up the database directory. Ensure that you have sufficient disk space for the future growth of your database.
Click **Next**.

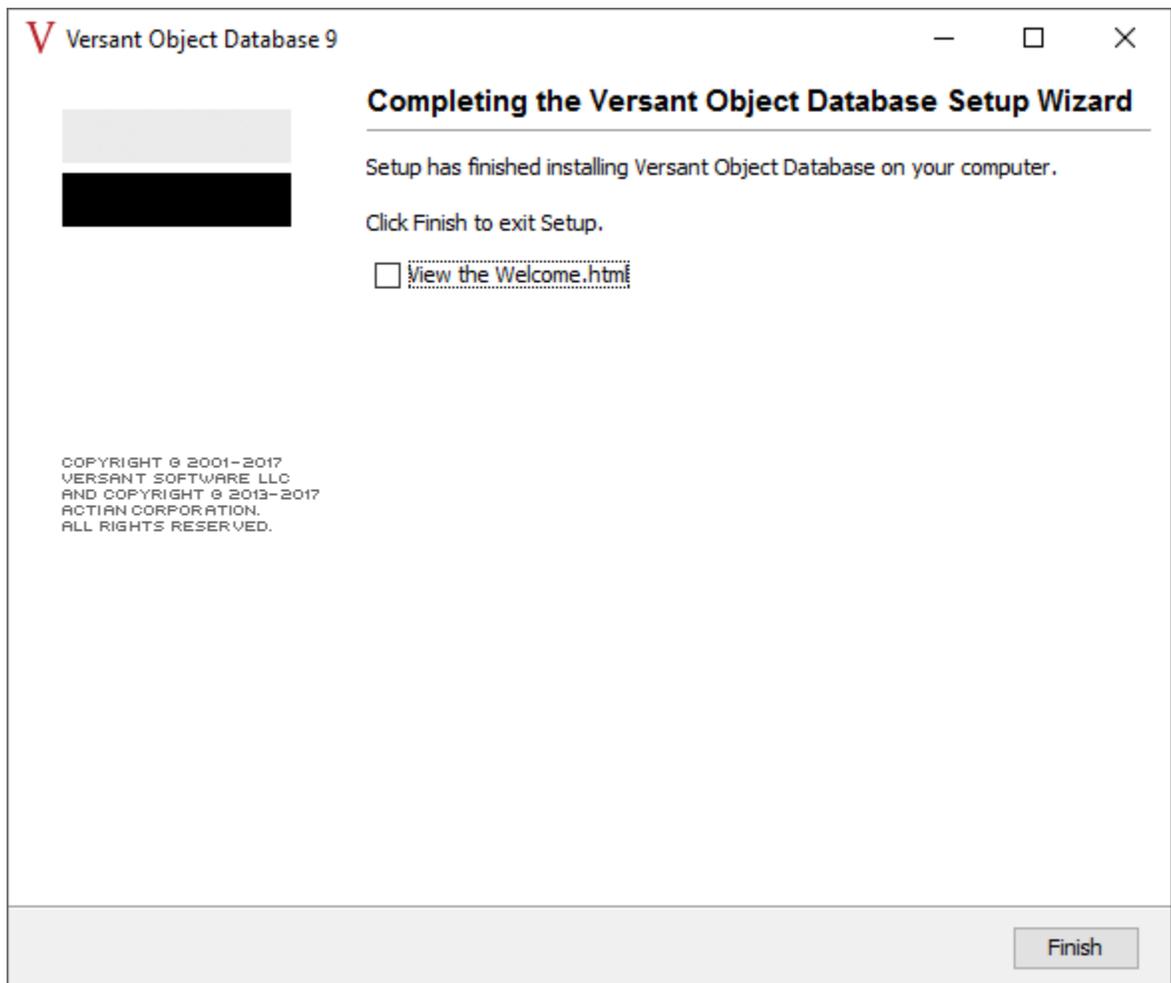


9. On the **Ready to install** screen, review the options that you have selected to install and click **Next**.

Wait for the installation process to complete.



10. On the **Completing the Versant Object Database Setup Wizard** screen, clear the **View the Welcome.html** check box and click **Finish**.



 When newly installing VOD 9.3.0.12 on Windows Server 2016 64-bit, you may encounter an error dialog after clicking this the "Finish" button on the final step of installation. The title of the error dialog would be "Setup", and the message would be "X com.intstall4j.runtime.beans.actions.finish,ShowFileAction". This message is of no consequence, and should be disregarded. The error dialog can be dismissed with no further action required.

11. Restart your computer.

 Although Versant does not prompt you to restart your computer, you must restart it for proper functioning of Versant.

You can verify the Versant installation before continuing with the upgrade process.

Verifying the Versant Installation

Run the following commands by logging in to the computer with the same user ID that you used to perform the Versant installation.

If Versant is installed properly, you should see several lines of output indicating the various Versant-related paths that you entered during the installation, such as Versant root path, Versant runtime path, and Versant database directory.

If you see an error message, restart the Versant service.



Execute the following commands from a new command window.

- **Database Server:**

Run the following commands to test the Versant installation:

o `oscp -i @localhost`

Sample output for `oscp -i @localhost`:

```
Versant Product Version: 9.3.0
Versant Root Path: D:\Versant\9
Versant Runtime Path: D:\Versant\9
Versant DB Directory: D:\Versant\db
Versant osc-dbid node name: hostname
Versant osc-dbid path: D:\Versant\db
```

o `itest -v localhost`

Sample output for `itest -v localhost`:

```
host name: localhost IP addr: xxx.xxx.xxx.xxx
Connection to localhost successful.
```

o `vinfo -l`

Sample output for `vinfo -l`

```
PRODUCT: Versant Object Database
-----
VERSION          9.3.0.12.2561
OS                Windows
BUILD INFO       VC 14.0 JDK 1.7 64bit
PATCH DATE      Feb 09, 2018
```

Included Components and Versions

```
-----
ODB              9.3.0.12.2561
JDO              9.3.0.12.2561
JVI              9.3.0.12.2561
VAR              9.3.0.12.2561
GUI              9.3.0.12.2561
```

Post-Versant Installation Steps

After installing the Versant database application, you must set the environment variables.

The last step in the Versant installation is to copy the Versant license file. This step must be completed after installing the Architect/Requirements server because the license file is included in the Architect/Requirements server installation.

Verifying Versant Environment Variables

The Versant installer updates all the required variables during installation. You can refer to the information below to ensure that the environment variables are set correctly.

The following environment variables must be set for the user who creates and manages the Versant database (typically, it is the user who installs and manages Architect/Requirements Server) on Windows machines:

- Verify that **VERSANT_ROOT** points to the location of your Versant installation.
For example, *C:\Versant\8*.
- Verify that **VERSANT_DB** points to the Versant database root directory.
For example, *C:\Versant\db*.
- Verify that **VERSANT_DBID** points to the location of the Versant database network identifier file, **osc-dbid**.
For example, *C:\Versant\db*.
- Verify that **VERSANT_DBID_NODE** has the name of the machine hosting the Versant database network identifier file, **osc-dbid**.
For example, *myVersantServer.myDomain.com*.
- Verify that the following are added to the **PATH** system environment variable:
 - Versant client **bin** directory installed with the Architect/Requirements server.
The default location of the Versant client **bin** directory is:
C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\jvi_root\bin
 - Versant server **bin** directory installed with the Versant object database server.
The Versant server **bin** directory is defined as *%VERSANT_ROOT%\bin*.

These variables must be set on all Architect/Requirements and Versant database server computers.

Installing Architect/Requirements Server



If system 1 and system 2 are not on the same machine, be sure to install your Architect/Requirements Server on system 2 prior to moving the database. It needs to be in place in order to complete that procedure.

You have now completed uninstalling your existing version. Continue by following the installation steps below.

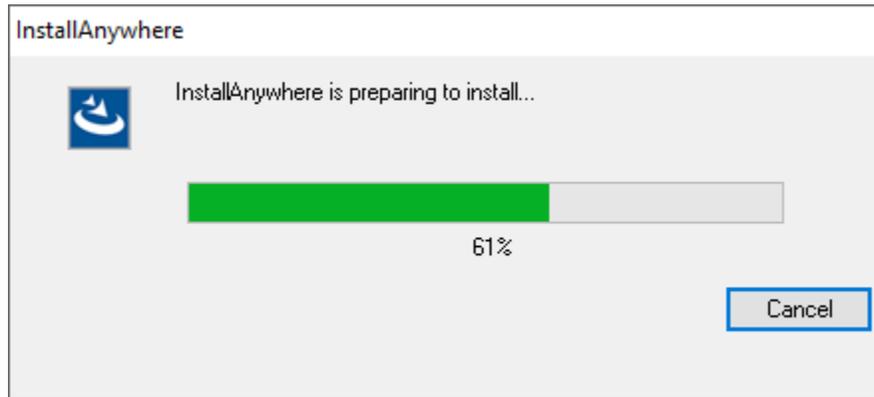
The installers are on the Architect/Requirements distribution archive.

The installer is located in the *Root-Folder\Server\Windows folder*.

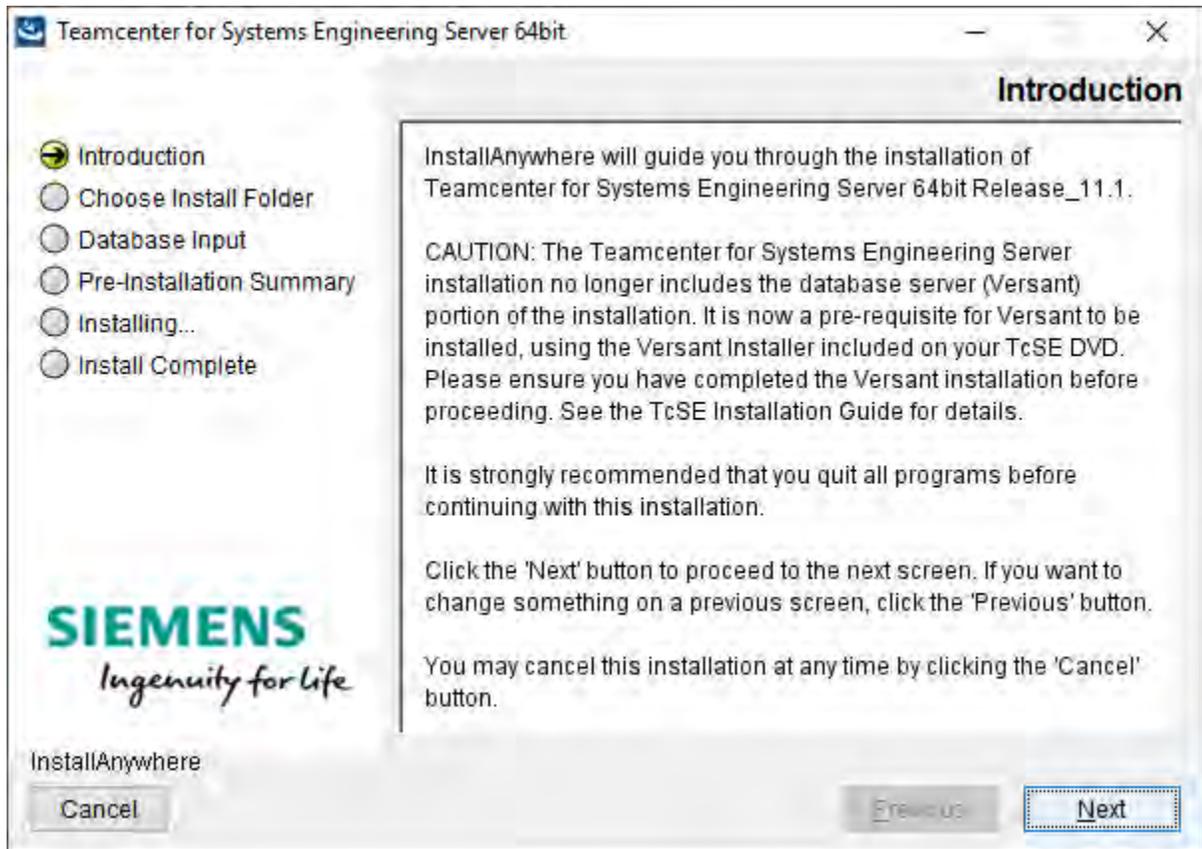
Installing the Server

To start the Systems Architect/Requirements Management Server installation process, perform the following steps:

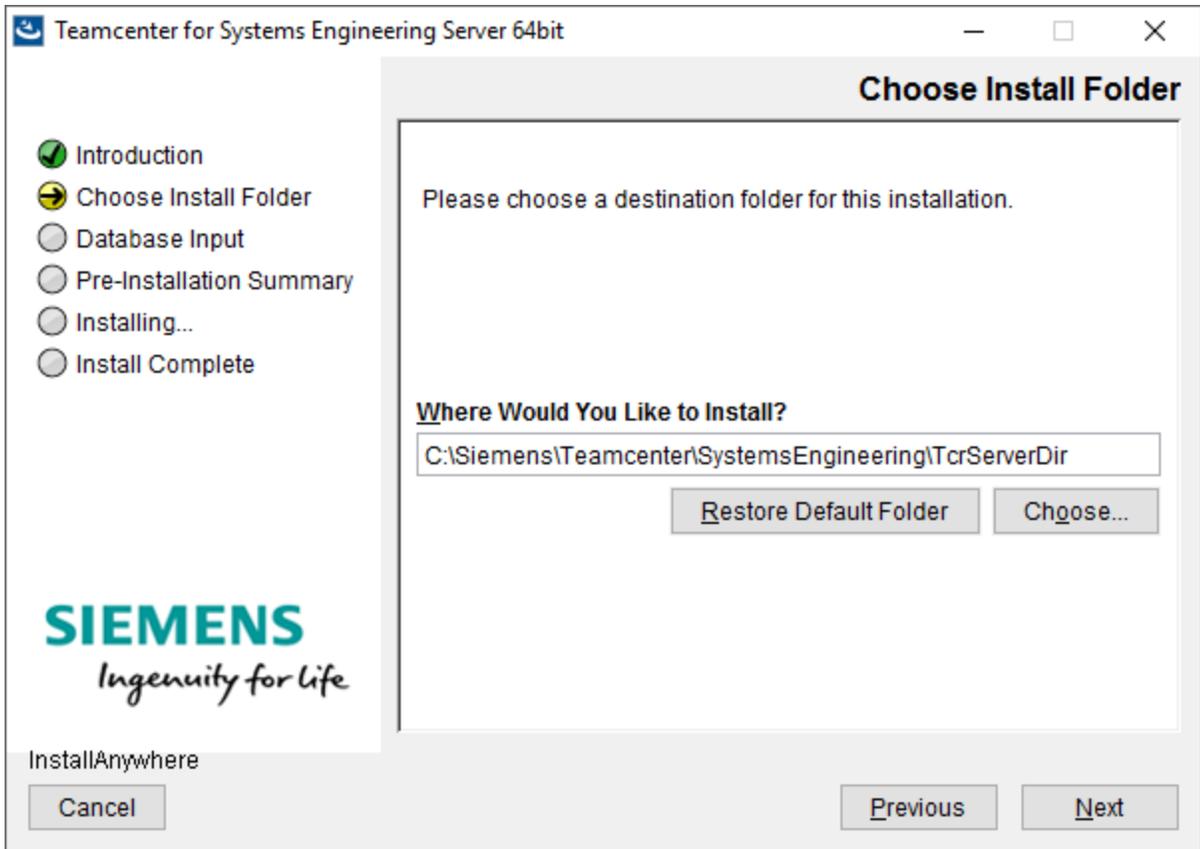
6. The installer file is **TcSE_Server_64.exe**. Execute the installer file.
It may take several minutes before the installation window appears.



7. In the **Introduction** screen, review the notes and click **Next** to begin the installation.



8. In the **Choose Install Folder** screen, click **Choose** to select the installation directory.
The installation path must not contain spaces.
To revert to the default location displayed by the installer, click **Restore Default Location**.
To revisit the previous options presented by the installer, click **Previous**.
Click **Next**.



- In the **Database Input** screen, enter the name of the database server's host name and the name of the database.

The installer defaults to the local host name and **TCR_db**. This should be correct if you are installing the application server and database server on one machine. If the database is on a separate machine, enter the information applicable to that machine.

It is not necessary for the database to already exist at this point in the installation process. The installer needs this information to update the **war** file with the information needed to connect to the database after Architect/Requirements is deployed.

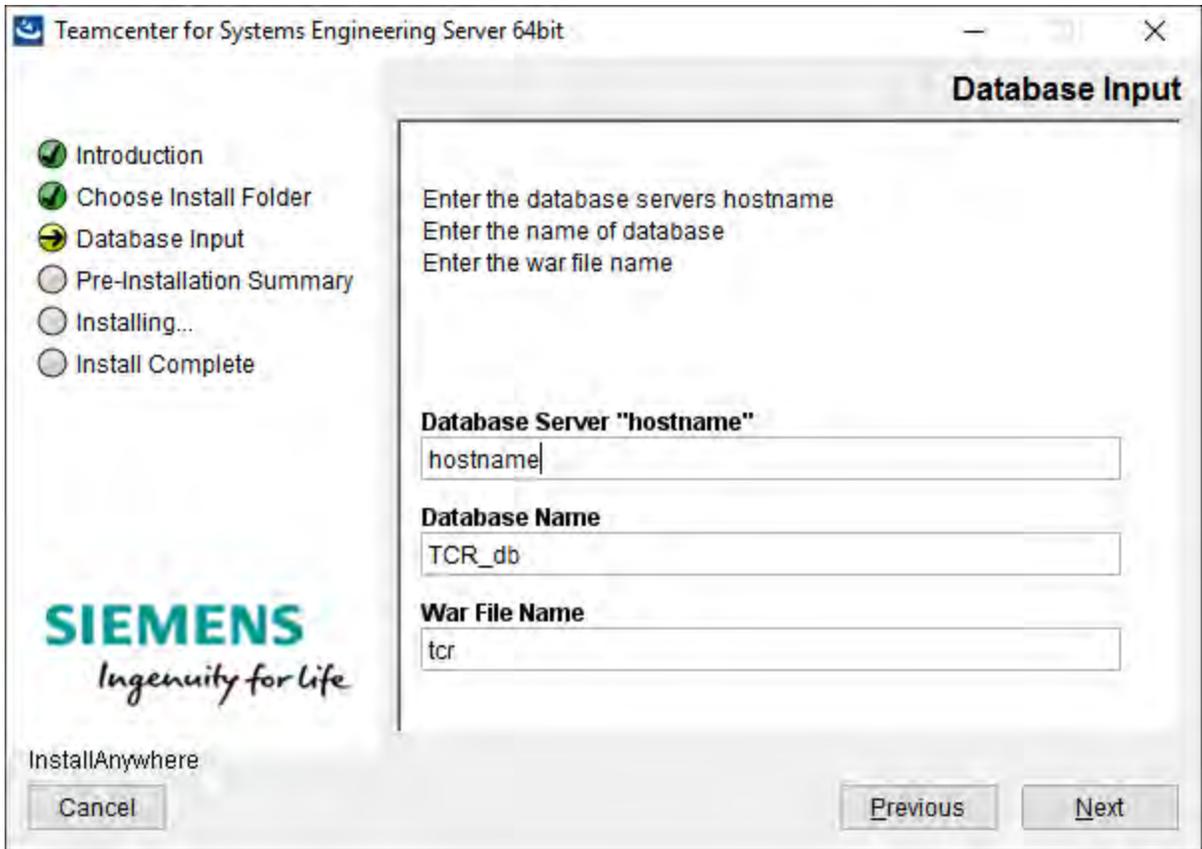


The **Database Server "hostname"** that you specify must be the same name that is used for the Versant installation.

The installer uses a default name **tcr**. If you customized the name of the war file during the original installation, you must specify the same name in the **War File Name** box.

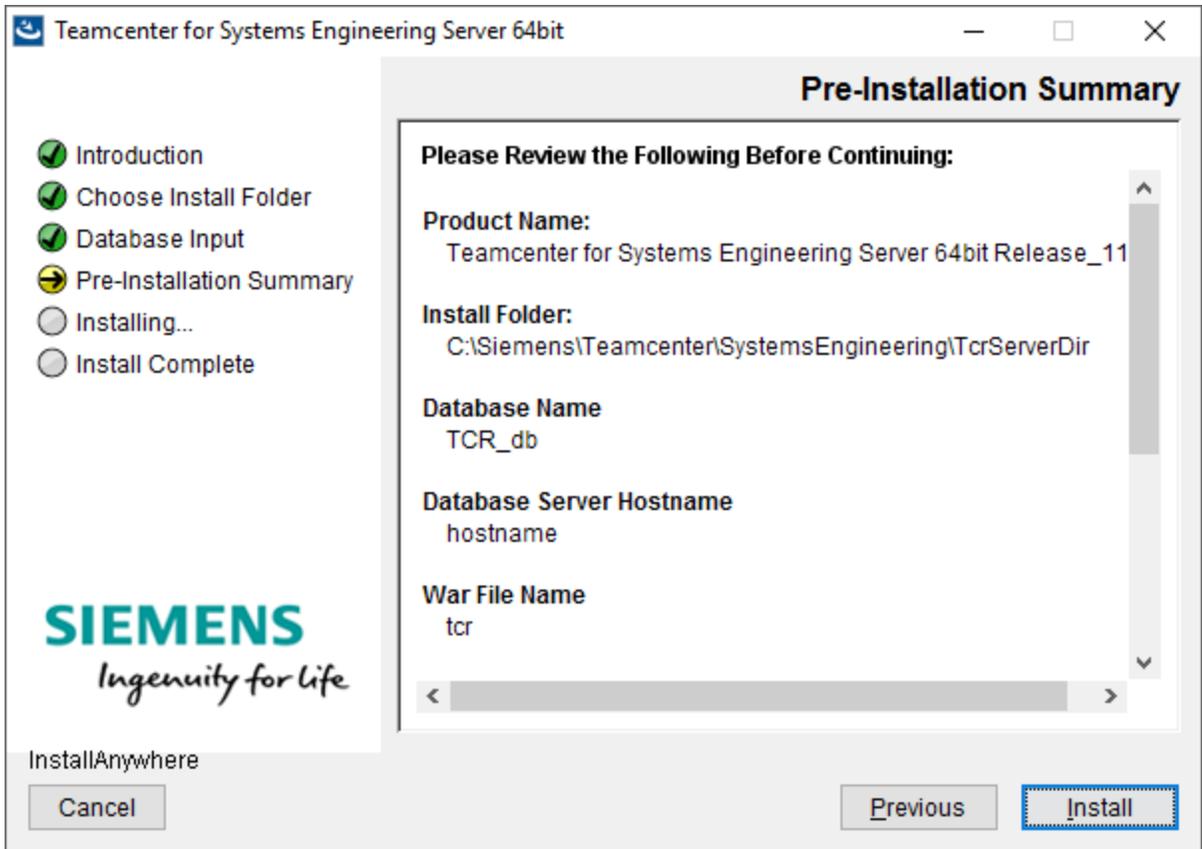
Using a different war file name for the upgrade is not supported.

Click **Next**.

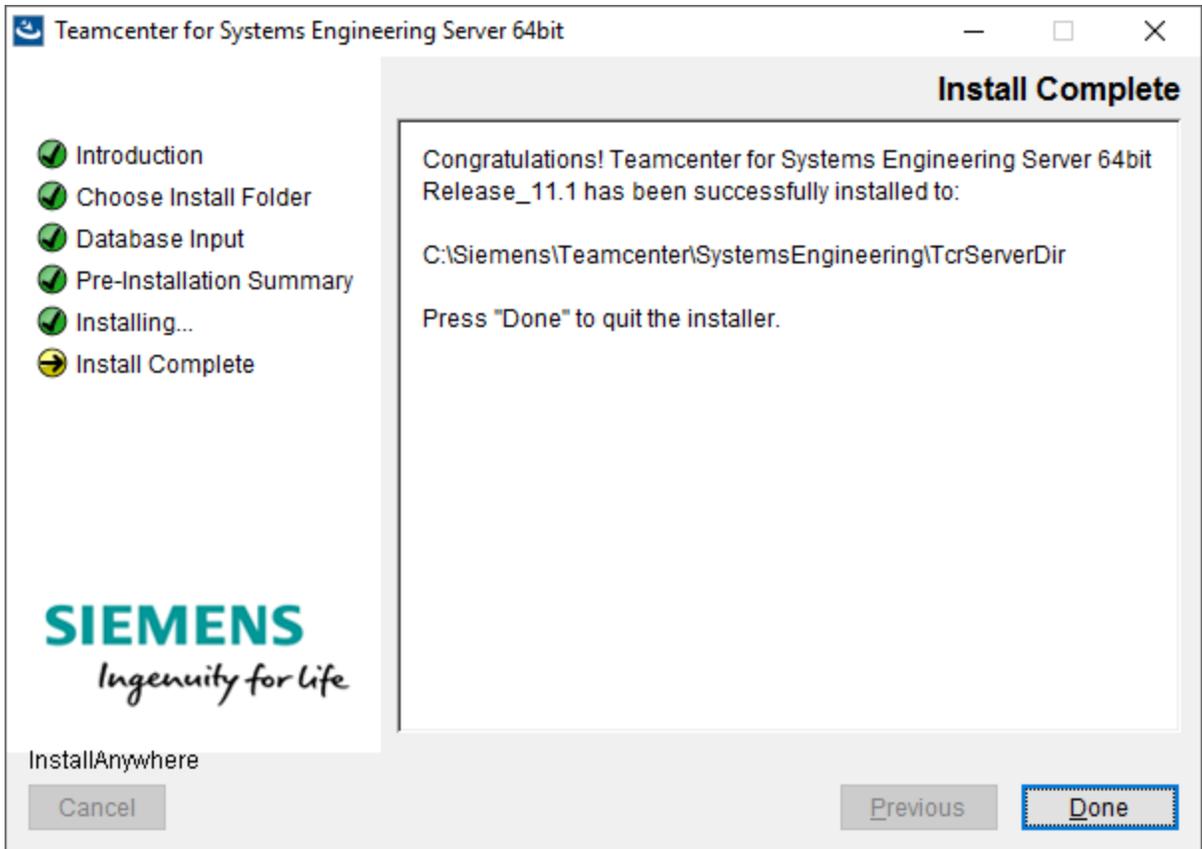


10. The **Pre-Installation Summary** screen displays information such as the product name, installation folder, database name, database server's host name, and disk space availability.

Click **Install** to begin the installation.



11. After the installation is successful, click **Done** to exit the installer.



Copying the Versant License File

The default location of the license file (**license.xml**) is `C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\versant_license\license.xml`. Copy the **license.xml** file located in the **versant_license** folder (in the Architect/Requirements installation folders) to the `VERSANT_ROOT` folder (for example, `C:\versant`). This file is needed to enable Versant.



You cannot create a database if you skip this step.

Upgrading the Database for use with Architect/Requirements 11.1



The steps are also applicable for **Versant Object Database only** machines.

After the installation, you must upgrade the database for use with Architect/Requirements 11.1. Perform the following steps to upgrade the database.

Upgrading the Architect/Requirements Schema



If you intend to move your database to a new server while completing this upgrade, see Appendix B, [Upgrading the Architect/Requirements Schema while moving the database server](#). After you complete the steps in Appendix B, proceed to [Completing the Upgrade](#).

1. Stop the database. Execute the following command at the command prompt:

```
stopdb TCR_db
startdb TCR_db
stopdb TCR_db
```

2. Upgrade the Versant schema

```
cnvrtdb TCR_db
```

3. Upgrade the Architect/Requirements schema.

From the command prompt, change the directory to the schema directory of your server installation. Execute the following command:

```
tcradmin -action upgradeDB -logToStdOut
```

Completing the Upgrade

1. Deploy the Architect/Requirements 11.1 web application **tcr.war** file.

For information on how to deploy the **tcr.war** file, see [Deploying the tcr.war File](#).

2. If you have also moved your database to a new server with this upgrade, you must restore the customer number and Architect/Requirements license information recorded in the prerequisite steps.
3. Upgrade the Architect/Requirements client.

To upgrade the Architect/Requirements client, you must uninstall the previously installed client.

For information on uninstalling the Architect/Requirements client, see *Uninstalling the Architect/Requirements Client* in the *Systems Architect/Requirements Management User's Manual*.



Architect/Requirements 11.1 requires JRE 1.8. Uninstall all previously installed JREs and install the 32-bit Java.

For information about the supported version of Java, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

Install the new Architect/Requirements client.

For information on installing the Architect/Requirements client, see *Installing the Architect/Requirements Client with Office Integration* in the *Systems Architect/Requirements Management User's Manual*.

4. Start an Architect/Requirements client and verify the upgrade.



Elevated user privileges (power user or administrator) are required for the installation as registry entries are updated with the patch information.

For information on how to start the client and perform the verification, see [Verifying the Installation](#).

Running the maintainDB Utility

Siemens PLM Software recommends that you run the **maintainDB** utility after upgrading your database from any earlier version.

To examine and correct the entire database, run the **tcradmin** command. From the command prompt on system 2, change the directory to the schema directory of your server installation. Execute the following command from the command prompt:

```
tcradmin -action maintainDB -logToStdOut
```

For more information about running **maintainDB**, see the *Systems Architect/Requirements Management System Administrator's Manual*.

Next Steps

Continue with the post-upgrade procedures such as:

- Restoring custom **JSP** files.
- Updating the Web application configuration parameters.
- Entering Architect/Requirements license information.
- Upgrading the Architect/Requirements clients.
- Verifying the installation.

- Restoring the customized schema.

The procedures are described in [Post-installation Tasks](#).

Chapter 4: *Post-installation Tasks*

This chapter describes the procedures to be followed after installing or upgrading the Systems Architect/Requirements Management server.

Overview

Some of the post installation or upgrade procedures are:

- Restoring custom JSP files.
- Deploying the **tr.war** file.
- Updating the Web Application Configuration parameters.
- Entering Architect/Requirements license information.
- Upgrading the Architect/Requirements clients.
- Verifying the installation.
- Restoring customized schema.

Restoring Custom JSP Files

Before upgrading to a newer version of Architect/Requirements, you may have backed up all your custom JSP files before undeploying an existing **tcr.war** file. You can restore these custom JSP files to ensure that your customized information is available after upgrading the server software.

The JSP files were located in the custom directory of the application server (such as BEA WebLogic), *application-server-installation-directory\tcr\custom*.

To restore your custom JSP files:

1. Create a temporary folder and extract the **tcr.war** file.
You can use Winzip, or run the command: `jar -xf tcr.war`
2. Add the backed up JSP files to the **custom** folder.
3. To prevent adding the existing **tcr.war** file to the new WAR file, delete or move it (the existing **tcr.war** file) from the temporary folder.
4. Zip the contents of the temporary folder to create the new WAR file.

You can use Winzip, or run the following command:

```
jar -cf tcr.war *
```

Deploying the tcr.war File

The Architect/Requirements Web component (**tcr.war**) contains libraries required for the functioning of the Architect/Requirements server. You should deploy the **tcr.war** file on the application server (such as BEA WebLogic) after installing or upgrading the Architect/Requirements server software.

The Architect/Requirements WAR file (**tcr.war**) is located in the **war_file** directory. The typical location of the **war_file** folder is *C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file*.

Use an appropriate example provided in the following sections to deploy the WAR files. For more instructions specific to your application server, see the documentation provided by the vendor.

Deploying tcr.war File on Oracle WebLogic

Use the examples provided in this section to deploy a WAR file on Oracle WebLogic.

Deployment Process

To deploy the WAR file on Oracle WebLogic, perform the following steps:

1. Edit the **CLASSPATH** variable.
2. Create a WebLogic domain for Systems Architect/Requirements Management.
3. Deploy the WAR file.
4. Restart WebLogic.

Edit the Classpath Variable



If the **CLASSPATH** includes the **jvi.jar** file before WebLogic is launched, deployment of **tcr.war** fails.

Before you start WebLogic, you must ensure that the **jvi.jar** file path is not included in the **CLASSPATH** environment variable. The Architect/Requirements application server must use its own copy of the Versant Object Database's **jvi.jar** file, and a **jvi.jar** file included in the **CLASSPATH** can prevent **tcr.war** from deploying successfully.

For example, if the **CLASSPATH** in your WebLogic environment includes a path to a **jvi.jar**, `C:\Folder1\jvi.jar`, the **tcr.war** deployment will fail:

```
CLASSPATH=C:\Folder1\jvi.jar;C:\Folder2\jarfile2.jar;C:\Folder3\jarfile3.jar
```

The remedy, prior to starting WebLogic, would need to set **CLASSPATH**, excluding `C:\Folder1\jvi.jar`, similar to the following:

```
set CLASSPATH=C:\Folder2\jarfile2.jar;C:\Folder3\jarfile3.jar
```

Create a WebLogic Domain

Before deploying the WAR file, you must create a WebLogic domain.

To create a domain:

1. From the Oracle Fusion Middleware Configuration Wizard, choose **Create a new WebLogic domain** and click **Next** to continue.



Typically, the Configuration Wizard is available at **Start** → **All Programs** → **Oracle** → **Configuration Wizard**.

2. Choose **Generate a domain configured automatically to support the following Oracle products:** and check the **Basic WebLogic Server Domain** check box. Click **Next** to continue.
3. Enter and note the default administrator user name and password for the domain. This information is required to start and stop the WebLogic application server, or to access the WebLogic Administrative Console. Click **Next** to continue.
4. In the **Domain Mode and JDK** screen, choose the **Production** Domain Mode, choose a supported 64-bit Oracle JDK, and click **Next** to continue.
5. In the **Advanced Configuration** screen, check the Administration Server box, and click **Next** to continue.
6. In the **Administration Server** window, enter and note the port number, such as **7010**. Then enter and note the name for the Systems Architect/Requirements Management domain, such as **tcr_7010**, and click **Next**.
7. In the **Configuration Summary** window, verify that the summary, and then click **Create**.
8. Once the **Configuration Progress** window shows 100%, click **Next**.
9. Verify that the **End of Configuration** window indicates that the domain creation succeeded, and click **Finish**.

For the domain in the example above, the WebLogic configuration files and folders are typically created at `C:\oracle\user_projects\domains\tr_7010`.

Deploying the War File

Start the WebLogic application server by running the **Start the Admin Server for WebLogic Server Domain** command. It is typically available at

`C:\oracle\user_projects\domains\domain_name\bin\startWebLogic.cmd`. Enter the default administration user name and password when prompted.

Launch the Oracle WebLogic administrative console and log on using the default administration user name and password. For a domain created with the port **7010**, the administrative console can be typically launched using **http://hostname:7010/console**.

To deploy the war file:

1. In the **Change Center** section in the left pane, click the **Lock & Edit** button to modify, add, or delete items in this domain.
2. In the **Domain Structure** section in the left pane, click **Deployments**. The **Summary of Deployments** page is displayed. Click **Install** under **Summary of Deployments**.
3. On the **Install Application Assistant** pane, browse to the location of the WAR file. The Architect/Requirements WAR file (**tr.war**) is located in the **war_file** directory. The typical location of the **war_file** folder is `C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file`.
4. Choose the WAR file and click **Next** to continue. The processing usually takes a few seconds. You must wait until the hour glass cursor disappears and the **Install Application Assistant** pane is displayed.
5. Choose the **Install this deployment as an application** installation type and click **Next**.
6. On the **Install Application Assistant** pane, click **Finish**.
7. In the **Change Center** section in the left pane, click **Activate Changes** to activate the pending changes.
8. Restart WebLogic.
9. When the restart is complete, login once again to the WebLogic administration console.
10. In the **Domain Structure** section in the left pane, click **Deployments**. The **Summary of Deployments** page is displayed.
11. Select the **Control** tab under **Summary of Deployments**, and select the check box next to the WAR file (for example, **tr**). Click the arrow next to the **Start** button and select the **Servicing all requests** option.
12. In the **Start Application Assistant** pane, click **Yes**.
After the processing is successful, the state column next to the WAR file (for example, **tr**) is **Active**.
Log out of the WebLogic administrative console.

Deploying tcr.war File on IBM WebSphere

The examples in this section are for a typical WebSphere installation. If you are not an experienced Web server administrator, Siemens PLM Software recommends that you do only the actions in these examples and retain other default WebSphere settings.

To deploy the WAR file on IBM WebSphere, perform the following steps:

1. Deploy the WAR file.
2. Restart WebSphere.
3. Install the security JAR files that are required for Systems Architect/Requirements Management licensing.

Deploying the War File

To deploy the war file:

1. Log on to the WebSphere application server administration console.
By default, the IBM WebSphere administrative console is run on Port 9060:
http://localhost:9060/ibm/console.
2. In the left pane, expand **Applications**→**Application Types**, and then click the **Websphere enterprise applications** link.
3. In the **Enterprise Applications** pane, click the **Install** button.
4. In the **Preparing for the application installation** pane, select **Local file system**, click **Browse** and select the Architect/Requirements WAR file.

The Architect/Requirements WAR file (**tcr.war**) is located in the **war_file** directory. The typical location of the **war_file** folder is

C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file.

5. In the **Preparing for the application installation** pane, select **Fast Path**, and click **Next**.
6. In the **Install New Application** pane, for **Step1: Select installation options**, click **Next**.
7. In the **Install New Application** pane, for **Step 2: Map modules to servers**, select the **Teamcenter 11 for systems engineering** check box and click **Next**.
8. In the **Install New Application** pane, for **Step 3: Map virtual hosts for Web modules**, select the **Teamcenter 11 for systems engineering** check box and click **Next**.
9. In the **Install New Application** pane, for **Step 4: Map context roots for Web modules**, enter **/tcr** as the **Context Root** and click **Next**.
10. In the **Install New Application** pane, for **Step 5: Summary**, click **Finish**.

This action initiates the deployment process, which may take several minutes. After the WAR file is deployed, the installation summary is displayed, and it includes a message such as:

`Application tcr_war installed successfully.`

11. Click the **Save Directly to Master Configuration** link to save the settings for Systems Architect/Requirements Management.

12. Restart WebSphere.
13. In the left pane, expand **Applications**→**Application Types**, and then click the **Websphere enterprise applications** link.
14. Check that the application (for example, **tcr_war**) is started.

The **Application Status** must display a green arrow. Systems Architect/Requirements Management is deployed and running.

If it is not started, a red cross is displayed as the **Application Status**. Select the check box in front of the application and click **Start**.

Installing the Security JAR Files

To install the security JAR files, perform the following steps:

1. Open the main Architect/Requirements login page.
2. Click the **Administrative Tools** link. On the Administrative Tools page, click the **Web Application Configuration** link.
3. Log on as an enterprise administrator user (typically **tcradm**).
4. Scroll to the bottom of the page and click the **Install Security Files** button.
5. On the page displaying the **From** and **To** locations, the **To** text box must include the only the complete path to the server JRE's **lib/ext** folder. For example, **D:\apps\IBM\WebSphere\AppServer\java\jre\lib\ext** . Delete any additional path information or semicolons.
Click **OK**.
6. After the process completes, click the **Close** button.

Deploying the tcr.war File on Apache Tomcat

This section provides procedure examples for deploying the WAR file on Apache Tomcat.



The examples in this section are for a typical Tomcat installation. If you are not an experienced Web server administrator, Siemens PLM Software recommends that you do only the actions in these examples and leave all other Tomcat settings at the defaults.

To deploy the WAR file on Apache Tomcat:

1. Install the Java Development Kit (JDK).
2. Install Apache Tomcat.
3. Deploy the WAR file.

Installing JDK

Install JDK, using the installation program provided by Oracle.

For information about versions of the JDK supported for Apache Tomcat, see the Siemens PLM Software Certification Database:

http://www.plm.automation.siemens.com/en_us/support/gtac/certifications.shtml and see the **Teamcenter Systems Engineering Software Certifications** section.

Installing Apache Tomcat

Install Tomcat, using the installation program provided by Apache.



The JDK must be installed first, because the Tomcat installation program detects the JDK version on your system. If the wrong location is detected, cancel the installation and set the **JAVA_HOME** environment variable to the location of JDK on your system. Tomcat then uses this location.

Deploying the WAR File

When both JDK and Tomcat are installed, you can deploy the WAR file.

1. Copy the WAR file from the Architect/Requirements installation directory to the **TOMCAT_HOME\Webapps** folder.

The Architect/Requirements WAR file (**tcr.war**) is located in the **war_file** directory. The typical location of the **war_file** folder is

C:\Siemens\Teamcenter\SystemEngineering\TcrServerDir\war_file.

The WAR file should be copied to the **Webapps** folder within your Tomcat directory structure.

2. Deploy the WAR file on Tomcat.

Ensure that Tomcat is running. Tomcat expands the WAR file automatically. Under the **Webapps** directory, Tomcat creates a directory named **tcr** with the contents of the WAR file.

3. Restart Tomcat.

The Apache Tomcat application server caches **.jsp** files that it compiles at runtime under the **tomcat_root\work** folder. In some cases, simply deleting and re-deploying the WAR file does not force

Tomcat to reload these cached files. This causes Architect/Requirements to run with the older versions of **.jsp** files from previous releases.

As a workaround:

1. Delete the old WAR file and the deployment folder.
2. Delete the **trc** folder, typically found at *tomcat_root\work\User\localhost\trc* where *User* is the user name.
3. Deploy the Architect/Requirements WAR file.
4. Restart Tomcat.

Updating the Web Application Configuration Parameters

If you are upgrading your installation, update the Web application configuration parameters by performing the following steps:

1. Open the main Architect/Requirements login page.
2. Click the **Administrative Tools** link. On the Administrative Tools page, click the **Web Application Configuration** link.
3. Log on as an enterprise administrator user (typically **tradm**).
4. Scroll down to the **JRE.Version** parameter, and select the **Reset to Default** check box to update the list of JRE software supported for this release.

The **Reset to Default** check box is not available in case of new installation.

5. Scroll to the bottom of the page and click the **Update** button. The updated values of the **JRE.Version** parameter are displayed.
6. Click **OK**.

Entering Architect/Requirements License Information

You receive license information in an e-mail message from Siemens PLM Software Customer Support when your Architect/Requirements application is registered.

The license information includes:

- Your customer number, located on the **Installation No** line in the message body.
-

Your license key, contained in the license file named **trc.lic** that is attached to the message.

The license key is an encrypted text string that controls a certain number of licenses and license types. The key contains information such as your customer number, an expiration date, and the number of seats for each license type.

Enter this license information after the **trc.war** file is deployed and the database is initialized or upgraded.

For more information about managing licenses, see the *Systems Architect/Requirements Management System Administrator's Manual*.

Entering Your Customer Number

Your customer number is located on the **Installation No** line in the e-mail message that you receive from Siemens PLM Software Customer Support.

1. In Microsoft Internet Explorer, open the Architect/Requirements home page, and then click the **Administrative Tools** link.
2. On the Administrative Tools page, click the **Web Application Configuration** link.

The Architect/Requirements login page is displayed.

3.

Enter **tcradm** in the **User Name** field, and then click **Log In**.



If your server is newly upgraded, the password for the **tcradm** user is unchanged. For a newly installed server, leave the **Password** field blank. A password is not required as the database is newly initialized.

The Configuration Parameters page is displayed.



You can also display the Configuration Parameters page by entering the following URL in the Internet Explorer **Address** field.

```
http://server:port/tcr/ugs/tc/req/configtcr.jsp
```

4.

Locate the **LIC.CustomerNumber** parameter, and then enter your customer number in the corresponding **Current Database Value** field.

5. At the bottom of the page, click **Update**.

A confirmation page displays the proposed and current values for the parameter.

6. To commit the proposed value to the database, click **Ok**.

The Configuration Parameters page is displayed, with your customer number as the **LIC.CustomerNumber** parameter value.

For more information about managing licenses, see the *Systems Architect/Requirements Management System Administrator's Manual*.

Entering Your License Key

Your license key is in the license file named **tcr.lic**. This file is attached to the e-mail message that you receive from Siemens PLM Software Customer Support.



Your customer number must be entered before you start this procedure. For more information, see [Entering Your Customer Number](#).

1. On the Architect/Requirements home page, click the **Administrative Tools** link.

The Administrative Tools page is displayed.

2. Click the **TcSE Licensing** link.

The License Information page is displayed, with a summary of the license count from all license keys.

3.

Click the **Manage Licenses** link.

The Architect/Requirements login page is displayed.

4.

Enter your enterprise administrator user name and password, select a language, and click **Log In**.

The License Management page is displayed.



If your database is new and if your customer number is not entered, an alternate page is displayed. This page contains a link to the Web Application Configuration page.

Click that link, enter your customer number in the **LIC.CustomerNumber** parameter, and click the **Update** button. Then repeat this procedure.

5. Open the **tcr.lic** file in a text editor (for example, Microsoft Notepad), and copy the encrypted string to the clipboard.

6. In the text field at the bottom of the License Management page, delete the words **Enter Key Here** and paste the encrypted string into the field.



You can reverse this action by clicking **Clear**.

7. Click **Add Key**.

The license management utility checks the license key. When the license key is validated, a confirmation page displays the license key information in unencrypted format.

Click **Back** to return to the License Management page, where you can enter the key again or enter another key.



If the license key is invalid, expired, or a duplicate, an error message is displayed.

8. Click **Add**.

The license key is committed to the database. The License Management page displays the new license key in the table of encrypted strings.



If you have two or more Web application servers that point to the same database, and if you manage each server separately through the **Installation Key** parameter in the **web.xml** file, you must add this license key on each server.

Upgrading the Architect/Requirements Client

After you have completed the server upgrade, you must upgrade all the Architect/Requirements client.

Elevated user privileges (power user or administrator) are required for uninstalling the Architect/Requirements client.

To upgrade an Architect/Requirements client

1. Uninstall the existing client.

For information on uninstalling the Architect/Requirements client, see *Uninstalling the Architect/Requirements Client* in the *Systems Architect/Requirements Management User's Manual*.

2. Install the latest Architect/Requirements client.

For information on installing the Architect/Requirements client, see *Installing the Architect/Requirements Client with Office Integration* in the *Systems Architect/Requirements Management User's Manual*.

Verifying the Installation

Verifying the Installation on the Server

After the patch is deployed, you can verify the Architect/Requirements Server patch version.

To verify the Architect/Requirements Server patch version:

1. Open the Architect/Requirements client home page.
2. Click **Administrative Tools**.
3. Click **Diagnostic Tools**.
4. Click **Server Version**.

Ensure that the server patch version for the Architect/Requirements war file and **tcrServer.jar** file is displayed as **11.1.0**.

To verify that the correct version of the **tcrServer.jar** was copied to your schema directory, navigate to the schema directory and execute the following command:

```
tcradmin -action showVersion logToStdOut
```

Verify that the following details are displayed on the screen:

```
Product-Version: Release_11.1  
Product-Client-Patch: 11.1.0  
Product-Server-Patch: 11.1.0
```

In addition to the **Server Version** diagnostics, there are additional diagnostic utilities that provide information about the environment in which the Systems Architect/Requirements Management server is installed. This information helps you troubleshoot problems with the installation.

For detailed information about using these utilities, see *Running System Utilities* in the *Systems Architect/Requirements Management System Administrator's Manual*.

Verifying the Installation on the Client

You verify the Systems Architect/Requirements Management server installation by running the Systems Architect/Requirements Management client. If the client connects to the database, the installation is successful.

For detailed information about installing the client, see the *Systems Architect/Requirements Management User's Manual*.

To run the Systems Architect/Requirements Management client and verify the version:

1. Verify that the Systems Architect/Requirements Management application server is running.
Record the name and port number of the application server.

2. In the Microsoft Internet Explorer **Address** field, enter the URL of the Systems Architect/Requirements Management home page.

For example, if your host is named **MyHost** and is running on port **8080**, you would enter the following:

```
http://MyHost:8080/tcr
```

3. Click the **Launch Teamcenter systems engineering** link.
4. In the **User Name** and **Password** fields, enter your Systems Architect/Requirements Management user name and password.

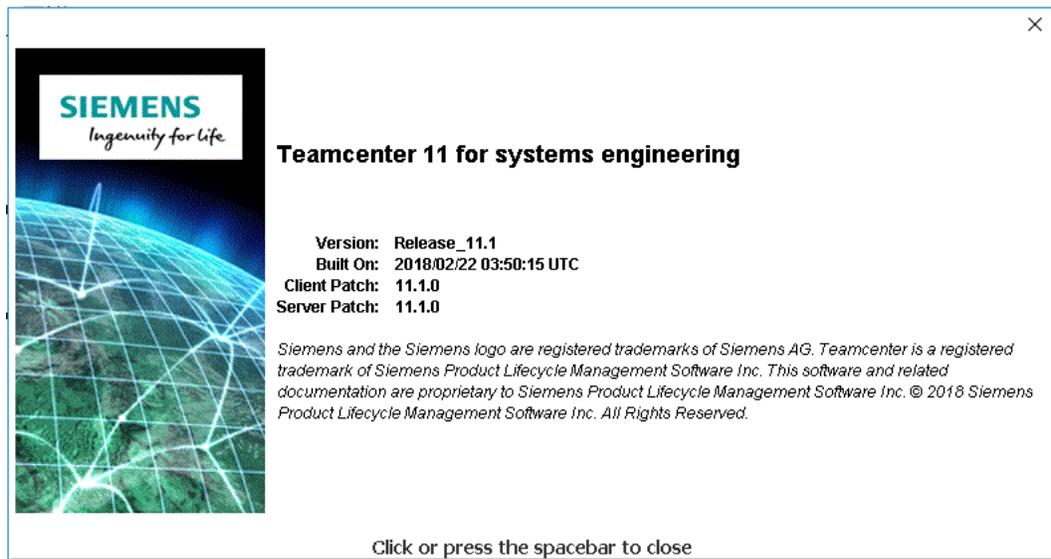
If you have just created the Systems Architect/Requirements Management database, enter **tcradm** as the user name. This is the default user name created at the time of initializing the Architect/Requirements database and it has no password.

5. Click **Login**, or press the enter key.
6. Follow the subsequent instructions in the installation wizard.

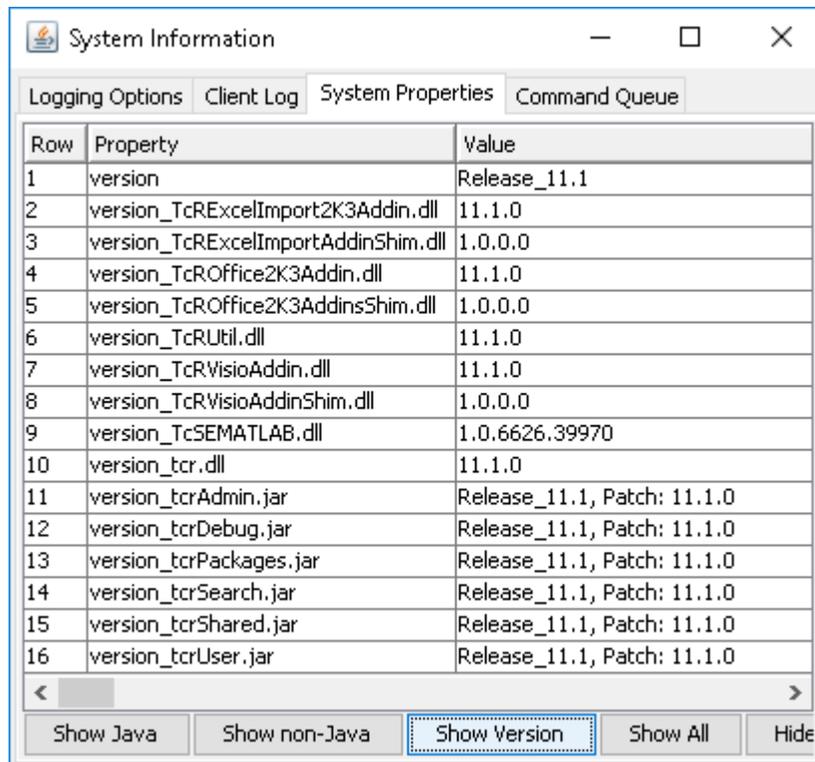


One of the JRE versions as defined in the **JRE.Version** Web configuration parameter must be present on the client machine. If that version is not present on the client machine, the Architect/Requirements server prompts you to install it the first time you run Architect/Requirements.

7. After the Architect/Requirements client is launched, verify the version by either of the following methods:
 - Click **Help**→**About** to view the **About** box. The splash screen displays the patch version as **11.1**.



- View the **System Properties** by clicking **Tools**→**System Information**→**System Properties**→**Show Version**. Architect/Requirements displays the version information for files installed on the client computers.



Restoring Customized Schema

Before upgrading to a newer version of Architect/Requirements, you may have exported the locally modified schema objects supplied by Architect/Requirements.

You may perform an XML schema import of those objects.

Importing Schema for Folder Move Confirmation

After you have verified the installation, you can optionally add schema to provide an optional confirmation when users attempt to move a top-level folder. See PR 6847222 for details.

To Import Schema to provide Folder Move Confirmation

1. Log on to Architect/Requirements as a user with the Enterprise Administrator privilege.
2. Navigate to the **Administration** module.
3. Right-click the **TcR Administration** project.
4. From the menu, select **Import**→**Import Schema**.
5. Select the **AdminProjectSchema.xml** file, extracted from the **TcSE_11.1.zip** file to the temporary folder and click **OK**.

Chapter 5: Uninstalling Architect/Requirements

This chapter describes the situations when you may need to uninstall the Systems Architect/Requirements Management server software. A reference to the detailed instructions is also provided.

Uninstalling the Server Software

In some situations, you may want to uninstall the Systems Architect/Requirements Management server software from your application server. For example:

- To move Architect/Requirements to a different server.
- To reinstall Architect/Requirements in a different location on the same server.
- To upgrade to a new version of Systems Architect/Requirements Management.
- When your evaluation is concluded.

The instruction to uninstall the Systems Architect/Requirements Management server is explained in [Uninstalling Architect/Requirements Web Applications](#).

Appendix A: Directories and Files Created During Server Installation

The Systems Architect/Requirements Management installation program creates the following directories under the Systems Architect/Requirements Management server directory:

- **Uninstall**
This directory contains files needed to uninstall the Systems Architect/Requirements Management server.
- **schema**
This directory contains the files necessary for administrative functions such as creating a new database and maintenance utilities such as **MaintainDB**.
The main script is **tcradmin**. The rest are supporting Java files. The **version.txt** file contains the Architect/Requirements version that these files are built from.
- **security_jars**
This directory contains Java files necessary for the licensing of Systems Architect/Requirements Management to function properly on WebSphere.
- **versant_license**
This directory contains the **license.xml** file. This file must be copied to your **VERSANT_ROOT** directory.
- **war_file**
Contains the Systems Architect/Requirements Management Web archive. This file must be copied to or deployed on your application server.
- **jvi_root**
Contains the Versant client runtime files used by the Systems Architect/Requirements Management server.

The following files are at the root level:

- **Teamcenter_for_Systems_Engineering_Server_64bit_VERSION_InstallLog.log**: This is the installation log file.

Appendix B: Upgrading the Architect/Requirements Schema while moving the database server

This appendix describes how to upgrade the Architect/Requirements schema while moving a database server. It is a useful if you need to move your database server to meet certification requirements on the upgraded system.



The following assumptions are made in this section:

- The name of the Architect/Requirements server war file is **tr.war**. While the Architect/Requirements installer allows you to use any name, **tr.war** is the typically used name.
- The name of the Architect/Requirements Versant object database is **TCR_db**. While the Architect/Requirements installer allows you to use any name, **TCR_db** is the typically used name.
- The *system 1* refers to the system as it exists prior to upgrade procedure, and the *system 2* refers to the system as it exists following the upgrade procedure.
- Commands executed on Window servers must be executed from an Administrator command prompt.

Ensuring unique database numbering

If this is the first database being moved to System 2, you must ensure that the Versant database IDs created on System 2 begin with a new numbering series. To do this :



You must perform this step only once. If you have performed this step previously, move on to the next appropriate section.

1. Determine the highest number used on System 1 to date by running the *dblist* command with no arguments at the System 1 command prompt:
2. From the output, you will see that the number for each database is shown in the “ID” field.
3. Record the highest database ID number viewed from the output.
4. Set the threshold database ID on System 2 by entering the following commands on the System 2 command prompt.:

```
dblist
```

```
makedb -nofeprofile TempDB
```

```
createdb TempDB
```

```
setdbid <HighestDatabaseIDNumberOnSolaris> TempDB
```

```
removedb -rmdir TempDB
```

System 1 running on Solaris Server

If you intend to move a database from a Solaris server to Windows server, complete the steps that follow. Otherwise, proceed to the next section.

1. If you are upgrading from a previous release, you must first install Versant Object Database 8.0.2 on System 2. Please refer to Chapter 2 of the Architect/Requirements 10.1 installation manual for details.



To perform this step, System 2 must be running on a Windows Server supported by Architect/Requirements 10.1. Specifically:

- If you intend to move the database to Windows Server 2012 R2, you can do so by simply completing the procedure in this section, since it is supported by both Architect/Requirements 10.1 and Architect/Requirements 11.1.
- If you intend to move the database to Windows Server 2016, you must first move the database to a Windows Server supported by Architect/Requirements 10.1 by completing the procedure in this section. Once that is complete, you can then move to Windows Server 2016 by performing a second database move per the next section, [System 1 running on Windows Server](#).

2. Ensure that the user performing the upgrade process on system 2 has the Versant DBA role assigned on the system 1 database. To check the user's existing status, execute the following command on the database server used by system 1:

```
dbuser -list TCR_db
```

If the user is not assigned the DBA role, execute the following command on the database server used by system 1:

```
dbuser -add -n <UpgradeUserName> -role DBA TCR_db
```

3. Note down your **Customer Number** and **License Key** information using the **Web Application Configuration** and **TcSE Licensing links** on the **Administrative Tools** page of system 1.

As the system 2 database is on a different machine, you need to enter the information while deploying the **tcr.war** file.

4. To create the database directory, execute the following command from the command prompt of system 2:

```
makedb -nofeprofile TCR_db
```

5. Copy the **profile.be** file from the system 1 **TCR_db** directory to the system 2 database directory.



The upgrade process causes a small increase in the database size.

Ensure that the Versant system volume, as specified by the **sysvol** parameter in **profile.be**, is large enough to accommodate the database that you are copying. For example, if the size of the database is **10 GB**, the **sysvol** must be a minimum of **12000M** including **20%** room for growth.

The **profile.be** syntax is **M** for **megabytes**.

6. To create the database, execute the following command from the command prompt of system 2:

```
createdb -i TCR_db
```

7. Ensure that the user who owns the database on system 1 has the Versant DBA role assigned on the system 2 database. To check the user's existing status, execute the following command on system 2:

```
dbuser -list TCR_db
```

If the user is not assigned the DBA role, execute the following command on system 2:

```
dbuser -add -n <System1DatabaseOwnerUserName> -role DBA TCR_db
```

8. Copy the database from system 1 to system 2 by executing the following command on system 2:

```
vcopydb -nolock -nocreate -optimize TCR_db@<System1> TCR_db
```

9. Copy the **profile.be** file from system 1 **TCR_db** directory to system 2 database directory.



The previous step overwrites the **profile.be** file so you must copy it again.

Ensure that the Versant system volume, as specified by the **sysvol** parameter in **profile.be**, is large enough to accommodate the database that you are copying. For example, if the size of the database is **10 GB**, the **sysvol** must be a minimum of **12000M** including **20%** room for growth.

The **profile.be** syntax is **M** for **megabytes**.

10. Stop the database. Execute the following command at the command prompt of system 2:

```
stopdb TCR_db  
startdb TCR_db  
stopdb TCR_db
```

11. If you are upgrading from a previous release, you must now uninstall the Versant Object Database 8.0.2 from system 2, and install Versant Object Database 9 in its place. Please refer to Chapter 3 of this manual for details.

12. Upgrade the Versant schema. Execute the following command at the command prompt of system 2:

```
cnvrtdb TCR_db
```

13. Create the session user. From the command prompt on system 2, change the directory to the schema directory of your server installation. Execute the following command from the command prompt of system 2:

```
tcradmin -action createSessionUser -logToStdOut
```

14. Upgrade the Architect/Requirements schema. Execute the following command on system 2:

```
tcradmin -action upgradeDB -logToStdOut
```

System 1 running on Windows Server

Complete the steps that follow if you intend to move a database from a Windows server to different Windows server:

1. If you are upgrading from a previous release, you must now uninstall the Versant Object Database 8.0.2 from system 1, and install Versant Object Database 9 in its place. Please refer to Chapter 3 of this manual for details.
2. Ensure that the user performing the upgrade process has the Versant DBA role assigned on the system 1 database. To check the user's existing status, execute the following command on the database server used by system 1:

```
dbuser -list TCR_db
```

If the user is not assigned the DBA role, execute the following command on the database server used by system 1:

```
dbuser -add -n <UpgradeUserName> -role DBA TCR_db
```

3. Note down your **Customer Number** and **License Key** information using the **Web Application Configuration** and **TcSE Licensing** links on the **Administrative Tools** page of system 1.

As the system 2 database is on a different machine, you need to enter the information while deploying the **tcr.war** file.

4. To create the database directory, execute the following command from the command prompt of system 2:

```
makedb -nofeprofile TCR_db
```

5. Copy the **profile.be** file from system 1 **TCR_db** directory to the system 2 database directory.



The upgrade process causes a small increase in the database size.

Ensure that the Versant system volume, as specified by the **sysvol** parameter in **profile.be**, is large enough to accommodate the database that you are copying. For example, if the size of the database is **10 GB**, the **sysvol** must be a minimum of **12000M** including **20%** room for growth.

The **profile.be** syntax is **M** for **megabytes**.

6. Copy the database from system 1 to system 2 by executing the following command on the system 2:

```
vcopydb -nolock -i -optimize TCR_db@<System1> TCR_db
```

7. Copy the **profile.be** file from system 1 **TCR_db** directory to system 2 database directory.



The previous step overwrites the **profile.be** file so you must copy it again.

Ensure that the Versant system volume, as specified by the **sysvol** parameter in **profile.be**, is large enough to accommodate the database that you are copying. For example, if the size of the database is **10 GB**, the **sysvol** must be a minimum of **12000M** including **20%** room for growth.

The **profile.be** syntax is **M** for **megabytes**.

8. Stop the database. Execute the following command at the command prompt of system 2:

```
stopdb TCR_db  
startdb TCR_db  
stopdb TCR_db
```

9. Upgrade the Versant schema. Execute the following command at the command prompt of system 2:

```
cnvrtdb TCR_db
```

10. Create the session user. From the command prompt on system 2, change the directory to the schema directory of your server installation. Execute the following command from the command prompt of system 2:

```
tcradmin -action createSessionUser -logToStdOut
```

11. Upgrade the Architect/Requirements schema. Execute the following command on system 2:

```
tcradmin -action upgradeDB -logToStdOut
```

Index

Architect/Requirements Client	
Installation in Silent Mode.....	24
Browser and dialog window examples	12
Client component	15
Code conventions.....	13
Command line entry conventions	13
Commands	
stopdb TCR_db.....	52
VERSANT_ROOT	44
Components	
Client.....	15
Database server	
Description.....	15
Web component	
Description.....	15
Components of Architect/Requirements	15
Configuration Parameters page	
Displaying.....	84
LIC. Customer Number parameter.....	84
Conventions	
Browser and dialog window examples	12
Code	13
Command line entries	13
File contents.....	13
Names	12
Revisions.....	12
Values	12
Copying	
Versant License.....	43
Create WebLogic Domain	78
Creating Systems Architect/Requirements	
Management database.....	43
Customer number, entering.....	84
Database	
Creating.....	43
Initializing.....	43
Testing	44
Database server component	
Description.....	15
Deployment example, WAR file	
Oracle WebLogic.....	77
Deployment examples, WAR file	
Apache Tomcat.....	82
IBM WebSphere.....	80
Dialog Window examples.....	12
Directories	93
VERSANT_ROOT.....	44
Webapps	82
Disk space requirements.....	17
Entering	
Customer number	84
License key	84
Enterprise Administrator privilege	44
Environment variables	
JAVA_HOME	82
File contents conventions	13
Files	93
tcr.lic	
License key, entering.....	84
WAR	
Deployment examples, Apache Tomcat ..	82
Deployment examples, IBM WebSphere	80
Deployment examples, Oracle WebLogic	77
Hardware requirements.....	17
Home page, Systems Architect/Requirements	
Management	87
IBM WebSphere	
Deploying the War file	80
Installing the security JAR files.....	81
Initializing Systems Architect/Requirements	
Management database.....	43
Installation	
Testing	
Database.....	44
Verifying.....	87
Installing Architect/Requirements Client in	
Silent Mode.....	24
Installing the Server.....	38, 53, 65
Upgrade	53, 65
JAVA_HOME variable	82
LIC. Customer Number parameter	84
License	
Customer number, entering	84
Key, entering	84
tcr.lic file.....	83
License Information page	85

License key, entering	84	Running the maintainDB Utility	72
License Management page.....	85	Updating the Web Application	
Memory requirements.....	17	Configuration Parameters	83
Name conventions.....	12	Versant License, copying.....	71
Oracle WebLogic		Software Requirements.....	18
Deploying the War file.....	79	Uninstalling server software	51
Overview of Installation	21	Upgrade Overview.....	47
Parameters		Verifying installation.....	87
LIC. Customer Number	84	tcrlc file.....	83
Post Upgrade		tcradm user ID	44, 87
Running the maintainDB Utility.....	72	Testing	
Updating the Web Application Configuration		Systems Architect/Requirements Management	
Parameters.....	83	database	44
Versant License, copying.....	71	Uninstalling	
Restoring Custom JSP Files.....	77	Web server.....	51
Restoring Customized Schema	89	Uninstalling Systems Architect/Requirements	
Revision conventions.....	12	Management server software.....	51
root user ID	44	Uninstalling Web and Database Servers.....	51
Running		Uninstalling Web Servers.....	51
Database initialization program	43	Upgrade Overview.....	47
Systems Architect/Requirements Management		Upgrading From Version 10.0 or later	48
client.....	87	User ID	
Server, Installation		root.....	44
Deploying Architect/Requirements 10.1 on		System administrator	44, 87
Tomcat	82	tcradm	44, 87
Setting environment variables.....	43, 65	User privilege, Enterprise Administrator.....	44
Software Requirements	18	Value conventions	12
stopdb TCR_db command	52	Verifying installation.....	87
System administrator user ID.....	44, 87	Versant	
Systems Architect/Requirements Management		Environment variables.....	43, 65
Components	15	Versant 64-bit Installation on Windows	
Client.....	15	Requirements	18
Database server, description	15	Versant Object Database Installation	
Web component, description.....	15	Installing on Windows.....	25, 55
Database		New Installation.....	25, 55
Creating.....	43	Overview	25, 54
Initializing.....	43	Post-Versant Installation Steps.....	71
Testing.....	44	Post-Versant Installation Steps	35
Hardware requirements.....	17	Verifying the Versant Installation	35, 63
Home page	87	VERSANT_ROOT command	44
Installing		VERSANT_ROOT directory.....	44
Versant License.....	43	WAR file	
Installing the Server	38, 53, 65	Deployment examples	
Upgrade.....	53, 65	Apache Tomcat.....	82
License		IBM WebSphere	80
Customer number, entering.....	84	Oracle WebLogic.....	77
Key, entering.....	84	Web component	
License file.....	83	Description.....	15
Overview of Installation	21	Web server	
Overview of Uninstallation.....	91	Uninstalling	51
Post Upgrade		Webapps directory.....	82

